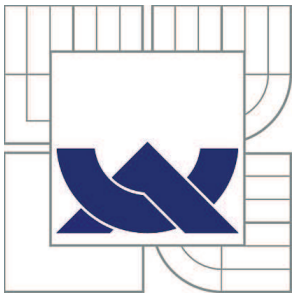
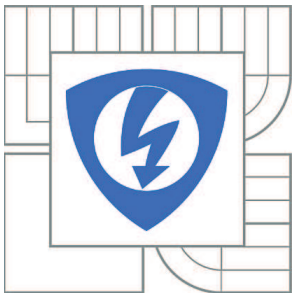


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

OVLÁDÁNÍ DIGITÁLNÍHO MULTIMETRU PŘES ROZHRANÍ GMC-USB

CONTROL OF DIGITAL MULTIMETER BY GMC-USB INTERFACE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK JANOŠÍK

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. FRANTIŠEK ZEZULKA, CSc.

SUPERVISOR

BRNO 2012

Vložit originální zadání bakalářské práce

Abstrakt

Cílem této bakalářské práce, je návrh a následné vytvoření uživatelské aplikace, pro komunikaci, ovládání a manipulaci s multimetry Metrahit.

Pro realizaci aplikace byl zvolen programovací jazyk C# a vývojové prostředí Microsoft Visual Studio 2010.

S pomocí vytvořeného software bude možné multimetry připojit k PC na vybraný COM port, připojené přístroje nastavovat, sledovat měřenou hodnotu a tyto hodnoty zpracovávat.

V této práci nalezneme popis vytvořené aplikace, rozbor jednotlivých instancí a funkcí programu, a návod jak jednotlivé prvky aplikace využívat.

Najdeme zde také krátký popis rozhraní, přes které jsou multimetry připojeny k PC, popis samotných multimetrů, jejich vlastnosti a použití.

Klíčová slova

GMC, Multimetr, Metrahit Energy, Interface, aplikace, C#,

Abstract

The aim of this thesis is the design and subsequent development of user applications for communication, control and manipulate with Metrahit multimeters.

To realize the application was chosen C # programming language and development environment Microsoft Visual Studio 2010.

With the developed software will be able to connect multimeters to a PC on selected COM port, the connected devices can be set, monitored the measured value and processing these values.

In this work, we will find a description of the created application, analysis of individual instances and features of the program. And instructions, how to use individual elements of the application.

There is also a short description of the interface which is used for connection between multimeters and PC, and description of multimeters themselves, their properties and uses.

Keywords

GMC, Multimetr, Metrahit Energy, Interface, application, C#,

Bibliografická citace:

JANOŠÍK, Z. Ovládání digitálního multimetru přes rozhraní GMC-USB. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 41s. Vedoucí bakalářské práce byl prof. Ing. František Zezulka, CSc.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Ovládání digitálního multimetru přes rozhraní GMC-USB jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **25. května 2012** ……
podpis autora

Poděkování

Děkuji panu Ing. Josefu Pazderkovi, za účinnou metodickou a odbornou pomoc a další cenné rady při řešení problémů v komunikaci s multimetry a objasnění neznámých pojmů.

Děkuji kamarádu Ondřeji Fibichovi za účinnou metodickou a odbornou pomoc a další cenné rady při řešení problémů s implementací software.

Děkuji vedoucímu bakalářské práce Prof. Ing. Františku Zezulkovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **25. května 2012** ……
podpis autora

Obsah

Seznam obrázků

1 ÚVOD

Pro mnohé uživatele měřicích přístrojů, hlavně těch, kteří s přístroji manipulují krátce nebo úplně poprvé, se může zdát jejich obsluha poměrně nepřehlednou a zdlouhavou záležitostí. Pro všechny uživatele, kteří hledají v měření elektrických i neelektrických veličin zbytečné složitosti, nebo pro ty, kteří si chtějí svoji práci zjednodušit a zpříjemnit. Nabízím možné řešení v podobě zpracování mé bakalářské práce.

Zabývá se návrhem a implementací uživatelského software, právě pro ovládání a manipulaci s multimetrem Metrahit Energy. Software je vhodný, jak pro úplné začátečníky, tak pro zkušené uživatele. Každý uživatel ocení zpřehlednění ovládacího rozhraní multimetru a možnost zpracování naměřených hodnot, pro další použití v digitální podobě.

2 METRAHIT

Nejprve trocha teorie, k použitým přístrojům. V kapitole si představíme měřící přístroj Metrahit Energy od společnosti Gossen Metrawatt.

Řekneme si, jaké jsou jeho možnosti a vlastnosti, co vše je možné s jeho pomocí změřit a jaké je použití v praxi.

Představíme si také rozhraní použité k připojení multimetru k PC.

2.1 Metrahit Energy

Metrahit Energy je vysoce přesný digitální ruční multimetr určený pro profesionální použití v oblasti měření a analýzy elektrické energie.

Metrahit Energy zaujme již na první pohled propracovaným designem, přehledně uspořádanými ovládacími prvky, spolehlivým a robustním nárazuvzdorným provedením pouzdra, jehož odolnost ještě zvyšuje standardně dodávaný pryžový obal. Velký modře podsvícený displej usnadňuje spolehlivé odečítání hodnot, i za zhoršených světelných podmínek. Přístroj je napájen dvěma bateriemi typu AA, lze využít i externího síťového adaptéru.

Metrahit Energy je osazen kvalitním TRMS (True Root Mean Square) převodníkem s rozlišením 60 000 digitů a 35 měřícími funkcemi.

Ze základních veličin dokáže měřit skutečné efektivní hodnoty napětí / proudů na rozsazích 60 mV ÷ 600 V / 600 μA ÷ 10 A s šířkou pásma 100 kHz. S přístrojem lze měřit i odpor (rozsahy 600 Ω ÷ 60 MΩ), kapacitu (rozsahy 60 nF ÷ 600 μF) a frekvenci (600 Hz ÷ 1 MHz). Dále pak multimetr nabízí měření teploty (Pt100, Pt1000, termočlánek typu K), měření vodivosti, nízkoohmová měření, měření střídy, test vodivosti, test diod a další.

Hlavní předností tohoto přístroje je však schopnost provádět měření a analýzu kvality elektrické energie. METRAHit ENERGY dokáže měřit jednofázově výkon, a to jak pro střídavý, tak i pro stejnosměrný proud. Přičemž proud lze měřit buď přímo, nebo pomocí proudového převodníku. Kromě činného, jalového a zdánlivého výkonu a účiníku, lze sledovat i maximální a minimální hodnoty za určitý časový interval. Adekvátně pak při sledování elektrické energie přístroj měří činnou, jalovou a zdánlivou složku a umožňuje sledovat maximální a minimální hodnoty v daném časovém úseku.

Co však tento přístroj zcela odlišuje od běžných digitálních multimetrů je přítomnost modulu pro analýzu sítě. Do paměti o kapacitě přes 300 000 měřených hodnot lze ukládat veškeré hodnoty a události, jako jsou přepětí, podpětí, špičky, výpadky, přechodové děje a další hodnoty charakterizující kvalitu elektrické energie, a to na sítích

s kmitočty 16.7, 50, 60 a 400 Hz. Díky integrovanému sofistikovanému matematickému aparátu zvládá přístroj rovněž výpočet harmonické analýzy až do 15. harmonické.

Veškeré hodnoty lze buď přímo sledovat na multifunkčním trojnásobném displeji, nebo je lze ukládat do paměti přístroje a z ní pak pomocí infračerveného komunikačního rozhraní přenést do počítače a následně zpracovat. Za zmínku jistě stojí i možnost plně dálkového ovládání multimetru pomocí počítače bez nutnosti otáčet přepínačem ručně.

Ovládání a řízení multimetru na dálku, se provádí posíláním předem definovaných řetězců z PC do multimetru. Na každý dotaz, který odešleme, multimetr vrací odpověď. Buď je to hodnota na, kterou se dotazujeme, nebo vrací zprávu „OK“ pokud žádáme o nastavení funkcí multimetru.

Více informací o přístroji naleznete v příloze č. 1. [1]

Obrázek 1: Metrahit Energy [2]

2.2 USB X-TRA Interface adapter

S použitím tohoto adaptéru, je možné připojit multimetry Metrahit vybavené sériovým IR rozhraním, k USB portu na jakémkoliv osobním počítači. To nám umožní přenos dat mezi multimetrem a PC. Protože interface komunikuje s multimetrem prostřednictvím IR paprsků, je zaručené galvanické oddělení multimetru a PC.

Jeden konec adaptéru jednoduše připojíme do volného USB portu. Druhý konec zapojíme do výřezů na horní straně multimetru, určených pro toto připojení. Adaptér nevyžaduje žádné externí napájení, je napájen skrze USB port.

Pro správnou funkčnost spojení je nutné nainstalovat ovladač, který v počítači vytvoří virtuální COM port, pod tímto portem pak v počítači nalezneme připojený multimetr. Ovladač nalezneme na přibaleném CD nebo lze stáhnout na stránkách výrobce. Podrobný návod k použití a detailní technické informace naleznete v příloze č. 2.

Obrázek 2: Interface adaptér připojený k multimetru [3]

3 APLIKACE

Zde se dostáváme k praktickému zpracování celého projektu, je potřeba říct, že podstatná část této práce, je v podobě software, který je přiložen na CD.

Tuto kapitolu si můžeme představit, jako kompletní dokumentaci k vyvájené aplikaci. Kapitola nás provede od vzniku aplikace až po testování, konkrétní příklad měření a zpracování výsledků měření. Vývoj aplikace rozebereme od počátečního návrhu a požadavků, až po konečnou realizaci a řešení jednotlivých problémů při implementaci. Taky si ukážeme, jak používat jednotlivé funkce aplikace a jak aplikaci efektivně využívat.

3.1 Analýza, zadání

Základním předpokladem vytvořené aplikace je její jednoduchost a nenáročnost na uživatele. Snadná ovladatelnost a přehlednost aplikace, aby uživatel i po krátké době užívání dokázal využít všechny potřebné funkce a možnosti aplikace, i multimetru, který jejím prostřednictvím ovládá.

Dalším důležitým požadavkem je správa dat (naměřených hodnot) uložených v paměti multimetru. Umožnit uživateli vyčtení jednotlivých bloků, nebo kompletní paměti multimetru a umožnit konverzi vyčtených hodnot na formát, který bude možné zpracovat a případně později využít. Například v programech Excel nebo Matlab.pro podrobnější analýzu měřených hodnot.

Program by měl komunikovat s multimetrem po USB, umožnit dálkové nastavení funkcí, rozsahů a jiných parametrů multimetru, zobrazení jeho aktuálního stavu, zobrazení hodnot displeje, popřípadě ostatních údajů zobrazených na displeji. Umožnit automatické vyhledávání multimetrů a konfiguraci připojení.

Veškeré ovládací a zobrazovací prvky aplikace vhodně zakomponovat do jednoduchého grafického rozhraní v podobě oken.

Aplikace má být kompatibilní se systémy Windows. První grafický návrh okna vypadal, viz Obrázek 3.

Soubor	Nastavení	Data	Script	Připojení	O programu	Help
Ulož data	Read setup	Data online		Výběr COMu		
Open data	Edit setup	Data store				
Export data	Ulož setup	Read mem				
	SET	Clear mem				
	Comand line					

Obrázek 3: Grafický návrh vzhledu okna aplikace

- V okně Nastavení zobrazit stránku parametrů multimetru s možností editace, následné uložení parametrů do souboru, respektive jejich načtení ze souboru.
- V okně Data, umožnit čtení dat, nahlížení do celého paměťového bloku multimetru a export vybraných bloků nebo celého obsahu paměti do souboru.
- Okno připojení sloužící pro výběr COM portu a nastavení připojení.
- Aplikace realizována jako MDI (Multi Dialog Interface), ovládání pomocí menu v hlavním okně, které umožňuje přístup k dalším dialogových oknům, s možností využití toolbaru, pro rychlejší přístup k některým funkcím.

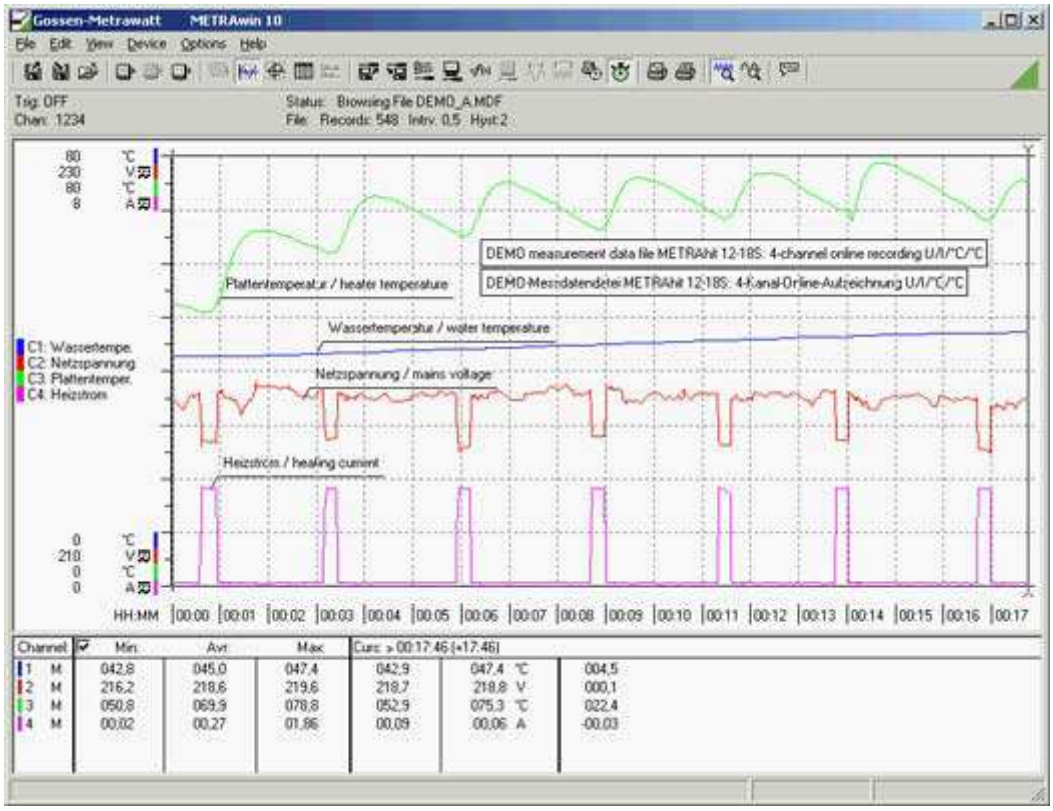
3.2 Porovnání

Společnost Gossen Metrawatt má pro své multimetry Metrahit vytvořený vlastní software MetraWin10, který je poměrně sofistikovaný a složitý na obsluhu. A chvíli vám potrvá, než se s ním dokážete sžít. Nicméně multimetry Metrahit připojené k PC, se společně s tímto softwarem, stávají profesionálním systémem pro záznam měřených hodnot.

Měřené hodnoty jsou přijímány z multimetru a zpracovávány v MetraWin10 a můžou být zobrazeny v grafu jako Yt nebo XY (až pro 6 kanálů), nebo v tabulkové formě (až 10 kanálů). Naměřená data jsou zpracovávána online, později mohou být data exportována do ASCII souboru, nebo exportována jako soubor spustitelný ve Windows aplikacích (Word, Excel, apod.).

S pomocí MetraWin10 však nelze zasílat jednotlivé příkazy ve formě řetězců, proto nelze nastavovat veškeré funkce a parametry multimetru, nelze získávat všechny interní informace o multimetru, nebo získat data přímo z paměťového bloku.

MetraWin10 je určen především pro záznam a zobrazení měřených hodnot.



Obrázek 4: Okno programu MetraWin10 [4]

3.3 Implementace

Aplikaci jsem se rozhodl psát v programovacím jazyce C# a využil jsem k tomu vývojové prostředí Microsoft Visual Studio 2010. Protože, jsem se s programováním, jako takovým setkal jenom na školních projektech, kde jsme používali výhradně Microsoft Visual Studio (MSV), je mi toto vývojové prostředí (IDE) nejblíže. V MSV 2010 jsou integrovány i nástroje pro vývoj grafického rozhraní aplikací.

S programovacím jazykem C#, jsem se sice nikdy dříve nesetkal, ale jeho syntaxe vychází z jazyka C a je tak podobná i C++. Proto nebyl problém, se tomuto modernímu a poměrně jednoduchému nástroji přizpůsobit a hlavně jej postupem času poznat a získat potřebné zkušenosti. Jak se samotným C# tak s programováním obecně.

Aplikace je dle návrhu řešena jako MDI, tzn. že prostřednictvím hlavního okna (ParentForm) můžeme otevřít další okna (ChildForm).

Každé okno (Form) je v IDE sestaveno z několika souborů, k těm hlavní patří:

- Form_Name.cs – pro nás asi nejdůležitější soubor. Každé okno je třída (instance), a právě v tomto souboru si může programátor deklarovat a definovat vlastní proměnné, vytvářet nové metody (funkce) a událostem jako je např. stisk tlačítka, výběr položky v menu apod. přiřadit cokoliv, co má daná událost vykonat.

- Form_Name.cs [Design] – soubor určený k tvorbě grafického rozhraní, pomocí prvků (tlačítko, jmenovka, textové pole, apod.) umístěních v toolboxu IDE, můžeme sestavit námi požadovaný vzhled každého okna. Vlastnosti jednotlivých prvků lze nastavit, staticky zde v grafickém rozhraní, nebo později vlastnosti měnit dynamicky při běhu programu.

- Form_Name.Designer.cs – tento soubor generuje IDE automaticky, jsou zde deklarovány veškeré grafické prvky obsažené v okně.

Každé okno v aplikaci je nastaveno jako FixedDialog, tzn. při běhu programu nelze měnit velikost okna a otevřené můžeme mít maximálně jedno ChildForm, vždycky můžeme pracovat pouze v jednom aktivním okně.

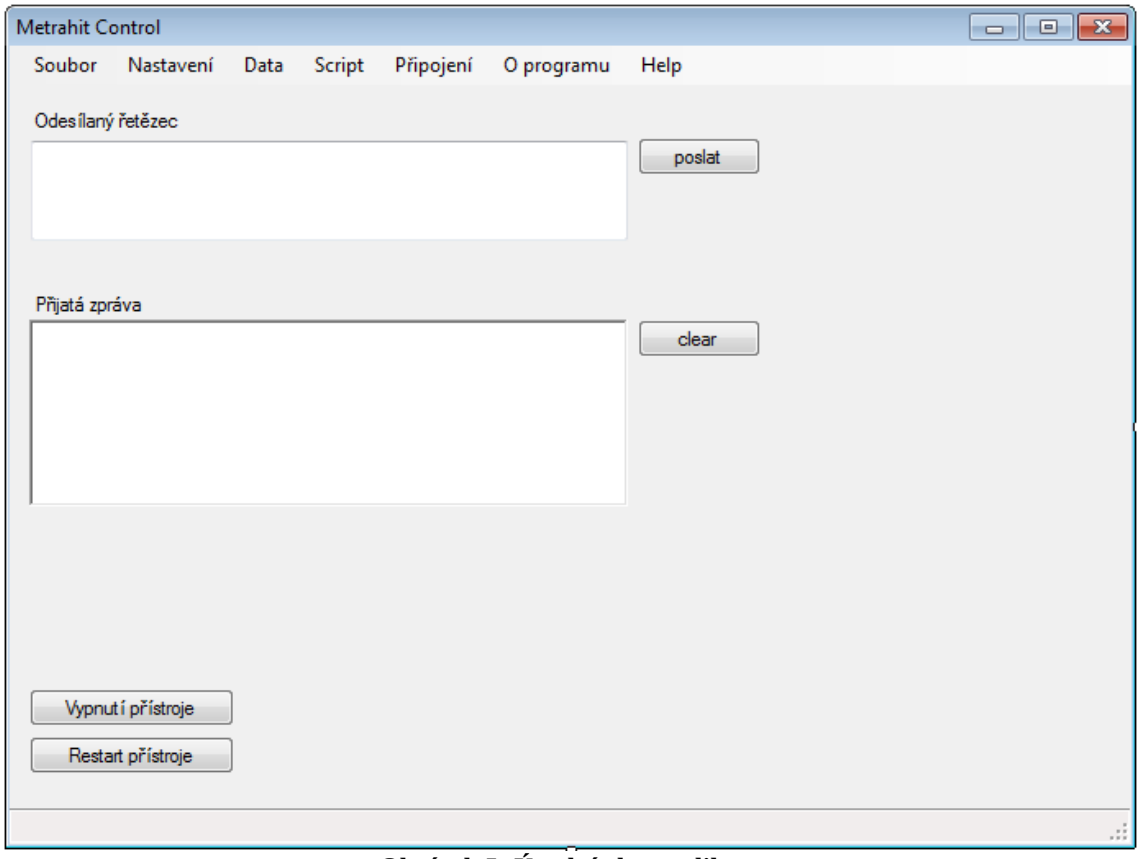
V aplikaci je použito šest oken, v následujících kapitolách si je představíme po designové stránce. Popíšeme jejich ovládací a zobrazovací prvky, to k čemu jsou určeny a jak je požívat. Rozebereme jejich programové řešení, problémy při implementaci a další problematiku na úrovni zdrojového kódu aplikace.

3.3.1 Hlavní okno „Metrahit Control“

Prvním krokem v tvorbě mého projektu, bylo vytvoření okna typu `Form` pojmenovaného „Metrahit Control“. Okno je tzv. ParentForm. Hlavní okno obsahuje menu, s pomocí jeho položek můžeme otevřít další okna programu tzv. ChildForm, které si probereme v dalších kapitolách. Dále zde najdeme dvě textová okna, jedno pro psaní řetězců posílaných do multimetru, druhé pro zobrazení příchozích zpráv z multimetru. Použita je zde i čtveřice tlačítek, které už podle jejich názvu napovídají, k čemu jsou určena.

Tlačítko „poslat“ slouží k posílání řetězců, zapsaných v okně „Odesílaný řetězec“ přes sériový port do multimetru. Posílá se vždy celý textový obsah okna, proto je nutné příkazy přepisovat a ne psát po řádcích. V okně „Přijatá zpráva“ se téměř okamžitě zobrazí odezva na poslaný příkaz. Abychom mohli zahájit komunikaci mezi PC a multimetrem, je nutné nastavit a připojit sériový port viz kapitola 3.3.2.

Funkce tlačítek Vypnout a Restartovat přístroj, je jasná. Tyto tři tlačítka využívají pro přenos dat volání funkce `write(string str)`, viz kapitola 3.3.1.1. Tlačítko „clear“ slouží pouze pro vymazání textu v okně „Přijatá zpráva“.



Obrázek 5: Úvodní okno aplikace

3.3.1.1 Funkce write

Funkce Write je vpodstatě jediná funkce, která je v aplikaci použita ke komunikaci s multimetrem a přenosu dat. V celé aplikaci jsou dva typy této funkce, `public void write(string str),` funkce nevrací nic, řetězec, který je přijímán z multimetru, je přímo v těle funkce zobrazován do textového pole, tento typ funkce je využit pouze v okně „Metrahit Control“. Druhý typ funkce je `public string write(string str)`, vrací řetězec přijatý z multimetru a funkce je využívána ve zbytku celé aplikace. Jinak je principiálně tělo funkcí shodné.

Funkci write je předáván jeden parametr typu string, je to řetězec, který je odesílán na sériovou linku.

Ve funkci, se nejprve otestuje, zda je sériový port otevřen, pokud ne, vypíše se chybové hlášení „Port není otevřen“. V opačném případě se předávaný řetězec převede na ASCII a přidá se k řetězci ukončovací sekvence `"\r\n"`. Následně řetězec odešleme na sériový port a ve smyčce se čeká na odpověď z multimetru. Odpověď se zobrazí v textovém poli pro přijaté zprávy, nebo je funkcí vrácena jako návratová hodnota.

Pokud by došlo ke ztrátě spojení mezi PC a multimetrem, mohl program uvíznout ve smyčce a čekat na odpověď donekonečna, proto je čekací doba omezena na 3 sekundy, při vypršení tohoto intervalu se vypíše chybové hlášení „Chyba spojení“.

Aby multimetr stíhal odpovídat na zřetězené zprávy je ve funkci `write` zavedeno pozastavení běhu programu na 50ms.

```
public void write(string str)
        {    try
            {
                if (!port1.IsOpen) { sl.Text = "Port není otevřen...";
return; }
                /* Posleme retezec na seriovou linku */
                byte[] asciiBytes = null;
                asciiBytes = Encoding.ASCII.GetBytes(str+crlf);
                port1.Write(asciiBytes,0,asciiBytes.Length);
                Thread.Sleep(50);
                string s = string.Empty;
                DateTime a = DateTime.Now;
                DateTime plus5 = a.AddSeconds(3);
                while (port1.BytesToRead == 0)
                {
                    DateTime aktual = DateTime.Now;
                    int c = DateTime.Compare(aktual, plus5);
                    if (c > 0) { sl.Text = err; return; }
                }
                s = port1.ReadExisting();
                rtb_received.AppendText(s);
            }
            catch (Exception)
            {
                /* V pripade nezdaru vypiseme chybove hlaseni */
                sl.Text = "Chyba při posílání...";
            }
        }
```

3.3.1.2 Fukce volaná k otevření ChildForm

Pokaždé když v menu hlavního okna vypereme položku k otevření ChildForm, vytvoří se událost, která volá níže uvedenou část zdrojového kódu.

```
private void měřeníToolStripMenuItem_Click(object sender, EventArgs e)
       {
              measure meas = new measure();
          meas.FormBorderStyle = FormBorderStyle.FixedDialog;
          meas.ShowDialog();
       }
```

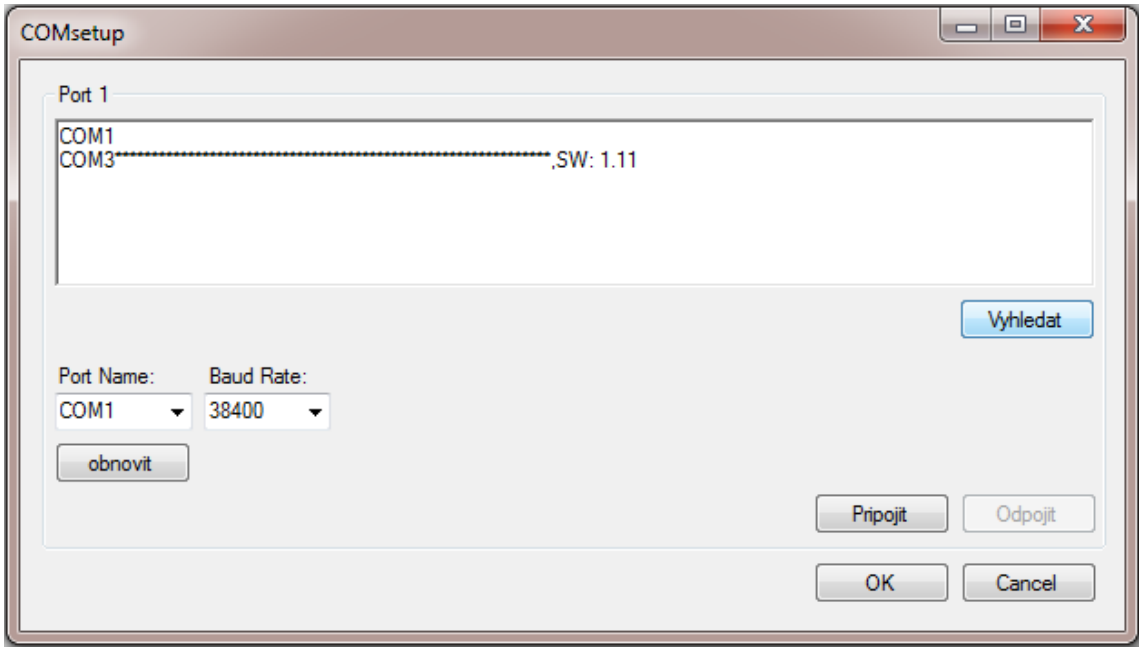
Zde konkrétně je volání pro zobrazení okna „Měření“. Nejprve se deklaruje proměnná typu `measure`, pak se okno nastaví jako FixedDialog a nakonec se zobrazí. Otevření jiných oken je principiálně naprosto stejné.

3.3.2 Okno „COMsetup“

Jedna z nejdůležitějších součástí programu. Abychom mohli zahájit datovou komunikaci mezi multimetrem a PC, je třeba se připojit na virtuální COM port, pod kterým je multimetr fyzicky připojen k PC.

V okně „COMsetup“ je nám právě výběr a konfigurace COM portu umožněna. V jedné z roletových nabídek volíme jméno COM portu a ve druhé rychlost připojení portu. Již při otevření „COMsetup“ se veškeré dostupné porty přidají do roletové nabídky se jmény portů, stiskem tlačítka „obnovit“ nabídku jmen portů aktualizujeme. Výběr parametrů potvrzujeme tlačítkem „Připojit“. Ostatní parametry, jako počet stop bitů, počet data bitů a parita, jsou nadefinovány staticky, protože jsou pro všechny multimetry stejné a není potřeba jejich hodnoty měnit. Stiskem tlačítka „Připojit“ nadefinujeme parametry třídy SerialPort, která je implementována přímo v IDE a otevřeme sériový port, více o této třídě v kapitole 3.3.2.3.

COMsetup, také umožňuje vyhledávání připojených multimetrů k PC, po stisku tlačítka „Vyhledat“, je volána funkce, která na každý COM port PC pošle příkaz „IDN?“, pokud je na portu připojen multimetr, odešle odpověď o verzi svého firmwaru. Odpověď se vypíše do textového okna společně s informací, na kterém portu se multimetr nachází.



Obrázek 6: Výpis nalezeného multimetru

Po připojení můžeme okno zavřít a pokračovat v další práci, při znovuotevření „COMsetup“, se v textovém okně zobrazuje, zda je port připojený.

3.3.2.1 Funkce „Připojit“

Definuje parametry třídy SerialPort, otvírá port, dynamicky mění možnosti ovládacích prvků v okně a oznámí, zda bylo připojení úspěšné či nikoliv.

```
private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                // Nastavime vybrany port ke komunikaci
                Form1.port1.PortName = cb_port.Text;
                //Nastavime jeho rychlost
                Form1.port1.BaudRate = Convert.ToInt32(cb_baud.Text);
                //nastaveni DataBits
                Form1.port1.DataBits = 8;
                //nastaveni timeouts
                Form1.port1.ReadTimeout = 500;
                Form1.port1.WriteTimeout = 500;
                //Otevreme spojeni
                Form1.port1.Open();
                /* Pokud jsme pripojeni, dynamicky zmenime moznosti
okna */
                if (Form1.port1.IsOpen)
                {
                    /* Do status baru vypiseme soucasny stav */
                    Form1.sl.Text = "Připojeno na port " +
cb_port.Text + " rychlostí " + cb_baud.Text;
                    /* Vypneme/zapneme prvky v okne */
                    cb_port.Enabled = false;
                    cb_baud.Enabled = false;
                    bt_pripojit.Enabled = false;
                    bt_odpojit.Enabled = true;
                    bt_vyhledat.Enabled = false;
                    bt_obnovit.Enabled = false;
                    Form1.poslat.Enabled = true;
                    if (!Form1.port1.IsOpen) { Form1.sl.Text = "Port
není otevřen..."; return; }
                }
            }
            catch (Exception)
            {
                /* V pripade nezdaru vypiseme chybove hlaseni*/
                Form1.sl.Text = "Nepodařilo se otevřít " +
cb_port.Text + " port...";
            }
        }
```

3.3.2.2 Funkce „Odpojit“

Funkce uzavírá port, uvolňuje paměť, dynamicky mění ovládací prostředky v „COMsetup“ a vypíše výsledek odpojení.

```
private void odpojit_Click(object sender, EventArgs e)
        {
            try
            {
                 /* Zavreme spojeni */
                 Form1.port1.Close();
                 Form1.port1.Dispose();
                if (!Form1.port1.IsOpen)
                 {
                    /* Do status baru vypiseme soucasny stav */
                    Form1.sl.Text = "Odpojeno...";
                    /* Zapneme/vypneme prvky v okne */
                    cb_port.Enabled = true;
                    cb_baud.Enabled = true;
                    bt_pripojit.Enabled = true;
                    bt_odpojit.Enabled = false;
                    bt_obnovit.Enabled = true;
                    bt_vyhledat.Enabled = true;
                    Form1.poslat.Enabled = false;
                }
            }
            catch (Exception)
            {
           /* V pripade nezdaru vypiseme chybove hlaseni že port nelze
zavřít*/
         Form1.sl.Text = "Vyskytla se chyba při pokusu odpojení se od"
                       + Form1.port1.PortName + " portu ...";
            }
        }
```

3.3.2.3 Třída SerialPort

Protože je komunikace po sériové lince poměrně složitou záležitostí, rozhodl jsem se využít, již v IDE implementovanou třídu SerialPort, která obsahuje několik konstruktorů, mnoho vlastností a metod.

V aplikaci využívám pouze implicitní konstruktor `SerialPort()` této třídy. A parametry, jako `PortName`, `BaudRate`, `DataBits`, `Parity` a `StopBits` nastavuji, nebo získávám až při běhu programu dynamicky. Z metod (funkcí), užívám `BytesToRead` – vrací počet bajtů, které je možné přečíst, ze sériové linky, `Close` – uzavře SerialPort, `GetPortNames` – vyhledá všechny porty na PC, `Open` – otevře SerialPort, `ReadExisting` – přečte veškerá dostupná data na sériové lince, `Write(byte[], Int32, Int32)` – zapíše na sériovou linku určitý počet bajtů dat.

3.3.3 Okno „Měření“

Prostředí okna „Měření“ nám plně nahrazuje otočný přepínač měřících funkcí a z části i displej multimetru. Najdeme zde i možnost volby vzorkovací frekvence ukládání dat do paměti, spouštění a vypínání ukládání dat, zobrazení zaplnění paměti a spouštění zobrazování minim a maxim z měření.

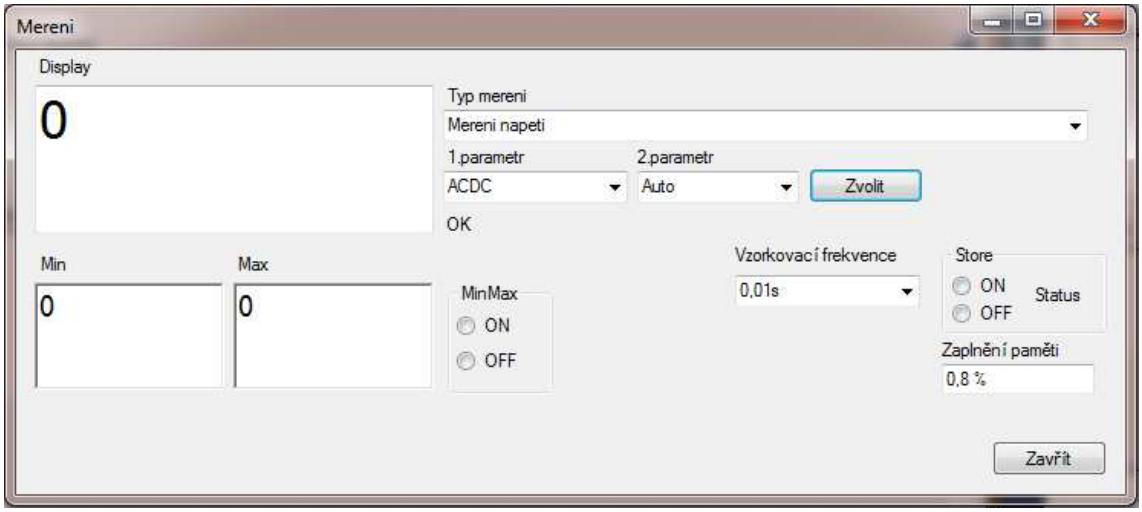
Každé tři sekundy se obnovuje hodnota displeje, spolu s hodnotami minim a maxim pokud je funkce zapnuta. A také údaj o zaplnění paměti je obnovován ve stejném intervalu.

Všechny číselné hodnoty jsou z multimetru posílány v exponenciálním tvaru, aby byly hodnoty v textových oknech čitelnější a přehlednější, jsou převáděny na čísla v desetinném tvaru voláním funkce `parse` viz kapitola 3.3.3.1. V textovém poli, kde se zobrazuje kopie displeje, však nejsou zobrazeny žádné jednotky. Pokud tuto hodnotu chceme odečítat, je potřeba dbát na to jakou měřící funkci máme nastavenou a s jakým rozsahem. To platí i pro textová pole, kde se zobrazují minima a maxima.

Na pravé straně okna „Měření“, je několik prostředků k ovládání ukládání dat. V roletovém menu volíme vzorkovací frekvenci v rozmezí od setin sekundy až po násobky hodin. Výběrem hodnoty zrovna posíláme požadavek do multimetru a tedy i potvrzujeme volbu. To samé platí pro zapínání a vypínání funkce „Store“, tedy vzorkování měřených hodnot do paměti multimetru.

Nastavení měřící funkce a jejich parametrů provádíme pomocí tří roletových menu, obsah menu s parametry se mění dynamicky v závislosti na zvolené funkci. Výběr funkce a příkaz pro nastavení multimetru potvrdíme tlačítkem „Zvolit“. Pro dynamicky měnící se obsah menu jsem implementoval několik tříd viz kapitola 3.3.3.2.

Nastavení parametrů multimetru je dosaženo pomocí předání příslušných příkazů (řetězců) funkci `write`, jejíž návratové hodnoty jsou následně zobrazeny, jako hlášení o stavu, nebo hodnoty displeje.



Obrázek 7: okno "Měření"

3.3.3.1 Funkce „parse“

Funkci typu `string` je předáván jako parametr řetězec, který chceme převést. Funkce nejprve testuje, zda je otevřen sériový port a zda je předávaný řetězec ve správném tvaru. Pokud vše projde, je řetězec převeden na desetinné číslo typu `double` a následně na typ `string`, aby mohl být zobrazen v textovém okně.

```
public string parse(string a)
        {
            try
            {
                string noport = "Port není otevřen...";
                if (a == "Port není otevřen...") { return noport; }
                double test = double.Parse(a,
System.Globalization.CultureInfo.InvariantCulture);
                string ret = test.ToString();
                return ret;
            }
            catch (Exception) {return "Spatny format"; }
        }
```

3.3.3.2 Třídy pro dynamickou změnu roletových menu

Implementoval jsem dvě jednoduché třídy, do kterých jsem zapouzdřil jména zobrazená v roletových menu a ke každému jménu přiřadil řetězec, který se odesílá do multimetru jako příkaz.

Třída `Type`, obsahuje tři členské proměnné, `name` – představuje jméno, zobrazené v roletovém menu, pro výběr typu měření, `sender` – příkaz odesílaný do multimetru, pro daný typ měření, `par1` – třída `param` ve které jsou zapouzdřeny proměnné umožňující dynamickou změnu roletových menu pro nastavení rozsahů multimetru.

Ve třídě `Type` je jeden konstruktor, kterému jsou předávány tři parametry a tři vlastnosti, pomocí kterých nastavujeme a získáváme hodnoty členských proměnných.

Pro každý typ měření, je deklarována jedna třída `Type`. Třídy `Type` jsou definovány při spuštění okna „Měření“, poté jsou vloženy do Třídy `types` typu `List<Type>` a jejich jména jsou přidána do roletového menu.

```
class Type
        {
            private string name;
            private string sender;
            private param par1 = new param();
            public Type(string name,string sender,param par)
            {
                this.name = name;
                this.sender = sender;
                this.par1 = par;
```

```
            }
            public string Name
            {
                get { return name; }
                set { name = value; }
            }
            public string Sender
            {
                get { return sender; }
                set { sender = value; }
            }
            public param Par
            {
                get { return par1; }
                set { par1 = value; }
            }
        }
```

Třída `param` obsahuje šest členských proměnných, `n1` – představuje jména zobrazená v roletovém menu pro první parametr, `s1` – těmto jménům asociované řetězce odesílané do multimetru. Stejně tak to platí pro druhý parametr roletového menu, `n2` – jména, `s2` – řetězce. Tyto čtyři proměnné jsou typu `List<string>`, při definici třídy `param` se konstruktoru jako parametry předávají pole `stringů` a ty jsou postupně přidávány a indexovány právě do těchto čtyř proměnných.

`active1` – aktivuje roletové menu pro první parametr, `active2` – aktivuje roletové menu pro druhý parametr.

Najdeme zde dva konstruktory, jeden implicitní a druhý se šesti parametry. Stejně tak, je zde, šest vlastností, k nastavení a získání hodnot členských proměnných.

Hodnoty třídy `param` se definují při výběru typu měření, v závislosti na výběru se dynamicky změní obsah roletových menu.

```
class param
        {
            private List<string> n1 = new List<string>();
            private List<string> s1 = new List<string>();
            private List<string> n2 = new List<string>();
            private List<string> s2 = new List<string>();
            bool active1;
            bool active2;
            public param() { }
            public param(string[] name1, string[] sender1,string[]
name2, string[] sender2,bool a1,bool a2)
            {
                for (int i = 0; i < name1.Length; i++)
                {
                    n1.Add(name1[i]);
```

```
                s1.Add(sender1[i]);
            }
            for (int i = 0; i < name2.Length; i++)
            {
                n2.Add(name2[i]);
                s2.Add(sender2[i]);
            }
            this.active1 = a1;
            this.active2 = a2;
        }

        public bool A1
        {
            get { return active1; }
            set { active1 = value; }
        }
        public bool A2
        {
            get { return active2; }
            set { active2 = value; }
        }
        public List<string> N1
        {
            get { return n1; }
            set { n1 = value; }
        }
        public List<string> S1
        {
            get { return s1; }
            set {s1 = value;}
        }
        public List<string> N2
        {
            get { return n2; }
            set { n2 = value; }
        }
        public List<string> S2
        {
            get { return s2; }
            set {s2 = value;}
        }
    }
```

Výběr měření a nastavení obou parametrů, respektive rozsahů, potvrdíme tlačítkem „Zvolit“. Stiskem tlačítka, je volána funkce, která ze tříd `Type` a `param` získá hodnoty, sloučí je do jednoho řetězce a voláním funkce `write` je odešle do multimetru, pokud je vše v pořádku, multimetr odpoví „OK“.

```
private void bt_meas_Click(object sender, EventArgs e)
        {
            string send = null;
            Type selectedType = cb_meastype.SelectedItem as Type;

            if (selectedType != null)
            {
                send += selectedType.Sender;
            }


            int p1 = cb_1par.SelectedIndex;
            int p2 = cb_2par.SelectedIndex;
            if (selectedType.Par.A1 == true)
            {
                send += selectedType.Par.S1[p1];
            }
            if (selectedType.Par.A2 == true)
            {
                send += selectedType.Par.S2[p2];
            }
            lab_status.Text = write(send);
        }
```

3.3.4 Okno „Setup“

Rozhraní okna je rozčleněno na několik skupin, každá skupina slouží pro zobrazení nebo naopak nastavení určitého parametru multimetru. Vyjma skupiny umístěné v levé horní části, ta je pouze informativní a jsou zde zobrazeny informace o připojeném multimetru. Najdeme zde verzi firmwaru, typ multimetru, vnitřní teplotu multimetru, stav baterie a využití paměti. Hodnoty se aktualizují každých 5 vteřin.

Každá skupina je pojmenována dle parametru, ve skupině je zobrazeno aktuální nastavení parametru a textové okno, nebo roletová nabídka, pro volbu nové požadované hodnoty parametru.

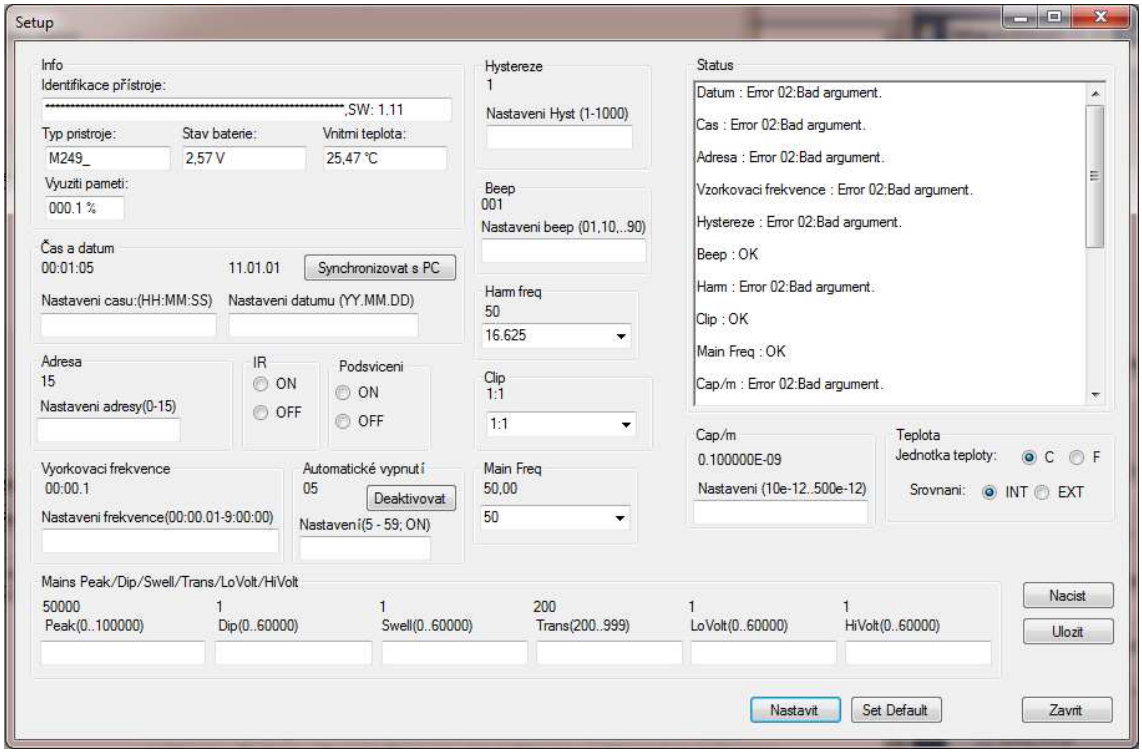
Abychom parametr opravdu nastavili a příkaz poslali do multimetru, je nutné stisknout tlačítko „Nastavit“, tím je volána několikanásobně funkce `write` a postupně odešle všechny textové řetězce (požadované hodnoty parametrů) do multimetru.

Na každou zprávu odeslanou do multimetru, dostáváme zpětně odpověď, odpovědi se nám zobrazují ve velkém textovém okně v pravé části okna „Setup“¨. Tedy hned získáme přehled o tom, zda se parametry multimetru podařilo úspěšně nastavit, popřípadě kde v zadávání parametru se vyskytla chyba.

Pokud si chceme námi vytvořenou konfiguraci nastavení uložit, provedeme tak tlačítkem „Uložit“, tím se řetězce napsané v textových polích uloží do souboru. Naopak je lze ze souboru načíst tlačítkem „Načíst“ a následně poslat do multimetru tlačítkem „Nastavit“.

Nastavení data a času je možné, ručním zadání hodnot, nebo tlačítkem „Synchronizovat s PC“, tím nastavíme do multimetru datum a čas shodný s PC.

Uvedení multimetru do defaultního nastavení provedeme stiskem „Set Default“. Jak celé okno funguje a funkce, které jsou volány pomocí tlačítek, si ukážeme v následujících podkapitolách.



Obrázek 8: Okno "Setup"

3.3.4.1 Rozbor funkcí okna “Setup“

Veškeré příkazy jsou posílány a přijímány pomocí volání funkce `write` ostatně jako v celém programu viz 3.3.1.1.

Ukládání a načítání konfigurace celého okna „Setup“, se prování serializací, respektive deserializací do a z XML souboru. Pro tento účel, jsem si implementoval vlastní třídu `XMLsetup`, s tolika vlastnostmi, kolik řetězců a proměnných potřebuji ukládat.

```
public class XMLsetup
          {
              public XMLsetup(){}
              public string time { get; set; }
              public string date { get; set; }
              public string adress { get; set; }
              public string rate { get; set; }
              public string hyst { get; set; }
```

```
        public string beep { get; set; }
        public bool iron { get; set; }
        public bool iroff { get; set; }
        public bool bcklon { get; set; }
        public bool bckloff { get; set; }
        public string harm { get; set; }
        public string main { get; set; }
        public string clip { get; set; }
        public string camp { get; set; }
        public bool tempC { get; set; }
        public bool tempF { get; set; }
        public bool tempINT { get; set; }
        public bool tempEXT { get; set; }
        public string trigP { get; set; }
        public string trigD { get; set; }
    }
```

Stiskem tlačítka „Uložit“ se nejprve hodnoty řetězců textových polí a vybrané položky roletových menu, uloží do třídy `XMLsetup`. Po uložení je volána funkce `SerializeToXML`, které je jako parametr předána třída `XMLsetup` s uloženými hodnotami.

```
static public void SerializeToXML(XMLsetup setup)
        {
            string savepath;
            SaveFileDialog savedialog = new SaveFileDialog();
            savedialog.Filter = "XML file (*.xml)|*.xml";
            savedialog.RestoreDirectory = false;
            savedialog.Title = "Kam chces ulozit XML soubor?";
            savedialog.CheckPathExists = true;
            savedialog.InitialDirectory = Application.StartupPath;


            if (savedialog.ShowDialog() == DialogResult.OK)
             {
                  savepath = savedialog.FileName;
                  using (Stream savestream = new FileStream
                  (savepath, FileMode.Create))
                 {
                  XmlSerializer serializer = new XmlSerializer
                  (typeof(XMLsetup));
                     serializer.Serialize(savestream, setup);
                 }
```

```
            }
            savedialog.Dispose();
            savedialog = null;

        }
```

Funkce `SerializeToXML` nejprve otevře okno typu `SaveFileDialog`, s možností výběru adresáře, kam chceme uložit soubor, a taky souboru přidělíme jméno. Pokud otevření dialogového okna proběhne korektně, jsou hodnoty uložené ve třídě `XMLsetup` přepsány do XML souboru pomocí funkce `FileStream`. Po uložení souboru, se dialogové okno zavře a uvolní paměť.

Tlačítko „Načíst“ funguje na podobném principu jako „Uložit“, jen volání funkcí postupuje v obráceném pořadí. Nejprve je volána funkce `DeserializeFromXML`, která má návratovou hodnotu typu `XMLsetup`.

```
static public XMLsetup DeserializeFromXML()
        {
            OpenFileDialog opendialog = new OpenFileDialog();
            opendialog.Multiselect = false;
            opendialog.Filter = "XML file (*.xml)|*.xml";
            opendialog.Title = "Vyber XML soubor";
          XMLsetup load = new XMLsetup();
          if (opendialog.ShowDialog() == DialogResult.OK)
            {
            string loadpath = opendialog.FileName;
            FileStream loadstream = new FileStream(loadpath,
FileMode.Open);
          XmlSerializer deserializer = new XmlSerializer
(typeof(XMLsetup));
          load = (XMLsetup)deserializer.Deserialize(loadstream);
            loadstream.Close();
            return load;
            }
            opendialog.Dispose();
            return load;
            }
```

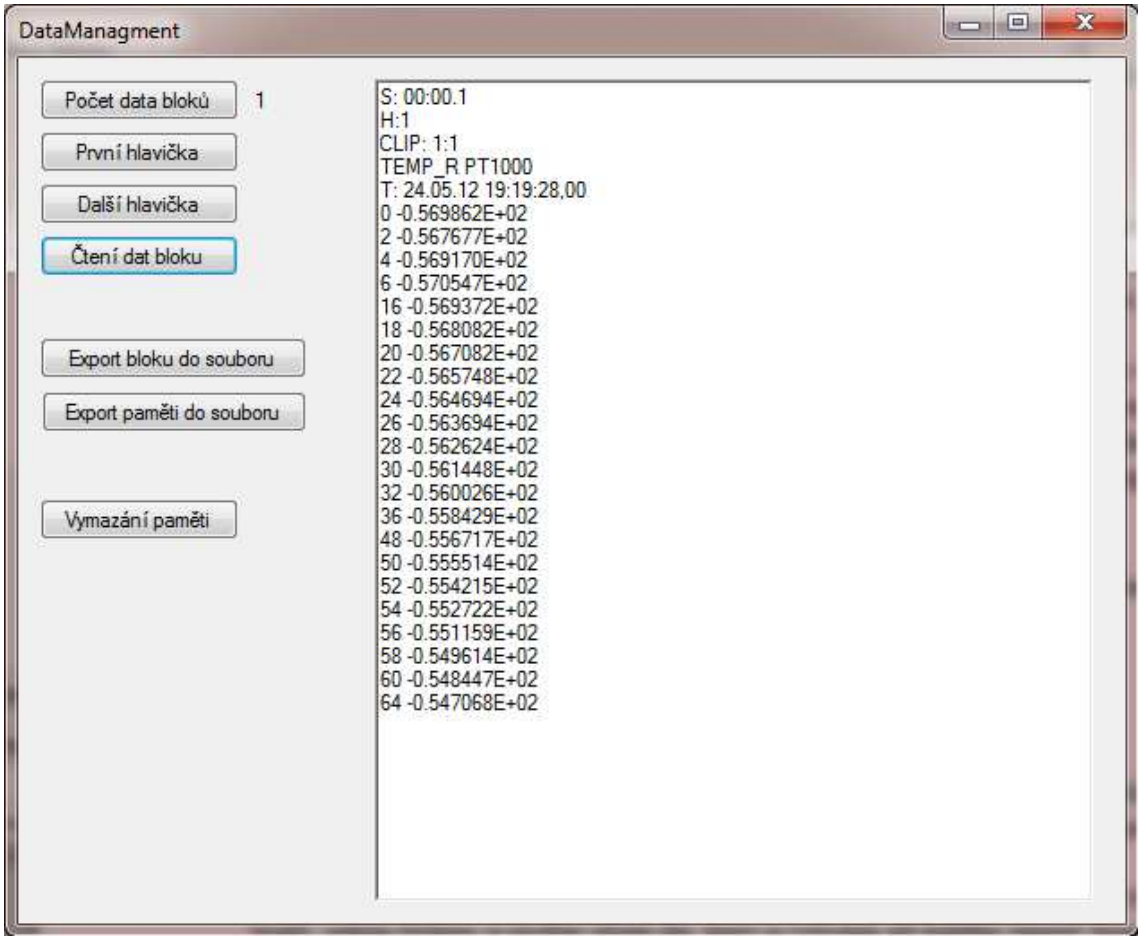
Podobně jako v `SerializeToXML`, i zde se otvírá dialogové okno, ale typu `OpenFileDialog`, pro výběr požadovaného XML souboru. Po úspěšném otevření souboru, je obsah XML souboru funkcí `FileStream` převeden a uložen do třídy `XMLsetup` a hodnoty ze třídy jsou postupně načteny do textových polí okna „Setup“.

3.3.5 Okno „DataManagment“

Jak již název napovídá, jedná se o prostředí pro zprávu dat uložených v paměti multimetru. Protože je zde více tlačítek, probereme si jejich funkci v krátkém seznamu:

- „Počet data bloků – vrací číselnou hodnotu datových bloků. Jedním blokem je myšlen objem dat, který se uloží při každém zapnutí Store“.
- „“První hlavička“ – vrací první hlavičku uloženou v paměti, hlavička je vždy na počátku každého bloku a je tvořena při zapnutí Store. Hlavička obsahuje informace o vzorkovací frekvenci, typu měření, datu a času vytvoření. Někdy jsou uvedeny i informace o hysterezi, nebo jiných parametrech multimetru, pokud jsou nastaveny.
- „Další hlavička“ – zobrazí každou další hlavičku paměťového bloku.
- „Čtení dat bloku“ – při každém kliknutí, postupně zobrazuje hodnoty uložené v paměti.
- „Export bloku do souboru“ – uloží do souboru jeden blok paměti bez hlavičky.
- „Export paměti do souboru“ – uloží do souboru cely obsah paměti multimetru, včetně hlaviček bloků.
- „Vymazání paměti“ – smaže celou paměť
-

Všechny tlačítka využívají pouze volání funkce `write` viz kapitola 3.3.1.1 a posílají do multimetru příkaz, kterým je docíleno požadovaného efektu.
Implementace tlačítek pro export paměti do souboru je složitější, proto si je popíšeme v kapitole 3.3.5.1.



Obrázek 9: Okno "DataManagment"

3.3.5.1 Tlačítko „Export do souboru“

Zdrojový kód vykonávaný po stisku „Export paměti do soboru“. Jeho implementace je složitější než „Export bloku do souboru“. Část kódu je však stejná, pokud tedy popíšeme tuto funkci, nebude problém s pochopením druhého tlačítka.

```
private void button1_Click(object sender, EventArgs e)
        {
            var pos1 = new List<string>();
            var pos2 = new List<string>();
            int isnumber = 0;
            var all = new List<string>();
            int pocet =Convert.ToInt32(write("rec?"));
            string hlavicka;
            for (int i = 0; i < pocet; i++)
            {
                if (i == 0) { hlavicka = write("rdf?"); }
                else { hlavicka = write("rdnh?"); }
                string[] splitHlavicka = hlavicka.Split('\r');
```

```
            for (int c = 0; c < splitHlavicka.Length; c++)
            {
                pos1.Add(splitHlavicka[c]);
                pos2.Add("");
            }
            for (; ; )
            {
                string a = write("rdn?");
                if (a == "\r\n") { break; }
                else
                {
                    a = a.Replace("\r\n", string.Empty);
                    string[] t = a.Split(' ');
                    bool test = int.TryParse(t[0], out isnumber);
                    if (!test)
                    {
                        string dohromady = string.Join(" ", t);
                        string[] hlav = dohromady.Split('\r');
                        for (int c = 0; c < hlav.Length; c++)
                        {
                            pos1.Add(hlav[c]);
                            pos2.Add("");
                        }
                                                }
                    else
                    {
                        pos1.Add(t[0]);
                        pos2.Add(t[1]);
                    }
                }
            }
        }

        for (int c = 0; c < pos1.Count; c++)
        {
            all.Add(pos1[c] + ";" + pos2[c] + ";");
        }
        string savepath;
        SaveFileDialog savedialog = new SaveFileDialog();
        savedialog.Filter = "CSV file (*.csv)|*.csv";
        savedialog.Title = "Kam chces ulozit CSV soubor?";
        savedialog.InitialDirectory = Application.StartupPath;
        if (savedialog.ShowDialog() == DialogResult.OK)
        {
            savepath = savedialog.FileName;
            String csv = String.Join(Environment.NewLine,all);
            System.IO.File.WriteAllText(savepath, csv);
        }
        savedialog.Dispose();
        savedialog = null;
    }
```

Ve funkci nejprve vytvoříme proměnné pro ukládání hodnot vyčtených z paměti multimetru, tři proměnné typu `List<string>`. Dvě proměnné typu `int`, jedna pro uložení počtu paměťových bloků a druhou použijeme k testování, zda je přečtená hodnota celé číslo.

První `for` cyklus vykoná tolik cyklů, kolik je v paměti multimetru paměťových bloků. V těle cyklu posíláme dotaz na hlavičku bloku, uložíme ji do řetězce, rozdělíme ji na základě formátovacího znaku `'\r'` na jednotlivé řádky a vložíme je do proměnné typu `List<string>`. Dostáváme se k druhému cyklu `for`, který běží v nekonečné smyčce a v každém cyklu posíláme do multimetru příkaz, pro čtení dat z paměťového bloku. Cyklus opakujeme tak dlouho, dokud z multimetru nedostaneme prázdnou zprávu. Odpovědi z multimetru na dotaz o data jsou posílány v řetězcích ve tvaru `"0 0.146561E5"`, kde první číslice je pořadí měření a druhá reprezentuje naměřenou hodnotu v exponenciálním tvaru. Tento řetězec rozdělíme na dva a uložíme je do proměnných `List<string>`. Pořadí měření uložíme do `pos1` a měřenou hodnotu do `pos2`.

Po zpracování celé paměti, sloučíme proměnné `pos1` a `pos2` do proměnné `all` typu `List<string>`, hodnoty ale rozdělíme středníkem, tak abychom dostali následující tvar řetězce `"0; 0.146561E5;"`.

Nakonec ve funkci otevřeme dialogové okno typu `SaveFileDialog` pro nastavení cesty a přidělení jména ukládanému souboru. Soubor je ukládán v typu csv, který je možné otevřít v programu Excel i Matlab. Data jsou do souboru ukládána do dvou sloupců, první sloupec je pořadí měření a ve druhém sloupci jsou naměřené hodnoty.

3.3.6 Ostatní okna

V položce menu „O programu“ nalezneme okno „About“, ve kterém je napsáno pouze pár informací o programu.

Nevyužitá položka „Script“, má v budoucnu sloužit k otevření okna, v kterém by mohla být například předem nadefinovaná nastavení multimetru, nebo možnost řetězit příkazy tak aby multimetr pracoval automatizovaně apod.

Do položky „Help“ bude v budoucnu vložen dokument s podrobným návodem k aplikaci.

3.3.7 Použité technologie

Kapitola věnována stručnému popisu technologií, s jejichž pomocí byla aplikace vyvíjena. Jedná se zejména o programovací jazyk a vývojové prostředí. Objasníme si některé pojmy použité, v tomto dokumentu i v aplikaci samotné.

3.3.7.1 C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, později schválený standardizačními komisemi ECMA a ISO. Microsoft založil C# na jazycích C++ a Java (a je tedy nepřímým potomkem jazyka C, ze kterého čerpá syntaxi).

C# je jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk. Jazyk a jeho implementace poskytuje podporu pro principy softwarového inženýrství, jako jsou: hlídání hranic polí, detekce použití neinicializovaných proměnných a automatický garbage collector. Důležité jsou také jeho vlastnosti jako: robustnost, trvanlivost a programátorská produktivita.
Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích. Přenositelnost zdrojového kódu je velmi důležitá, obzvláště pro ty programátory, kteří jsou obeznámeni s C a C++. C# je navržen pro psaní aplikací jak pro zařízení se sofistikovanými operačními systémy, tak pro zařízení s omezenými možnostmi. Přestože by programy psané v C# neměly plýtvat s přiděleným procesorovým časem a pamětí, nemohou se měřit s aplikacemi psanými v jazyce C nebo jazyce symbolických adres.

V C# neexistuje vícenásobná dědičnost - to znamená, že každá třída může být potomkem pouze jedné třídy. Toto rozhodnutí bylo přijato, aby se předešlo komplikacím a přílišné složitosti, která je spojena s vícenásobnou dědičností. Třída může implementovat libovolný počet rozhraní.

Neexistují žádné globální proměnné a metody. Všechny funkce a metody musí být deklarovány uvnitř tříd. Náhradou za ně jsou statické metody a proměnné veřejných tříd.

V Objektově orientovaném programování se z důvodu dodržení principu zapouzdření často používá vzor, kdy k datovým atributům třídy lze zvenčí přistupovat pouze nepřímo a to pomocí dvou metod get (accessor) a set (mutator). V C# lze místo toho definovat tzv. Property, která zvenčí stále funguje jako datový atribut, ale uvnitř Property si můžeme definovat get a set metody. Výhodou je jednodušší práce s datovým atributem při zachování principu zapouzdření.

C# je typově bezpečnější než C++. Jediné výchozí implicitní konverze jsou takové, které jsou považovány za bezpečné jako rozšiřování Integerů (např. z 32 bitového na 64

bitový) nebo konverze z odvozeného typu na typ rodičovský. Neexistuje implicitní konverze z typu Integer na Boolean, ani mezi výčtovým typem enum a typem Integer.

C# neobsahuje a ani nepotřebuje dopřednou deklaraci - není důležité pořadí deklarace metod.

Jazyk C# je case sensitive - to znamená, že rozlišuje mezi velkými a malými písmeny.

Common Type System je unifikovaný typový systém, používaný všemi jazyky pod .NET Framework, tedy i jazykem C#. Všechny typy, včetně primitivních datových typů jako je Integer, jsou potomky třídy System.Object a dědí od ní i všechny její metody jako například ToString().

První verze jazyka C# 1.0 byla vydaná v roce 2002 společně s .NET Frameworkem 1.0 obsahovala základní podporu objektového programování, ve které vycházela z jazyka C++ a zkušeností s jejich aktualizací v jazyce Java.

C# 2.0 Na další verzi se čekalo do konce roku 2005. Mezi její nové vlastnosti patří: nativní podpora generik vycházející z podpory na úrovni CLI, Částečné a statické třídy, iterátory, anonymní metody pro pohodlnější užívání delegátů (odkazů na metody), nullovatelné hodnotové typy a operátor koalescence.

C# 3.0 Vyšel na konci roku 2007 společně s .NET Frameworkem 3.5 a Visual Studiem 2008. Obsahuje poměrně revoluční změny, které však nevyžadují změnu podkladového IL, takže aplikace v něm psané půjdou spouštět i na počítačích vybavených, druhým Frameworkem, ponesou-li si s sebou patřičné knihovny.

C# 4.0 Tato verze vyšla v dubnu 2010. Nová verze se zaměřuje hlavně na spolupráci s dynamickými aspekty programování a Frameworky. [5]

3.3.7.2 .NET Framework

.NET je zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu, která je dostupná pro Web, Windows i Pocket PC. Základní komponentou je Microsoft .NET Framevork, prostředí potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak potřebné knihovny. Pro vývoj .NET aplikací vydal Microsoft Visual Studio .NET.

Platforma .NET nepředepisuje použití žádného programovacího jazyka. Bez ohledu na to, v čem byla aplikace původně napsána, se vždy přeloží do mezijazyka Common Intermediate Language.

Nejpoužívanější programovací jazyky pro vývoj .NET aplikací jsou C#, Visual Basic .NET a Delphi.

Součásti .NET Framevorku:

- ASP.NET – technologie pro vývoj webových aplikací
- Windows Communications Foundation (WCF) – technologie pro vývoj webových služeb a komunikační infrastruktury aplikací

- Windows Workflow Foundation (WF) – technologie pro definování heterogenních sekvenčních procesů
- Windows Presentation Foundation (WPF) – technologie pro vytváření vizuálně působivého grafického uživatelského rozhraní pro aplikace
- Windows CardSpace – implementace standardu Information Cards
- LINQ – Language Integrated Query, objektový přístup k datům v databázi, XML a objektech, které implementují rozhraní IEnumerable

Verze:

- 1.0 – rok 2002, vývojové prostředí Visual Studio .net, uveden jazyk C# 1.0
- 1.1 – rok 2003, vývojové prostředí Visual Studio 2003
- 2.0 – rok 2005, nové verze jazyků C# 2.0 a VB.NET 8.0, vývojové prostředí Visual Studio 2005
- 3.0 – rok 2007, vývojové prostředí Visual Studio 2005 nebo 2008
- 3.5 – rok 2007, nové verze jazyků C# 3.0 a VB.NET 9.0, vývojové prostředí Visual Studio 2008
- 4.0 – rok 2010, nové verze jazyků C# 4.0 a VB.NET 10.0, vývojové prostředí Visual Studio 2010

Historicky byla vždy konkrétní verze Visual Studia spjata s konkrétní verzí .NET Frameworku. Změna nastala s verzí 3.0 a zejména s verzí Visual Studio 2008, která jako první podporuje multitargeting – možnost psát aplikace pro .NET Framework verzí 2.0, 3.0 i 3.5, vzhledem ke společnému jádru. Nejnovější verze Visual Studio 2010 taktéž podporuje multitargeting. [6]

3.3.7.3 Microsoft Visual Studio

Je vývojové prostředí (IDE) od Microsoftu. Může být využito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s aplikacemi Windows Forms, webovými stránkami, webovými aplikacemi a webovými službami, jak ve strojovém kódu, tak v řízeném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight.

Visual Studio obsahuje editor kódu podporující InteliSense a refaktorování. Integrovaný debuger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu aplikací s GUI, designer webu, tříd a databázových schémat. Je možné přidávat rozšíření, což vylepšuje funkčnost na téměř každé úrovni.

Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++ (použitím Visual C++), VB.NET (použitím Visual Basic .NET) a C# (použitím Visual C#).

Visual Studio nepodporuje žádný programovací jazyk nebo nástroj samo o sobě. Místo toho je mu možno přidat různá rozšíření funkčnosti. Každá funkčnost je zabalena

do balíčku VSPackage. Když je nainstalována, je dostupná jako služba. IDE poskytuje tři služby: SVsSolution, která umožňuje očíslovat projekty a sestavy; SVsUIShell, který poskytuje rozdělování na okna a UI funkce (jako panely, nástrojové lišty a okna nástrojů); a SVsShell, který se stará o registraci balíčků VSPackage. IDE je také odpovědné za koordinaci služeb a umožnění komunikace mezi nimi. Všechny editory, designery, typy projektů a další nástroje jsou implementovány jako balíčky VSPackage

Prvky:

- Editor kódu - Visual Studio, jako každé jiné IDE, obsahuje editor kódu, který podporuje zvýraznění syntaxe a automatické dokončování za použití IntelliSense nejen pro proměnné, funkce a metody, ale také konstrukce jako cykly a dotazy. Návrhy automatického dokončování se zobrazí ve vyskakovacím seznamu.
- Debugger - Visual Studio obsahuje debugger, který pracuje jak se spravovaným kódem, tak se strojovým kódem a může být použit pro debugování aplikací psaných v jakémkoliv jazyce podporovaném Visual Studiem. Debugger povoluje nastavování breakpointů (které umožňují zastavit běh programu na určité pozici) a watche (které sledují hodnoty proměnných během procesu). Kód lze krokovat, tedy nechat provádět kód po jednom řádku. Může také vstoupit do funkcí, aby je debugoval uvnitř, nebo je přejít. Debugger podporuje funkci Edit and Continue, takže je možné kód upravovat během debuggingu.
- Designer - Visual Studio obsahuje vizuální designery, které pomáhají s vývojem aplikací. Některé z designerů:

  WinForms Designer - WinForms designer je používán pro GUI aplikace za použití WinForms. Obsahuje paletu ovládacích prvků (včetně tlačítek, progress barů, popisek a jiných prvků), které mohou být uchopeny a umístěny na povrch formuláře. Rozložení je možné ovládat ukládáním prvků do kontejnerů nebo uzamykáním na stranu formuláře. Prvky, které zobrazují data (textové pole, rozbalovací pole, tabulka atd.) mohou být propojeny s datovými zdroji jako databáze nebo dotazy.

  WPF Designer - WPF designer, zvaný Cider, byl představen ve Visual Studiu 2008. Stejně jako WinForms designer podporuje drag-and-drop. Vytváří se s ním uživatelské rozhraní pro Windows Presentation Foundation.

  Web designer - Visual Studio také obsahuje editor [website|webových stránek] a designer, který je umožňuje vytvářet uchopováním a pokládáním prvků. Je používán pro vývoj aplikací ASP.NET a podporuje HTML, CSS a JavaScript.

Historie verzí, Microsoft poprvé vydal Visual Studio v roce 1997, když spojil mnoho svých programovacích nástrojů dohromady. Visual Studio 97 bylo vydáno ve dvou edicích, Professional a Enterprise.

Další verze, 6.0, byla vydána v červnu 1998. Tato verze byla základem vývojového systému Microsoftu na příští čtyři roky, protože Microsoft svou vývojářskou pozornost zaměřil na .NET Framework.

Microsoft vydal Visual Studio .NET, zvané Rainier, v únoru 2002. Největší změnou bylo představení vývojového prostředí pro spravovaný kód za použití .NET Framework. Microsoft představil C# (C-sharp), nový programovací jazyk, který je cílen na .NET.

V dubnu 2003 představil Microsoft menší vylepšení Visual Studia .NET zvané Visual Studio .NET 2003 s krycím názvem Everett. Obsahuje aktualizaci .NET Framework, verze 1.1, a je to první verze, které podporuje vývoj aplikací pro mobilní zařízení, buď za použití ASP.NET nebo .NET Compact Framework.

Visual Studio 2005, zvané Whidbey, bylo vydáno v říjnu 2005. Microsoft odstranil z Visual Studia 2005 příponu ".NET" (stejně jako ze všech ostatních produktů s .NET v názvu), ale přesto je stále zaměřeno hlavně na .NET Framework, který byl aktualizován na 2.0.

Visual Studio 2008, zvané Orcas, bylo vydáno pro odběratele MSDN 19. listopadu 2007 vedle .NET Frameworku 3.5. Visual Studio 2008 je zaměřeno na vývoj pro Windows Vista, Office 2007 a webových aplikací.

Visual Studio 2010, zvané "Hawaii", vydala společnost Microsoft 12. Dubna 2010. Visual Studio 2010 přichází s .NET Framework 4 a podporuje vývoj aplikací zaměřených na Windows 7. Visual Studio 2010 obsahuje nástroje pro ladění paralelních aplikací. Nové nástroje umožňují vizualizaci paralelních úloh a jejich běhu. [7]

4 TESTOVÁNÍ

Součástí vývoje každého software, jsou i chyby a nedostatky, které se postupem času odstraňují. Některé jsou odhaleny hned při vývoji, jiné při testování a další při samotném používání aplikace.

Ani tato aplikace nebyla výjimkou a při vývoji byla odstraněna řada chyb. Jako například: nesprávné povolení, respektive nepovolení ovládání prvků oken.
Bylo vyřešeno několik problémů s komunikací po sériové lince (ztráta znaků, překrývání se zpráv). Odstranění „zamrzání“ aplikace, vlivem ztráty spojení atd.
Nicméně zde bude ještě spousta dalších nedostatků, které bude nutné ladit až při použití aplikace v praxi. A postupem času vyladit aplikaci k bezproblémovému chodu a přizpůsobit ji potřebám uživatele.

Na závěr kapitoly zde uvedeme demonstrační příklad měření teploty a názorné zpracování výsledku. Bohužel nebyl k dispozici senzor pro měření teploty, tak bylo využito jen odporu připojeného ke vstupům multimetru, pro názornost je tento příklad dostačující.

V bodech si popíšeme postup k získání měřených hodnot:

- Po spuštění aplikace, otevřeme „COMsetup“, nastavíme parametry portu a připojíme.
- Zkontrolujeme nastavení v „Setup“, především čas a datum, hysterezi, jednotky teploty, případně další možná nastavení
- Můžeme zkontrolovat obsah a zaplnění paměti v okně „Data managment“, v případě potřeby paměť vymažeme
- Samotné měření začneme otevřením okna „Měření“, v roletových menu vybere „Měření teploty“, 1.prametr volíme „Pt 1000“. Volbu potvrdíme tlačítkem „Zvolit“. Vzorkovací frekvenci nastavíme např. na hodnotu 0,1 s a spustíme „Store“. Po doměření všech hodnot, zastavíme „Store“ a uzavřeme okno.
- Export naměřených hodnot provedeme v okně „Data managment“ stiskem „Export paměti do souboru“.
- Podrobnější analýzu dat, provedeme v aplikaci Excel.

S: 00:00.1	
H:1	
CLIP: 1:1	
TEMP_R PT1000	
T: 24.05.12 19:19:28,00	
0	-0.569862E+02
2	-0.567677E+02
4	-0.569170E+02
6	-0.570547E+02
16	-0.569372E+02
18	-0.568082E+02
20	-0.567082E+02
22	-0.565748E+02
24	-0.564694E+02
26	-0.563694E+02
28	-0.562624E+02
30	-0.561448E+02

Obrázek 10: Ukázka zobrazení hodnot v programu Excel

Prvních pět řádku je hlavička bloku paměti.

- S – vzorkovací frekvence
- H – hystereze
- CLIP – nastavení poměru proudových kleští
- TEMP_R PT1000 – typ a rozsah měření
- T – datim a čas

Na dalších řádcích je zobrazeno pořadí měření a k němu asociovaná hodnota v exponenciálním tvaru. Pokud první sloupec tabulky vynásobíme hodnotou vzorkovací frekvence, která je uvedena v hlavičce, získáme čas. Můžeme tedy vynést hodnoty do grafu v závislosti na čase. Další zpracování hodnot už je jen na uživateli, může je ponechat v tabulce, či vynést do grafu, vyvodit z měření různé závěry apod.

Naměřených hodnot bylo mnohem více, ale pro ilustraci postačí pouze část tabulky.

5 ZÁVĚR

Zpracování této práce spočívalo z větší části v návrhu a vývoji software. Ačkoli jsem se programováním nikdy dříve nezabýval, zvolil jsem si právě toto téma bakalářské práce jako výzvu. Postupným zpracováváním návrhů a požadavků na software jsem získával zkušenosti a vědomosti, v dané problematice. Po malých krocích, jsem se dokázal propracovat, až ke konečnému řešení, které se o mnoho neliší od zadaných požadavků.

Základem úspěchu projektu je zprostředkování funkčního spojení osobního počítače a multimetru, prostřednictvím sériového rozhraní GMC-USB. Pro tento účel byla v programu implementována třída, umožňující nastavení a připojení se k sériovému portu. Program bezproblémově zvládá obousměrnou datovou komunikaci po sériové lince.

Jedním z požadavků byla, jednoduchost a nenáročnost programu na uživatele, což se podařilo splnit bez větších problémů. Program je implementován v podobě šesti jednoduchých a přehledných oken s logicky rozmístěnými ovládacími a zobrazovacími prvky.

Dalším důležitým a taky splněným požadavkem, bylo, vyčítání kompletní paměti multimetru s naměřenými hodnotami. Program umožňuje hodnoty z paměti vyčíst, vhodně upravit a uložit do souboru, pro pozdější zpracování.

V programu je implementováno okno s rozhraním, které umožňuje téměř kompletní nastavení parametrů a funkcí multimetru. Stejně tak, se zde zobrazuje aktuální stav nastavení. Implementací rozhraní je splněn další z požadavků zadání.

Poslední z možností, kterou program poskytuje, je náhrada ovládacího otočného kolečka multimetru pro výběr měřící funkce a částečná náhrada displeje multimetru. Je-li multimetr připojen k PC a ovládán pomocí software, není na multimetr potřeba vůbec sahat a lze jej téměř plně využívat.

Vývoj programu je ve fázi, kdy jej můžeme využívat k praktickým měřením. Je tedy plně funkční, neprošel však žádným obsáhlým testováním a tak se při užívání mohou vyskytnout nahodilé chyby, které nebyly doposud objeveny. Po odladění by měl být program připraven pro komerční využití.

Další vývoj aplikace by mohl obsahovat rozšíření, které bude umožňovat více připojených multimetrů, možnost ovládání přes Ethernet, sofistikovanější možnosti zpracování dat, čtení hodnot online, vykreslování hodnot do grafu a další vylepšení, která uživateli usnadní a zpříjemní práci.

6 LITERATURA

[1] ŠINDELÁŘ, Marek. Digitální multimetr METRAHit ENERGY. [online]. [cit. 2012-05-23]. Dostupné z: http://www.etm.cz/index.php?option=com_content&view=article&id=852:multimetr-metrahit&catid=39:meraky

[2] METRAHIT ENERGY. [online]. [cit. 2012-05-23]. Dostupné z: http://eshop.gmc.cz/11-metrahit-energy.html

[3] USB X-TRA. [online]. [cit. 2012-05-23]. Dostupné z: http://www.gossenmetrawatt.com/gmc/english/produkte/usbx-tra.htm#

[4] METRAwin10/METRAHit. [online]. [cit. 2012-05-23]. Dostupné z: http://www.gossenmetrawatt.com/gmc/english/produkte/metrawin10metrahit.htm#

[5] C Sharp. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-25]. Dostupné z: http://cs.wikipedia.org/wiki/C_Sharp

[6] .NET. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-25]. Dostupné z: http://cs.wikipedia.org/wiki/.NET_Framework

[7] Microsoft Visual Studio. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-25]. Dostupné z: http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-vsenviron-0

7 SEZNAM PŘÍLOH