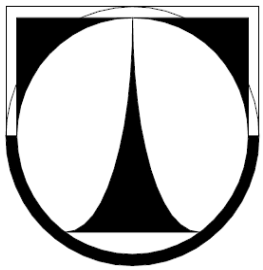
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií



BAKALÁŘSKÁ PRÁCE

Liberec 2011 **Radek Luňák**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

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

Záznam a vizualizace dat měření v prostředí .NET Compact Framework

Data Acquisition and Visualization of Measurings in .NET Compact Framework Environment

Bakalářská práce

Autor: **Radek Luňák**
Vedoucí: Ing. Tomáš Tobiška
Konzultant: Ing. Jan Kraus

V Liberci 19. 5. 2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení: **Radek LUŇÁK**
Osobní číslo: **M08000144**
Studijní program: **B2646 Informační tehnologie**
Studijní obor: **Informační technologie**
Název tématu: **Záznam a vizualizace dat měření v prostředí .NET Compact Framework**
Zadávající katedra: **Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s možnostmi .NET Compact Framework pro vývoj embedded aplikací.

2. Seznamte se se strukturou dat a komunikačním protokolem měřicích přístrojů fy. KMB, s podpůrnými knihovnami a s formátem souborů CEA.

3. Navrhněte a implementujte aplikaci pro archivaci a jednoduchou vizualizaci měření pro obecné zařízení s platformou .NET Compact Framework.

4. Ověřte správnost implementace a uveďte další možnosti rozvoje této aplikace.

Prohlášení

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

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou 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 bych chtěl poděkovat vedoucímu bakalářské práce Ing. Tomáši Tobiškovi a konzultantu Ing. Janu Krausovi za konzultace a užitečné rady k řešení. Firmě KMB Systems děkuji za zapůjčení měřícího přístroje a PDA. Také bych chtěl poděkovat své rodině za umožnění studia a za jejich podporu.

Abstrakt

Tato práce vznikla za účelem rozšířit software pro práci s měřícími přístroji firmy KMB Systems i na mobilní platformu Windows Mobile. Rešeršní část popisuje prostředí mobilních zařízení a vývojové prostředí Compact Framework. Dále informuje o používaných komunikačních protokolech podporovaných zmiňovanými měřícími přístroji, popisuje podpůrné knihovny pro snazší práci v tomto prostředí a uvádí možnosti ukládání měřených dat. V praktické části jsem řešil rozvržení aplikace, způsob vykreslení fázorového diagramu, výběr a stahování naměřených záznamů. Pro použití podpůrných knihoven bylo nutné je uzpůsobit chodu v prostředí Compact Framework. Na konci zprávy je srovnání s existující desktopovou aplikací.

Klíčová slova: .NET Compact Framework, mobilní zařízení, protokol KmbLong, knihovna KMB-Lib, CEA

Abstract

This work was created for purpose of extending software for measuring devices of company KMB Systems to mobile platform Windows Mobile. Research part describes environment of mobile devices and development environment Compact Framework. Then it informs about communication protocols supported by mentioned measuring devices, describes support libraries for easier work in this environment and states the options of saving measured data. In practical part I was solving application layout, the way of drawing phasor diagram, selection and download of measured record. For use of support libraries was necessary to adapt them to run in Compact Framework. At the end of the thesis is comparison with existing desktop application.

Keywords: .NET Compact Framework, mobile devices, protocol KmbLong, KMB-Lib library, CEA

Obsah

Seznam ilustrací

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

V dnešní době se stále více rozšiřují embedded zařízení, jako jsou kapesní počítače a mobilní telefony. Tato zařízení jsou již často výkonnější než počítače, které se běžně používaly před o něco málo více než deseti lety, a tak není divu, že aplikace vyvíjené pro tato zařízení se velmi podobají desktopovým aplikacím na jaké jsme dnes zvyklí. Proto je si vývoj aplikací až na pár výjimek dosti podobný. Pomineme-li menší spektrum funkcí, jsou zde další limitující faktory, a to omezené prostředky, zejména výdrž baterie.

Téma bakalářské práce jsem volil na základě toho, že částečně souvisí s bakalářským projektem, na kterém jsme s kolegou Kačmárem pracovali ve druhém ročníku, ale také proto, že vývoj aplikací pro mobilní zařízení se mi zdál zajímavý a může být užitečné se s touto problematikou seznámit. Projekt se týkal zejména prostudování komunikačních protokolů, které se používají u měřících přístrojů, jejich implementace a porovnání.

Úkolem mé bakalářské práce bylo vyvinout aplikaci pro mobilní zařízení s platformou Windows Mobile, která má sloužit ke zjišťování aktuálních dat a ukládání záznamů z přístroje od firmy KMB Systems. Následně bude možné tyto záznamy načíst a zobrazit v počítači aplikací ENVIS. Pro tento účel jsem se musel seznámit s prostředím Compact Framework a jeho možnostmi. Dále také s komunikačními protokoly měřících přístrojů a s podpůrnými knihovnami firmy KMB, které jsou určené pro vývoj aplikací pracujících s měřícími přístroji a upravit je tak, aby bylo možné je využít na mobilním zařízení.

Firma KMB Systems se zabývá výrobou panelových měřících přístrojů a programového vybavení pro práci s nimi. Přístroje slouží k měření napětí a dalších veličin elektrického proudu v elektrické síti. Díky ethernetovému konektoru je možné navázat komunikaci s aplikacemi, jejichž vývojem se firma také zabývá.

2 Použité nástroje a zařízení

2.1 SMP

Přístroj SMP je multifunkční registrační měřící přístroj pro soustavné vyhodnocování a sledování elektrických veličin. Umožňuje měření a záznam sdružených a fázových napětí, proudů, jalových výkonů, napěťových či proudových harmonických složek, činných i jalových energií, frekvence a dalších elektrických veličin v distribučních sítích.

Ilustrace 1: Měřící přístroj SMP

Přístroje jsou vybaveny vstupy pro připojení tří napěťových signálů a středního vodiče a tří izolovaných proudových vstupů. Modely „44“ mají navíc čtvrtý vstup pro měření proudu středního vodiče. Pro měření elektrické energie slouží zabudovaný třítarifní čtyřkvadrantní elektroměr, umožňující i záznam maximálních průměrných činných výkonů. Zaznamenávají se zvlášť údaje za právě probíhající měsíc, předchozí měsíc a celkový stav. Pro elektroměrové údaje lze využít samostatný archiv v paměti přístroje s možností programovatelné doby automatického odečtu.

Pomocí vestavěné klávesnice a displeje lze nastavit základní parametry přístroje. Přístroj tak lze používat jako multifunkční panelové měřidlo i bez použití počítače. Komunikace počítače s přístrojem je možná přes USB nebo Ethernet, jelikož obsahuje výstup RJ-45. Podrobnější popis přístroje je možné nalézt v [1].

2.1.1 Ukládání archivů v měřícím přístroji

Přístroj obsahuje paměť pro ukládání archivů. Archivy obsahují zaznamenané hodnoty z měření a časovou informaci, kdy měření probíhalo. Vedle hlavního archivu pro záznam průběhů vybraných veličin udržují přístroje další samostatné archivy pro ukládání tzv. napěťových událostí (rychlá překročení, poklesy napětí a přerušení napětí), týdenních záznamů, kvality napětí a událostí přístroje. Archivy se indexují v pořadí od 0 do maximální velikosti prostoru. Při zaplnění prostoru pro ukládání archivů se začnou zapisovat znovu od začátku, přičemž jsou jako první přepisovány nejstarší archivy. Tento způsob zápisu se označuje jako zápis cyklický. Při cyklení je aktuální archiv udáván zápornou hodnotou rozdílu mezi daným archivem a posledním v paměti.

Čas		U											
		prm				min				max			
čas záznamu	Trvání	U1[kV]	U2[kV]	U3[kV]	UN[V]	U1[kV]	U2[kV]	U3[kV]	UN[V]	U1[kV]	U2[kV]	U3[kV]	UN[V]
8. května 2011 4:33:00	00:01:00	21,75	21,75	21,67	185,2	21,68	21,68	21,60	184,5	21,77	21,77	21,69	185,3
8. května 2011 4:54:00	00:01:00	21,71	21,70	21,62	184,8	21,68	21,67	21,60	184,6	21,73	21,73	21,65	185,0
8. května 2011 4:55:00	00:01:00	21,73	21,73	21,65	184,9	21,70	21,69	21,61	184,7	21,74	21,74	21,66	185,1
8. května 2011 4:44:00	00:01:00	21,75	21,75	21,67	185,2	21,67	21,67	21,59	184,5	21,77	21,77	21,69	185,4
8. května 2011 4:52:00	00:01:00	21,75	21,74	21,67	185,1	21,73	21,73	21,65	184,9	21,77	21,77	21,69	185,3
8. května 2011 4:53:00	00:01:00	21,72	21,72	21,64	184,9	21,65	21,65	21,57	184,3	21,77	21,76	21,68	185,3
8. května 2011 4:21:00	00:01:00	21,76	21,76	21,68	185,3	21,70	21,70	21,62	184,7	21,80	21,79	21,71	185,5
8. května 2011 4:23:00	00:01:00	21,76	21,76	21,68	185,2	21,70	21,70	21,62	184,7	21,79	21,79	21,71	185,5
8. května 2011 4:24:00	00:01:00	21,74	21,73	21,65	185,0	21,69	21,68	21,61	184,6	21,76	21,76	21,68	185,2
8. května 2011 4:27:00	00:01:00	21,73	21,73	21,65	185,0	21,69	21,69	21,61	184,7	21,76	21,76	21,67	185,2
8. května 2011 4:29:00	00:01:00	21,72	21,72	21,64	184,9	21,66	21,66	21,58	184,4	21,77	21,76	21,69	185,3

Ilustrace 2: Stažené záznamy načtené v aplikaci ENVIS

2.2 PDA a mobilní zařízení

Mobilní zařízení v dnešní době již dávno neslouží jen k telefonování a posílání SMS zpráv. Často totiž obsahují nejrůznější moderní doplňky jako dotykový displej, podporují různé typu souborů, umožňují přehrávání audia a videa a disponují různými druhy připojení jako bluetooth či Wi-fi. Mají poměrně slušný výpočetní výkon a velikosti pamětí. Na trhu se vyskytují hlavně produkty s operačními systémy Windows, Android a Symbian.

Firmou KMB mi bylo zapůjčeno k vývoji zařízení Acer n310. Tento model bohužel patří ke starší generaci a používá starší verzi operačního systému. Je také vybaven nepříliš velkou vnitřní pamětí a především postrádá podporu Wi-Fi připojení. Pro připojení k ethernetu jsem tedy byl nucen využívat možnost připojení přes počítač pomocí USB kabelu. Aby bylo možné připojit zařízení s Windows Mobile k systému Windows je nutné mít nainstalován program Windows Mobile Device Center (u Windows XP ActiveSync), které slouží ke správě souborů, synchronizaci údajů a zároveň sdílí síťové připojení pro mobilní zařízení.

Technické parametry	
Typ procesoru	Samsung SC32440
Rozlišení	480 x 640
Operční systém	Microsoft Windows Mobile 5
Rozhraní pro připojení k PC	Bluetooth, USB
Frekvence procesoru [Mhz]	300
Velikost paměti RAM [MB]	64
Hmotnost [g]	135

Tabulka 1: Parametry modelu Acer n310

Ilustrace 3: Acer n310

Pro budoucí práci by bylo vhodnější mít k dispozici některý z novějších modelů s možností připojení k Wi-Fi síti a větší vnitřní paměť. Větší vnitřní paměť by umožnila snazší ladění aplikace, jelikož se při něm nahrává program a přidružené knihovny právě do ní a byly chvíle, kdy se aplikace s knihovnami do paměti nevešla. Důležitějším požadavkem je, ale podpora Wi-Fi připojení. Připojení přes počítač je sice užitečné, ale je dostatečné akorát tak pro vývoj prototypu aplikace. Aplikace je primárně určená pro použití přes Wi-Fi připojení, a tak je důležité vyzkoušet její funkčnost v tomto směru.

2.2.1 Ladění mobilních aplikací pomocí Visual Studia

Aktuální verze je Visual Studio 2010, ale bohužel už nepodporuje vývoj aplikací pro Windows Mobile, ale pouze pro nový Windows Phone. Proto jsem se využil dřívější verzi Visual Studio 2008, do kterého bylo nutné doinstalovat balíček pro vývoj mobilních aplikací Windows Mobile SDK.

Aplikaci jsem nejprve testoval v emulátoru, který je obsažen ve Windows Mobile SDK, později mi bylo zapůjčeno zmíněné PDA. Při ladění aplikace ve Visual Studiu je na výběr z několika emulátorů a také je možné ladit přímo na fyzickém zařízení. Této vlastnosti jsem si všiml bohužel až po nějaké době, a tak jsem zpočátku testoval hlavně v emulátoru, kde načítaní trvalo poměrně dlouho, a vyladěnější verze nahrával do PDA jako spustitelné soubory s příponou exe.

V popisu PDA jsem uvedl, že je možné sdílet síťové připojení počítače pomocí programu Windows Mobile Device Center (dále jen WMDC). Lze dosáhnout stejného cíle i u emulátoru? Ano lze, a to pomocí stejného programu a nastavení emulátoru. Ve WMDC je nutné vybrat připojení přes DMA. Ve Visual Studiu vybereme pod záložkou Tools, Device Manager a v něm nalezneme používaný typ emulátoru a po pravém kliknutí vybereme možnost Cradle. Pozor, je nutné, aby byl emulátor již spuštěný. Bohužel WMDC ne vždy nalezne emulátor, a tak jsem si dopomáhal připojením fyzického zařízení a následně jeho odpojením, čímž jsem nejspíše přiměl WMDC prohledat dostupná zařízení a nalézt emulátor. Další užitečnou pomůckou je v konfiguraci emulátoru možnost nastavit adresář počítače jako paměťovou kartu nebo při jeho ukončování uložit stav zařízení a ten se při příštím spuštění opět načte.

2.3 Compact Framework

Compact Framework je verzí .NET Framework určenou pro vývoj aplikací pro platformu Windows Mobile. Oba frameworky mají hodně společného, zejména podporované datové typy, názvy namespace, tříd a jejich vlastností, metod a událostí. Tyto společné prvky mají ulehčit programátorům přechod z programování pro desktopy a servery na programování mobilních zařízení. Podobná architektura umožňuje sdílet kód mezi projektem pro desktop a pro zařízení. Popravdě je poměrně snadné napsat kód v Compact Framework, který může běžet na desktopu nebo serveru. Opačný postup je už větší výzva, jelikož Compact Framework podporuje pouze podmnožinu .NET Framework. Překladu kódu pro Compact Framework se věnuji v kapitole 4. Pro vývoj aplikací je doporučeno používat nástroje Microsoft Visual Studio pomocí programovacích jazyků C# a Visual Basic. První verze byla vydána v roce 2002 a aktuální verze je 3.5.

Platform Invoke zkráceně P/Invoke je způsob, jak volat funkce, které jsou obsaženy v dynamických knihovnách. Ve většině případů poskytuje Compact Framework vše potřebné pro vytvoření programů. Občas však nastane situace, kdy programátor potřebuje využít funkci nebo datový typ, který není součástí Compact Framework, ale je obsažen v knihovnách Win32. Právě pro tyto případy slouží P/Invoke. Chceme-li toho využít v praxi, nejprve si musíme vytvořit tzv. P/Invoke obal. Obal vypadá skoro jako metoda, s tím rozdílem, že postrádá tělo funkce. Skládá se tedy z názvu metody, návratového typu a vstupních parametrů. Navíc P/Invoke obaly ještě vyžadují atribut DllImport pro identifikaci požadované knihovny a jména konkrétní funkce. Jak takový obal může vypadat následuje níže.

```
[DllImport("coredll.dll")]
public static extern int
MessageBox(IntPtr hWnd, String lpText,
String lpCaption,UInt32 uType);
```

Kód 1: Ukázka P/Invoke

Pokud to lze, je lepší využít prostředí Compact Framework. Existuje hned několik důvodů. Zaprvé je to automatické čištění paměti. Jestliže vytváříme systémový objekt pomocí funkcí Win32, musíme se postarat i o jeho odstranění z paměti. Dalším důvodem je přenositelnost. Dnešní aplikace napsaná, jak v .NET framework tak

v Compact Framework je lépe přenositelná než program používající Win32. Dále je důvěryhodnější, bezpečnější a jednodušší na programování.

V Compact Framework nelze použít relativní adresování, jelikož už není podporováno v samostatném systému Windows Mobile.

2.3.1 Síťová komunikace v Compact Framework

Pokud je naším cílem vytvořit aplikaci, která má komunikovat po sítí, je většina tříd, se kterými bychom se měli seznámit, umístěna ve jmenném prostoru System.Net. Obsahuje třídy pro práci s IP adresou, DNS, streamem, konkrétními protokoly (např. HTTP) a mnoho dalších.

Při komunikaci přes síť pomocí protokolu TCP/IP je důležitá IP adresa, která se ve verzi 4 skládá z adresy sítě a počítače. Veškeré metody a vlastnosti, které využijeme, nalezneme ve třídě System.Net.IPAddress. Nejužitečnější metodou je metoda Parse(). Díky ní dokážeme jednoduše vytvořit IP adresu pomocí řetězce se standardním oddělením tečkami, jak ukazuje následující příklad.

```
System.Net.IPAddress IP = System.Net.IPAddress.Parse("127.0.0.1");
```

Kód 2: Vytvoření instance třídy IPAddress

Přestože třída IPAddress je sama osobě užitečná pro správu adresy, většina síťových funkcí v Compact Framework používá k určení jiného zařízení v síti třídu IPEndPoint. Třída IPEndPoint poskytuje navíc informaci o portu, který bude použit k připojení ke službě běžící na vzdáleném zařízení. Pro vytvoření instance IPEndPoint slouží konstruktor se dvěma parametry. První parametr je typu IPAddress a druhý je typu int a značí číslo portu.

Další třídy, které stojí za zmínku jsou třídy TcpClient a TcpListener v namespace System.Net.Sockets. TcpClient se používá na straně klienta a slouží k připojení na vzdálený server a dokáže poskytnout stream pro odesílání a přijímání dat. Zatímco TcpListener je používán na straně serveru a zajišťuje vyřizování příchozích požadavků.

```
TcpClient client = new TcpClient();
IPEndPoint serverEndPoint =
new IPEndPoint(IPAddress.Parse("192.168.1.2"), 2101);
client.Connect(serverEndPoint);
NetworkStream clientStream = client.GetStream();
byte[] msg = { 1, 0, 0, 0x01, 0, 0 };
clientStream.Write(msg, 0, msg.Length);
clientStream.Flush();
```

Kód 3: Ukázka odeslání požadavku na server

3 Komunikace s přístrojem

Existují tři způsoby, jak komunikovat s měřícími přístroji. Každému způsobu odpovídá právě jeden port. Jsou to porty 80 pro webový server, 2101 pro KMB Long a 502 pro MODBUS. Hlavním úkolem komunikace je získávání informací o přístroji, naměřených hodnotách, popřípadě nastavování různých konfiguračních parametrů. Tyto funkce jsou dostupné přes KmbLong a MODBUS, zatímco webový server má čistě jen informativní charakter. Jeho hlavní výhodou je zjištění informací o naměřených hodnotách kdekoliv, kde je připojení k internetu a webový prohlížeč. Odpadá tím nutnost mít speciální aplikaci vyvinutou pro práci s přístrojem. WWW stránka obsahuje aktuální data, statistiky (údaje za tento měsíc, minulý měsíc a celkově), nastavení a další informace. Knihovna Kmb-Lib obsahuje také podporu protokolu KmbShort, což je starší verze protokolu KmbLong, ale ten je zde jenom z důvodu zpětné kompatibility a podporuje jenom požadavek k identifikaci přístroje. Pro následnou komunikaci s přístrojem jsem zvolil protokol KmbLong z důvodu dobré přehlednosti a srozumitelnosti kódu.

3.1 KmbLong

KmbLong je komunikační protokol používaný firmou KMB Systems, spadá do rodiny TCP/IP protokolů s architekturou klient-server a běží na portu 2101. To znamená, že komunikace probíhá způsobem požadavku a odpovědi. Klient otevře kanál komunikace na server, pošle požadavek o určená data a potom naslouchá na kanálu do doby než mu server zašle odpověď. Server nepřetržitě naslouchá a odesílá odpovědi na přijaté žádosti. Pro každé naslouchání je nutné mít zvláštní vlákno, aby nebyl program v tomto stavu zaseknutý a mohl vykonávat i jiné činnosti.

Struktura zprávy v protokolu KMB Long:

1.adresa zařízení (1 byte), hodnoty 0 a 255 jsou rezervovány

2.délka těla zprávy (2 byty)

3.typ zprávy(1 byte)

4.tělo zprávy se liší podle typu

5.16-bit CRC

Příkladem typu zprávy může být požadavek na identifikaci (hodnota 0x01) nebo požadavek na aktuální data 0x3A, kde se ještě v těle zprávy zasílá maska určující blíže data, o která žádáme. Ve výčtu požadavků lze také nalézt dotazy na všechny možné konfigurace, nebo požadavek ke stahování archivů. Typ archivu se určuje číselnou hodnotou, přičemž hlavní archiv má hodnotu 0. V dokumentu [2] je dostupný kompletní seznam všech požadavků včetně návodu k jejich používání a popis navrácených dat.

3.1.1 Maska

Maska 32bitová slouží k výběru parametrů u aktuálních dat. První čtyři bity určují fázi a je nutné mít zadanou alespoň jednu ze tří fází, jinak bychom žádná data nezískali. Jaké informace budou poslány, říkají zbylé bity. Zaslaná odpověď pak obsahuje nejen požadovaná data, ale mimo jiné i poslanou masku. Pro uchování a práci s maskou existuje třída SmpMsgCfgInt. Maska je podrobněji popsána v [2].

3.2 Modbus

Druhý protokol, který lze použít pro komunikaci s měřícími přístroji je obecně otevřený protokol pro komunikaci s různými zařízeními, který umožňuje přenos dat po různých sítích a sběrnicích. Funguje také na principu klient - server. Modbus je mimo jiné, také možno přenášet přes TCP/IP na portu 502. MODBUS nabízí možnost čtení i zápisu dat přístroje. Zařízení pošle zpět odpověď do 200ms po přijmutí každého příkazu a dokáže zpracovat až tři požadavky současně.

Podporované funkce

3 (0x03) čtení uchovávaných záznamů (read holding registers)

4 (0x04) čtení vstupních záznamů (read input registers)

16 (0x10) zápis více záznamů (write multiple registers)

Mód "Broadcast" není podporován.

4 Podpůrné knihovny

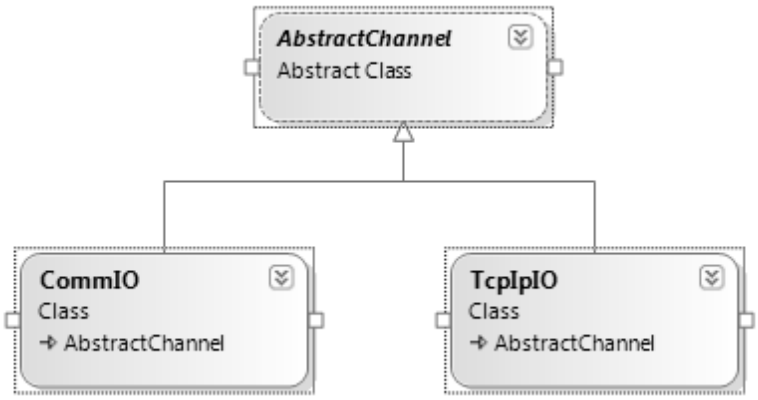
4.1 Kmb-Lib

Základní knihovnou pro vývoj aplikací pracující se zmiňovanými měřícími přístroji je knihovna Kmb-Lib. Obsahuje různé pomocné třídy, struktury pro ukládání naměřených dat a konfigurace z přístroje, třídy ke komunikaci a zpracování odpovědi. Ostatní rozšiřující knihovny ji pak mají v referencích a využívají její třídy a metody.

Knihovna Kmb-Lib je rozdělena do několika jmenných prostorů. Jsou to KMB.Lib, KMB.Communication, KMB.Structures, KMB.Types, KMB.SMO a KMB.Resources, přičemž já jsem nejvíce využíval první tři jmenované. KMB.Lib obsahuje různé informační třídy, výčtové typy, nástroje pro správu a obecné typy, které nelze jinam zařadit.

4.1.1 KMB.Communication

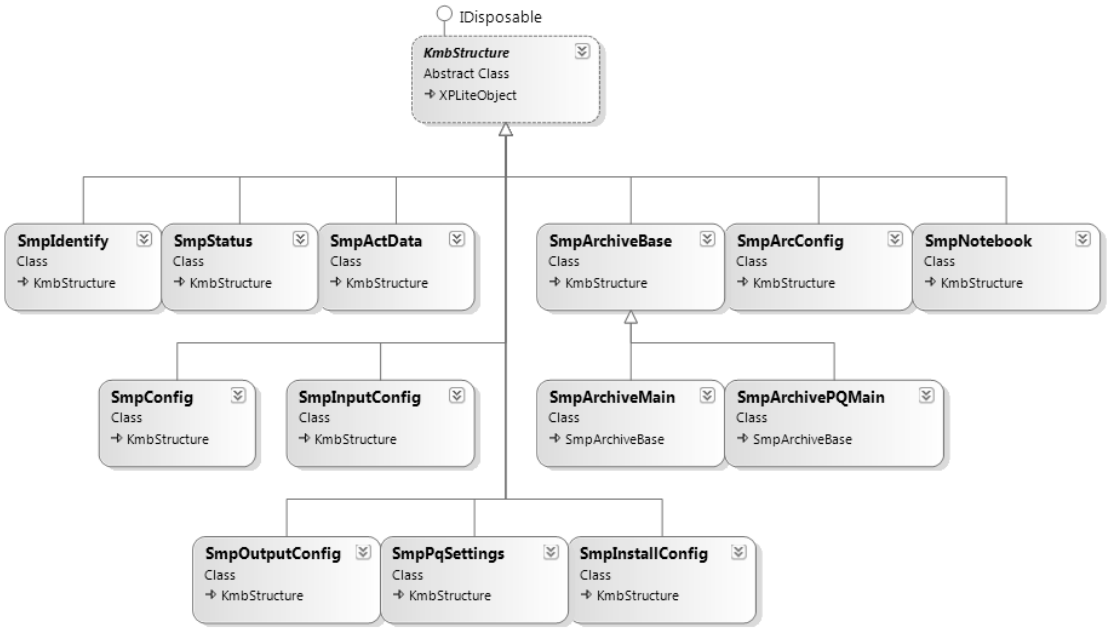
Namespace KMB.Communication obaluje třídy sloužící ke komunikaci. Při použití protokolu KmbLong se vytváří instance abstraktní třídy TcpIpIo s parametry IP adresy a čísla portu. Získáme tak komunikační kanál přes, který odesíláme a zároveň přijímáme zprávy.



Ilustrace 4: Class Diagram namespace KMB.Communication

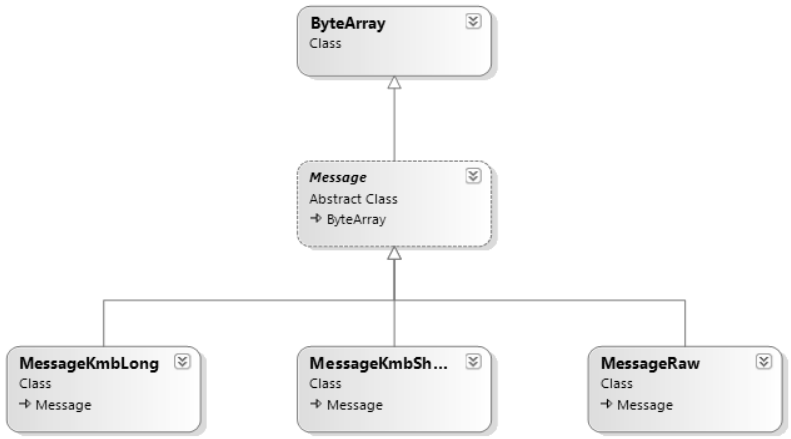
4.1.2 KMB.Structures

V tomto jmenném prostoru jsou uloženy struktury k uchovávání dat. Většina tříd v něm dědí od abstraktní třídy KmbStructure. Mezi tyto třídy se řadí různé konfigurace, třídy k práci s aktuálními daty či archivy. Na níže uvedeném obrázku jsou zastoupeny pouze třídy, se kterými jsem se setkal a použil je při psaní aplikace.



Ilustrace 5: Class Diagram namespace KMB.Structures

Mimo jiné je zde třída ByteArray starající se o práci s poli bytů. Tato třída je nejvíce využívána v případě zpracovávání odpovědí od přístroje a následném vytváření objektů k jejich uchování.



Ilustrace 6: Class Diagram třídy ByteArray

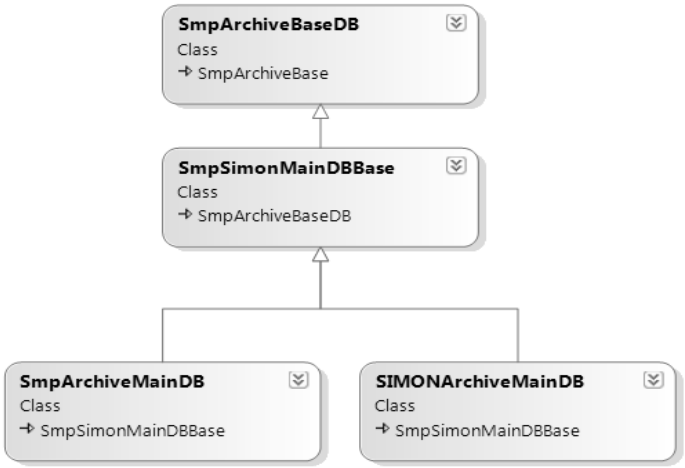
```
//identify
List<MessageKmbLong>result;
result=channel.communicate(null,MessageKmbLong.getIdentify(1),null);
SmpIdentify ident = new SmpIdentify(result[0].read_ByteArray(4,
(ushort)(result[0].messageLength + 4)));
```

Kód 4: Ukázka zaslání požadavku a zpracování odpovědi identifikace

V kódu (4) je uveden příklad zaslání požadavku identifikace přístroje přes kanál (instance třídy TcpIpIo) a jeho následné zpracování. Pro zaslání požadavku se používá metoda communicate, která obsahuje jako parametr typ požadavku. Odpověď se ukládá do kolekce MessageKmbLong. V tomto případě je navrácen pouze jeden řádek odpovědi, a tak jsou stažená data uložena pod indexem 0. Jako vstupní parametr konstruktoru třídy SmpIdentify dáváme pouze tělo zprávy, a proto první čtyři byty přeskočíme.

4.2 Envis.Model

Envis.Model je rozšiřující knihovna ke KMB-lib, která slouží k uchování modelů pro databáze. Tuto knihovnu jsem použil při ukládání dat do formátu CEA, jelikož v parametrech metod pro zápis se objevovaly objekty z této knihovny. Objekty z této knihovny poznáme podle toho, že většina končí písmeny DB. Přikladem je identifikace, která má v Kmb-Lib název SmpIdentify a v Envis.Model je to SmpIdentifyDB.



Ilustrace 7: Ukázka části namespace Envis.Model

5 Možnosti uložení archivů

Po stažení záznamů z přístroje je potřeba je nějakým způsobem ukládat a uchovat. K tomuto účelu je možno využít několika metod, které jsou spojeny s konkrétními typy souborů. Mezi nejsnazší a zároveň nejméně flexibilní patří ukládání dat do excelovských tabulek. Další možností je využít soubory typu CSV, což je textový soubor, v kterém jsou jednotlivá data oddělena čárkou. Zajímavý je formát PQDIF, který je standardem IEEE 1159.3. Podle internetové stránky [5] je formát PQDIF vhodný pro výměnu měřených a simulovaných dat zaměřených na kvalitu elektrické energie. Já jsem však pro archivaci dat zvolil formát CEA. Hlavním důvodem tohoto rozhodnutí byla podpora formátu programem ENVIS, čímž je zaručena možnost načtení archivovaných dat v na osobním počítači a zároveň splněn smysl práce.

5.1 Formát souboru CEA

Firma KMB používá pro archivaci naměřených dat z přístroje vlastní formát souboru s příponou cea. Tento soubor je komprimovaný ZIP archiv. Každý binární soubor v tomto ZIP archivu obsahuje data pro jeden typ archivu, který může obsahovat větší počet jednotlivých měření. Následuje popis struktury jednoho souboru v CEA, kde je nejprve uvedeno, co je na daném místě uloženo a jakou má velikost.

Popis struktury verze 3 ¶

hlavička - řetězec "Archive"
verze souboru - 1 byte
číslo archivu - 1 byte
počet záznamů - int
DateRange - 2 x U64
notebook - délka(ushort)+byteArray
identify - délka(ushort)+byteArray

//dále jsou uloženy jednotlivé configy
počet configu - int
postupně všechny configy - délka(ushort)+ByteArray

//následuje uložení záznamů
číslo příslušného configu - int
počet záznamů se stejným configem a délkou ByteArray
délka jednoho ByteArray
postupně záznamy se stejným configem a délkou
záznam – byteArray

6 Požadavky a Návrh

6.1 Požadavky na aplikaci

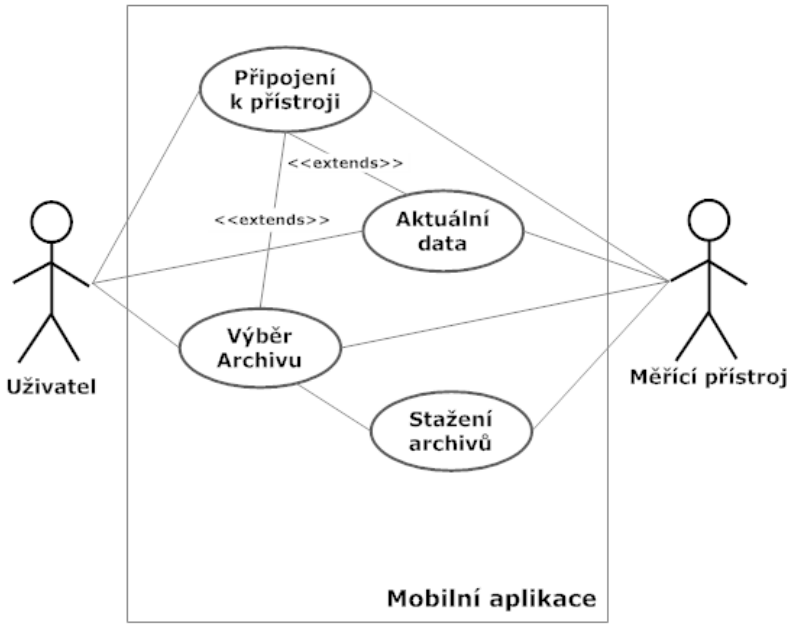
Základní myšlenkou bylo vytvoření prosté aplikace pro mobilní telefon, která bude v základu obsahovat dvě funkce. První funkcí je možnost připojit se k měřícímu přístroji, získat a zobrazit aktuální data. Druhá má za úkol z připojeného přístroje stáhnout a uložit naměřené archivy podle výběru. K většímu rozbor dat je lepší využít osobní počítač, pro který už existují pro tento účel pokročilejší aplikace. Výhodou mobilní aplikace je, že mobilní telefon máme většinou u sebe, a tak můžeme rychle zjistit aktuální data anebo si můžeme stáhnout archivy, které si lze po přenosu do osobního počítače načíst programem ENVIS. Jediným limitujícím faktorem je zajištění propojení obou zařízení v rámci Ethernetu či Wi-Fi. K ukládání archivů slouží formát CEA, který popisuji v předchozí kapitole. Aplikace má mít grafické uživatelské prostředí a srozumitelné ovládání.

Pro vývoj mobilních aplikací existují dvě hlavní větve, aplikace pro OS Android vyvíjené pomocí prostředí Java a pro OS Windows Mobile pomocí knihoven Compact Framework. Již v zadání práce bylo určeno použití Compact Framework, a tak jsem žádné srovnání nebo rozhodování, který způsob využít nedělal. Cílovou platformou je tedy Windows Mobile s knihovnami Compact Framework. Toto kritérium je celkem oprávněné vzhledem k tomu, že na zapůjčeném PDA běží Windows Mobile a také to, že firma KMB používá pro své aplikace a knihovny jazyk C#. Osobně po setkání s OS Android v rámci předmětu „Programování mobilních zařízení“ bych volil stejně, jelikož se mi zdá přechod z .NET Framework na Compact Framework snazší než obdobný postup v prostředí Java. Pro vývoj aplikace lze využít zmíněné podpůrné knihovny.

6.2 Návrh řešení

Aplikace bude schopna komunikovat pomocí Wi-Fi připojení, případně sdíleného přípojení z počítače, pomocí protokolu KmbLong. Uživatel zadá IP adresu přístroje, port bude přidělen automaticky. Jakmile je navázána komunikace, může si uživatel zobrazit aktuální data nebo stahovat archivy dle výběru. Vybírat se může dle několika kritérií. Ta se dají rozdělit na dvě hlavní skupiny. Jedna skupina vybírá archivy na základě zadaného časového období a druhá podle počtu posledních měření.

Výběr podle data bude využívat metodu binárního či interpolačního vyhledávání pro nalezení příslušných archivů. Interpolačního vyhledávání je variantou binárního, v němž se porovnávací prvek neurčuje půlením intervalu, ale odhadem. Stažené archivy se dále exportují do formátu souboru CEA na základě uživatelem zadané cesty. Tento postup může uživatel vícekrát opakovat.



Ilustrace 8: Princip aplikace pomocí use case diagramu

Pro práci s jednotlivými datovými strukturami se využijí podpůrné knihovny. Ty je nejprve nutné uzpůsobit chodu v Compact Framework. Této problematice se věnuji v následující kapitole.

7 Překlad kódu pro Compact Framework

Transport kódu z platformy .NET na platformu .NET Compact Framework není obtížný, jelikož se zde používá stejný programovací jazyk a některé třídy tu jsou podobné i shodné. Můžeme, ale narazit na různé překážky. Jak jsem již zmínil, mobilní zařízení mají omezené prostředky spojené s nižším výpočetním výkonem, velikostí paměti, případně výdrží baterie. Proto je Compact Framework optimalizovaný pro práci s mobilními zařízeními a také má menší počet dostupných tříd a metod. Prvním krokem konverze je vytvoření projektu pro Compact Framework. Poté je možno vložit původní kód a pokusit se jej přeložit a zjistit, kde jsou konkrétní problémy. Může zde nastat situace, kdy Compact Framework nezná pouze způsob zadání vstupních parametrů, konkretní metodu, nebo dokonce celou třídu či namespace. V takových případech máme několik možností. Pokud jde jen o nesrovnalosti ve vstupních parametrech, postačí pouze jejich úprava do požadovaného tvaru. Jestliže ale nějaká část chybí, máme dvě možnosti. Můžeme zkusit vyhledat na internetu, zda již někdo jiný požadovanou implementaci nevytvořil a nezveřejnil ji k volnému užití nebo musíme napsat potřebný kód sami. V mém případě jsem využil oba způsoby. Ve třídách, při jejichž překladu nastal problém s chybějící částí kódu a u kterých jsem funkčnost nevyžadoval jsem si často ulehčil práci pouze vytvořením prázdných tříd a metod za účelem dosáhnouti úspěšného překladu bez chyb. Zajisté toto řešení není úplně správné, ale mně posloužilo a až někdo bude dané třídy potřebovat, má možnost si je doplnit.

Za zmínku také stojí podmínka pro preprocesor #if PocketPC, která použije vnořený kód na základě použité platformy. Toho jsem využíval v případech, kdy některé metody v Compact Framework používaly jiné vstupní parametry. Výhodou tohoto postupu je, že původní kód zůstane zachován při použití na desktopu.

7.1 BackgroundWorker

Nyní následuje konkrétní příklad postupu překladu. Třída BackgroundWorker v Compact Framework zcela chybí. Dle výše uvedeného postupu jsem nejprve hledal na internetu, zda již někdo její implementaci neposkytl k volnému užití. Daniel Moth na svém blogu [3] poskytuje dll s implementací třídy BackgroundWorker pro Compact Framework. Z pohledu mé práce je bohužel tato binární knihovna nedostatečná, protože neobsahuje všechny metody, které jsou vyžadovány k úspěšnému překladu. Potřeboval

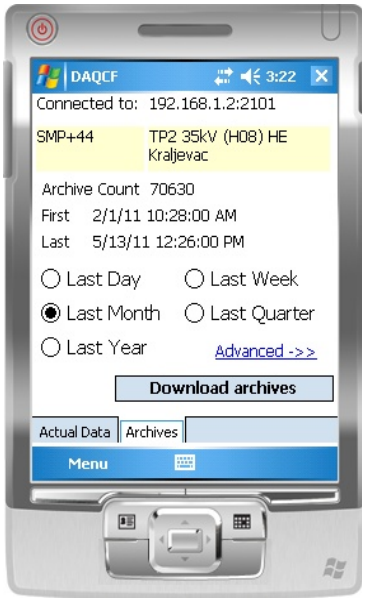
jsem tedy získat přímo zdrojové kódy, do kterých by bylo již možné doplnit chybějící třídu. Zmíněná internetová stránka poskytuje kromě knihovny, pouze kód ve Visual Basicu a verze pro C# má být obsažena ve Smart Device Framework v rámci projektu OpenNETCF. OpenNETCF je open source projekt, který poskytuje framework pro vývoj pro mobilní platformu a případné rady a konzultace ohledně vývoje spadají pod komerční oblast projektu. Stáhl jsem si volně dostupnou verzi a vyhledal jsem požadovaný BackgroundWorker. Zde jsem doplnil třídu isBusy podle implementace v klasickém .NET. V takovýchto případech se mi osvědčil webový portál [4], kde jsou uvedeny kompletní kódy tříd v .NET. Nyní již stačilo zjistit, jaké proměnné si navzájem odpovídají a přepsat je. Samozřejmě jsem také mohl v knihovně odstranit závislosti na BackgroundWorker, ale jak se později ukázalo, potřeboval jsem jej při stahování archivů z přístroje.

8 Vlastní aplikace

8.1 Popis GUI

Úvodní obrazovka je prázdná a obsahuje informaci o připojení, přičemž po startu aplikace je zobrazen text Not Connected. Jediné položky dostupné po startu jsou položky Connect a Quit v Menu. Možnost Quit slouží k ukončení aplikace a Connect pro připojení k přístroji. Po kliknutí na Connect se otevře nové dialogové okno, ve kterém se zadává IP adresa a port. V původním návrhu bylo, že port bude přidělován automaticky, ale vzhledem k tomu, že přístroj může být i někde v podsíti, díky přesměrování portů (port forwarding), rozhodl jsem se, že tuto položku udělám volitelnou.

Po připojení se změní informace o připojení a v Menu se změní položka Connect na Disconnect. Při kliknutí na Disconnect se dostaneme do stejného stavu, v jakém byla aplikace po startu. Dále se zobrazí název a popis přístroje a TabControl se dvěma záložkami. První záložka obsahuje aktuální data ve formě fázorového diagramu a tlačítko pro jejich obnovení.



Ilustrace 9: Screenshot aplikace při výběru archivů

Druhá záložka se věnuje archivům. Udává jejich počet, datum prvního a posledního vytvořeného archivu a formulář pro jejich stahování. Výběr archivů probíhá pomocí kontrolky RadioButton, přičemž pro běžné používání je nabízen výběr

podle určení posledního časového období. Například možnost stáhnout data za poslední den či měsíc. Po kliknutí na Advanced je uživateli nabídnuta možnost určit počet posledních archivů a možnost zadat interval dat. Pro určení počtu posledních archivů je použita kontrolka NumericUpDown s minimem 1 a maximem celkového počtu archivů. Data se volí pomocí dvou kontrolek způsobem od-do. Pro výběr data je použit DateTimePicker. Nakonec se zde nachází tlačítko ke stažení vybraných archivů. Při kliknutí na něj je zobrazeno dialogové okno pro výběr cesty k uložení souboru. Během stahování archivu se zobrazí ProgressBar, který informuje o průběhu akce.

8.2 Proces připojení

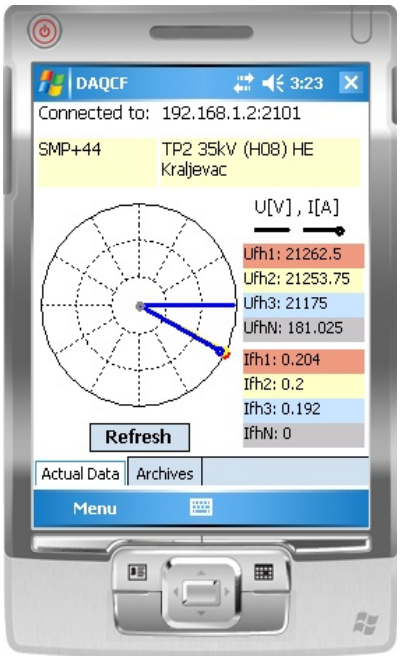
V počátcích tvorby své aplikace jsem ke komunikaci používal třídy z jmenného prostoru System.Net a vytvářel jsem konkrétní strukturu požadavků. Toto řešení bylo funkční, avšak potřebovalo by ještě doplnit o lepší režii. Z tohoto důvodu jsem raději využil třídy TcpIpIO, která je součástí Kmb-Lib a stará se o síťovou komunikaci. Použití této třídy zpřehledňuje kód a poskytuje požadovanou režii. Při připojení k zařízení si vytvořím instanci třídy TcpIpIo, kterou si předávám mezi funkcemi využívající komunikaci s přístrojem pomocí vstupního parametru. Při kliknutí na Disconnect se spojení s přístrojem ukončuje. Kromě použití TcpIpIo jsem z Kmb-Lib dále využil třídu MessageKmbLong, s jejíž pomocí se generují zprávy požadavků. Nemusel jsem tedy hledat v manuálu, jak má daný požadavek vypadat, stačilo zadat konkrétní metodu, která vrací výsledek jako pole bytů. Tímto postupem se také zvyšuje čitelnost kódu.

Při procesu připojení se také stahují informace o přístroji, status, konfigurace a aktuální data. Při zobrazovaní aktuálních dat je nutné nejdříve udělat přepočet podle nastavené konfigurace. K tomu jsou určeny multiplikátory definující poměr převodu. Jsou to MTN pro napětí a MTP pro proud. Multiplikátory jsou uloženy v Install Config a při určitém nastavení se musí také přepočítávat. Například napětí se vypočítá vztahem U*MTN/40, přičemž toto platí pro první tři fáze a pro čtvrtou fázi n se používá vztah jiný. Podrobnější informace o přepočtech jsou v dokumentu [2].

Aktuální data jsou zobrazena pomocí fázorového diagramu, který informuje nejen o hodnotách napětí a proudu, ale i o úhlech jednotlivých fází. Tato aktuální data lze pomocí tlačítka refresh obnovit.

8.2.1 Fázorový diagram

Fázorový diagram je možné aplikovat pouze na sinusové průběhy u střídavého proudu. Slovem fázor se označují otáčivé vektory, které mají určitou velikost a směr "zamrzlý" v určitém bodě v čase. Tyto vektory mají pevný bod počátku a rotují v protisměru hodinových ručiček. Ve fázorovém diagramu lze zobrazovat více jak dvě sinusoidy, ale musí se shodovat ve frekvenci. U fázorového diagramu se třemi fázemi se standardně používají barvy červená, žlutá a modrá, přičemž červená označuje fázi referenční a obecně se kreslí podél horizontální osy. Ostatní fáze se určují podle posunu od referenční fáze. Ilustrace(10) ukazuje stažená data z přístroje, který bohužel měří pouze jednu fázi.



Ilustrace 10: Screenshot aplikace s fázorovým diagramem

Vykreslování diagramu jsem řešil pomocí třídy Graphics s metodami drawLine() a drawElipse(). Pro vykreslení čáry je nutné znát oba koncové body. Zatímco první bod nalezneme ve středu, druhý bod je nutné spočítat pomocí úhlu a velikosti. Pro tento účel jsem napsal metodu, která vrací proměnnou typu Point a vstupní parametry má výchozí bod, úhel v intervalu 0 až 2π a vzdálenost. Pro ošetření vstupního úhlu jsem napsal metodu k jeho převedení do požadovaného intervalu. Pak už zbývá jenom tyto dva body spojit. Velikost se nastavuje v poměru k maximální hodnotě, která má hodnotu poloměru kruhu diagramu.

8.3 Výběr archivů

Jak již bylo řečeno, archivy se vybírají podle dvou kritérií. V případě počtu posledních archivů je řešení snadné. Startovní číslo archivu se volí na základě hodnoty aktuálního archivu méně 1 méně počet archivů a konečné číslo archivů je hodnota aktuálního archivu méně jedna. V případě cyklického zápisu archivu v přístroji má aktuální archiv zápornou hodnotu odečtu od maximálního čísla archivu a je potřeba jej převést do správného tvaru.

Vybíráme-li archivy podle data, nastává problém s určením, které archivy stahovat, jelikož při žádosti o archiv se zasílá jeho číslo, nikoliv čas měření. Z tohoto důvodu je nutné stahovat jednotlivé archivy a zjišťovat čas jeho měření. Archivy se zapisují postupně a chronologicky takže máme zaručenou posloupnost časů a můžeme využít binárního vyhledávání. Zvažoval jsem i metodu interpolačního vyhledávání, ale pro mou práci nebylo zcela vyhovující. Tato metoda je totiž založená na předpokladu, že jsou data v poli rovnoměrně rozprostřená, což by při neustálém měření mělo odpovídat, ale při častém vypínání či výpadkům přístroje by tomu tak nebylo. Důležitějším aspektem proč jsem tuto metodu nezvolil, byly větší nároky na komunikaci. Při počítání odhadu se zjišťují hodnoty krajů v aktuálním hledaném intervalu. Hodnota je v tomto kontextu čas měření a ten lze zjistit pouze stažením archivu, což způsobuje mírně vyšší nároky na přenos dat.

$$index = \frac{(pravy - levy)}{2}$$

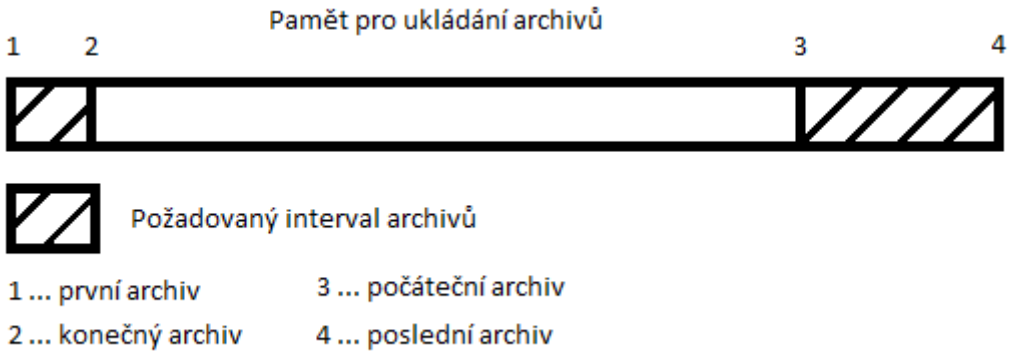
$$index = levy + \frac{pravy - levy}{a[pravy] - a[levy]} . (hodnota - a[levy])$$

Ilustrace 11: Určení indexu u binárního a interpolačního vyhledávání

Počáteční a konečné číslo archivu se zjišťuje zvlášť pomocí metody, která vrací číslo archivu, který je časově nejblíže požadovanému datu. Pokud je zadáno datum, které předchází prvnímu měřením nebo následuje po posledním, je vráceno číslo archivu prvního, ve druhém případě posledního. Další problém nastává v situaci, kdy je navrácený archiv časově před zadaným počáteční datem, v takovém případě použijeme archiv následující. Obdobně se řeší případ, kdy je navrácený čas po zadaném konečném datu, s tím rozdílem, že chceme předchozí archiv místo následujícího.

8.4 Stažení archivů

Pro stažení archivů jsem vytvořil metodu, do které vstupuje číslo počátečního archivu a číslo konečného archivu v požadovaném intervalu. Archivy se stahují v cyklu po jednom, jelikož je odpověď omezena na 7kB a archivy, které stahuji mají průměrně kolem 6kB, a tak lze v rámci jednoho cyklu stáhnout pouze jeden archiv. Původně jsem stažené archivy ukládal do datové struktury list, ale vzhledem k velikosti dat a urychlení programu jsem se rozhodl, že budu stažený archiv rovnou zapisovat do výstupního souboru. Menší problém nastává v okamžiku, když ukládání archivů v přístroji cykluje. Informace o cyklení archivů se získává již v procesu připojení. Při cyklickém zápisu archivu může nastat stav, kdy je číslo počátečního archivu větší než číslo konečného. To znamená, že se musí stáhnout data ve dvou intervalech. První je od počátečního čísla až do posledního čísla archivu a druhý od prvního archivu až po konečný archiv. Pro lepší představu je uveden obrázek ke zmíněnému stavu.



Ilustrace 12: Příklad stavu při cyklickém zápisu archivů

8.5 Ukládání do formátu CEA

Nejprve je nutné založit si dočasnou složku, do které se budou ukládat jednotlivé soubory podle typu archivů. Složku vytvářím ve stejném adresáři, který zadal uživatel k uložení CEA souboru. V dočasné složce vytvoříme nový soubor s názvem složeným z popisu archivu, aktuálního času a přípony arch. Stream k souboru předáme konstruktoru třídy ExportBin spolu s typem archivu a instancí SmpMeasNameDB, která obsahuje popis a identifikaci přístroje. Třída ExportBin není součástí dvou výše jmenovaných knihoven, ale je na nich závislá. Její zdrojový kód mi byl poskytnut pro usnadnění práce s CEA. Následuje zápis konfigurací přístroje pomocí metody WriteConfig třídy ExportBin. Všechny konfigurace je před zápisem nejprve nutné

stáhnout. Archivy se zapisují třídou WriteRecord ve stejném cyklu jako se stahují od nejnovějšího k nejstaršímu. Po dokončení zápisu archivů se musí soubor zakončit metodou Finish a převrátit pořadí měření metodou ReverseOrder. Tímto způsobem se mohou ukládat i ostatní typy archivů, pokud jsou vyžadovány. Nakonec, když máme všechny soubory uložené, zkomprimujeme dočasnou složku pomocí metody ZIP do jednoho souboru s příponou cea. Pro vlastní komprimaci je využita knihovna ICSharpCode.SharpZipLib. V mém případě jsem použil třídu FastZip.

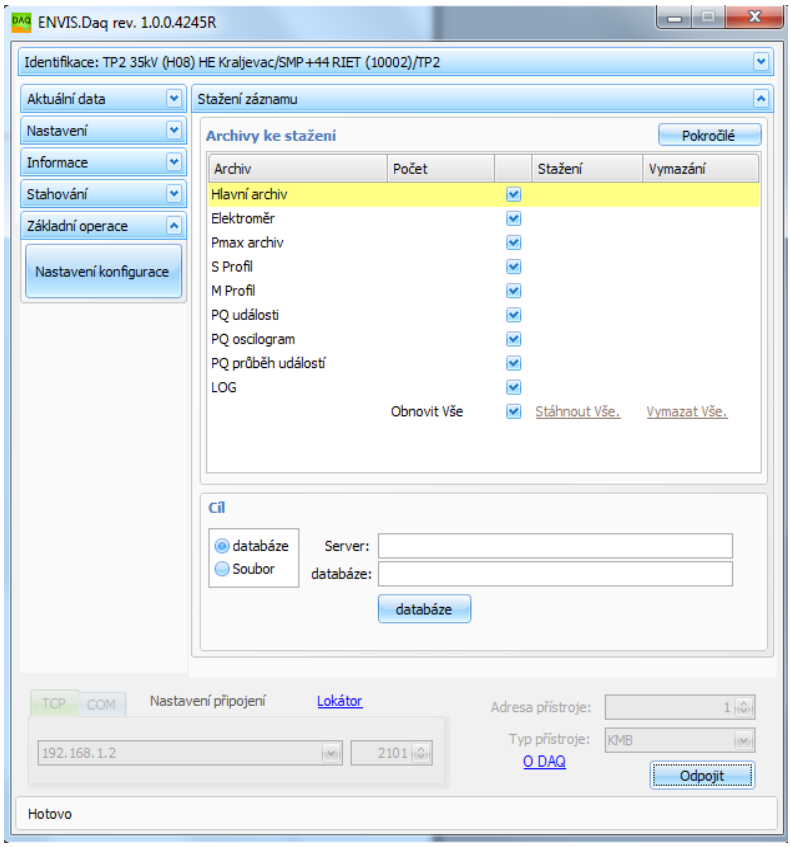
8.6 Vytvoření instalace aplikace

Pro snazší přenositelnost a instalaci mé aplikace jsem vytvořil instalační soubor s příponou CAB, který stačí zkopírovat na mobilní zařízení a spustit. Po spuštění tohoto souboru se zobrazí dialog, ve kterém je možné zjistit požadované volné místo a vybrat umístění instalovaného programu.

Vytvoření instalačního souboru podporuje přímo Visual Studio, musíme však vytvořit nový projekt. Tento nový projekt pro instalaci je také možné přidat k již existující solution. Jako typ projektu vybereme v Setup and Deployment typ instalačního souboru. Pro PocketPC vybereme možnost Smart Device CAB Project. Po přidání projektu lze změnit ve vlastnostech název produktu a firmy. V tomto projektu pak můžeme vytvářet adresářovou strukturu s definovaným obsahem. Do adresářů je možné přiřazovat přímo výstup jiných projektů nebo i externí soubory.

9 ENVIS.Daq

ENVIS.Daq je aplikace vyvinutá firmou KMB. Prostřednictvím ní je možné sledovat aktuální data, stahovat různé druhy archivů, měnit nastavení přístroje a různé další možnosti. V položce aktuálních dat zobrazuje hodnoty měřených veličin v tabulkách, fázorovém diagramu, oscilogramu a grafech. Archivy lze stahovat do souborů i do databáze a vybírají se podle druhu a zadaného úseku. Úsek je možné určit časově, počtem či porovnáním s databází a identifikací chybějících archivů. Aplikace ENVIS.Daq se mi stala předlohou pro vlastní aplikaci a mým úkolem bylo vytvořit pouze podmnožinu této aplikace pro mobilní zařízení.



Ilustrace 13: Screenshot aplikace ENVIS.Daq

9.1 Srovnání s programem ENVIS.Daq

V připojovacím procesu je má aplikace o trochu pomalejší. Důvodem toho může být skutečnost, že při připojení zároveň stahuji aktuální data a konfigurace, tudíž stahuji větší objem dat. ENVIS.Daq toto řeší až po připojení při zažádání konkrétních dat nebo při potřebě. Rychlost mohou také ovlivňovat různé platformy. Vlastní obnovení aktuálních dat už je srovnatelné.

O poznání horších výsledků je dosaženo při stahování archivů. Problém je v tom, že v aktuální verzi se stahují archivy po jednom, což způsobuje delší dobu trvání. Nicméně způsob je funkční a výsledek je kromě doby stahování téměř stejný. Výstupní soubory CEA jsou u obou aplikací srovnatelné obsahem i velikostí. Komprimování je také pomalejší, ale tuto vlastnost bych přičítal nižšímu výpočetnímu výkonu.

	ENVIS.Daq		DAQCF (Mobilní aplikace)	
Počet archivů	Čas [s]	Velikost [kB]	Čas [s]	Velikost [kB]
136	2	46	26	47
1440	14	465	215	475
964/1114	7	311	150	368
573	5	149	80	189
425	4	139	60	141

Tabulka 2: Porovnání aplikací ENVIS.Daq a DAQCF při stahování hlavního archivu z přístroje

10 Závěr

Ukázalo se, že přechod z klasického .Net framework do Compact Framework, není až tak složitý, a tak mi předchozí zkušenosti s .Net přišly k užitku. Síťová komunikace byla prakticky stejná, a tak jsem nenarazil na větší problémy s tím spojené. Značnou dobu vypracovávání práce jsem strávil seznamováním se s prostředím měřících přístrojů. Bylo nutné naučit se používat komunikační protokol, zjistit jaká data a v jakém formátu jsou uložena v přístroji, a porozumět třídám a metodám v podpůrných knihovnách. Úprava podpůrných knihoven pro použití v Compact Framework byl nutný první krok, bez kterého by se práce značně zkomplikovala. I přes větší časovou náročnost, byl tento krok ve srovnání s alternativou, ve které bych se musel obejít bez těchto knihoven, výhodný. Podpůrné knihovny jsou celkem komplexní a rozsáhlé, a tak není zcela snadné se v nich zorientovat. Ačkoliv je dokumentace poměrně obsáhlá, nepokrývá úplně vše a způsob funkce a použití metod jsem zjišťoval formou konzultací. Práci také komplikovala cílová platforma, jelikož byl občas problém s připojením emulátoru či vlastního PDA. Samotný emulátor fungoval dle očekávání, ale jeho připojení k Ethernetu bývalo občas problematické.

Prostřednictvím výsledné aplikace je možné se připojit k přístroji zadáním IP adresy a portu. Po připojení je možné sledovat aktuální měřená data z přístroje a stahovat zaznamenané archivy. K zobrazení aktuálních dat je použit fázorový diagram, který informuje o hodnotách napětí, proudu i úhlů jednotlivých fází. Archivy lze vybírat skrze několik kritérií a je zavedena podpora cyklujícího zápisu archivů. Stažené archivy se ukládají do souboru formátu CEA, což umožňuje opětovné načtení v osobním počítači aplikací ENVIS. V této aplikaci jsem ověřoval správnost stažených CEA souborů z hlediska výběru archivů a částečně i jejich obsahu. Na závěr jsem vytvořil instalační soubor CAB spustitelný na mobilním zařízení.

Při vytváření aplikace jsem narazil na další omezení mobilních zařízení, o kterém jsem dříve neuvažoval a přitom vyplývá z jejich vlastností. Ačkoliv se to nemusí tak jevit, velikost displeje je dalším omezením, proto je nutné s místem šetřit a správně navrhnout rozložení prvků tak, aby aplikace zůstala ještě přehledná. Tento prostor lze uměle rozšiřovat pomocí záložek či posuvníků, ale při vyšším množství použití těchto rozšíření by se aplikace mohla stát příliš chaotickou.

V porovnání s aplikací ENVIS.Daq, která obsahuje stejné funkce jako mobilní aplikace, je pomalejší. Hlavně část stahování archivů je pomalá a v budoucí práci by bylo vhodné ji zlepšit. Rychlost ostatních funkcí již není o tolik pomalejší a lze ji přisuzovat nižšímu výpočetnímu výkonu. Aplikace v dnešní podobě podporuje pouze stahování hlavních archivů, a tak se nabízí její rozšíření v tomto směru i o další typy archivů.

Seznam použité literatury

[1] www.kmb.cz [online]. 2011 [cit. 2011-05-03]. SMV, SMP, SMPQ Multifunkční panelové přístroje a analyzátory kvality energie. Dostupné z WWW: <http://www.kmb.cz/07/doc/SMV_SMP_SMPQ-Manual-v4-cze.pdf>.

[2] www.kmb.cz [online]. 2011 [cit. 2011-05-03]. SMV, SMP, SMPQ Multifunctional Panel Meters & Power Quality Analyzers Communication Protocol Manual. Dostupné z WWW: <http://www.kmb.cz/07/doc/SMV_SMP_SMPQ-Communication_Protocol-v2-czeeng.pdf>.

[3] www.danielmoth.com/Blog/ [online]. 2011 [cit. 2011-05-03].BackgroundWorker Sample. Dostupné z WWW: <http://www.danielmoth.com/Blog/backgroundworker-sample.aspx>.

[4] DeveloperFusion [online]. 2011 [cit. 2011-05-04]. Dostupné z WWW: <http://labs.developerfusion.co.uk>.

[5] EPRI|Electric Power Reasearch Institute [online]. 2011 [cit. 2011-05-04]. Dostupné z WWW: <http://pqdif.epri.com/>.

[6] YAO, Paul; DURANT, David. Programming .NET Compact Framework 3.5 : Second Edition. Addison-Wesley , 2009. 694 s. ISBN 978-0321573582.

[7] MAKOFSKY, Steve. Pocket PC Network Programming. [s.l.] : Addison-Wesley Professional, 2003. The .NET Compact Framework, s. 559-602. Dostupné z WWW: <http://my.safaribooksonline.com/book/programming/microsoft-pocket-pc/0321133528>. ISBN 978-0-321-13352-6.

[8] Electronic Tutorials [online]. 2011 [cit. 2011-05-08]. Electronics Tutorial about Phasor Diagrams. Dostupné z WWW: <http://www.electronics-tutorials.ws/accircuits/phasors.html>.

[9] support.microsoft.com [online]. 2005 [cit. 2011-05-13]. How to create a Setup package by using Visual Studio .NET. Dostupné z WWW: <http://support.microsoft.com/kb/307353/en-us?fr=1>.

[10] www.fengcool.com [online]. 2007 [cit. 2011-05-13]. Connect Windows Mobile 5.0 Smartphone Emulator to the Internet on Vista . Dostupné z WWW: <http://www.fengcool.com/2007/09/connect-windows-mobile-50-smartphone-emulator-to-the-internet-on-vista/>.