PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Software pro řízení solárního systému

2010 Roman Loník

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně.

30. dubna 2010 Roman Loník

Anotace

Cílem této práce bylo vytvořit software pro řízení a monitorování solárního systému, konkrétně solárních kolektorů pro ohřev vody. Projekt vychází z konkrétních potřeb uživatele solárních kolektorů a poskytuje nástroje pro analýzu získaných dat.

Rád bych poděkoval všem za podporu i ve chvílích, kdy dokončení práce bylo velmi nejisté. Děkuji těm, kteří mě svými radami a připomínkami posouvali dále. Zvláštní poděkování patří paní inženýrce Monice Kochaníčkové, na kterou jsem se častokrát obracel s nestandardními dotazy. V neposlední řadě děkuji panu inženýru Jiřímu Hronkovi, který byl ochotný převzít vedení bakalářské práce.

Obsah

1. Úvod **1**
1.1. Motivace 1
1.2. Cíle práce 1

2. Zadání bakalářské práce **2**
2.1. Specifikace zadání 2
2.2. Požadavky na program 2

3. Problematika solárních systémů **3**
3.1. Úvod do problematiky 3
3.2. Hlavní prvky solární soustavy a základní rozdělení 3
3.3. Konkrétní zadání a řešení 5

4. Implementace **6**
4.1. Použité technologie a programy 6
4.2. Návrh architektury aplikace 6
4.3. Použité konvence 9
4.4. Popis jednotlivých modulů 11
4.4.1. HlavniKnihovna - modul poskytující obecně použitelné metody 11
4.4.2. Databaze - modul pro přímou komunikaci s databází . . . 12
4.4.3. Nastaveni - modul zajišťující veškeré nastavení 14
4.4.4. AnalyzaDat - modul provádějící operace nad získanými daty 16
4.4.5. ManazerGrafu - modul pro práci s grafy 17
4.4.6. ImportExport - modul zajišťující import a export dat . . . 18
4.4.7. Logovani - modul pro zaznamenávání výjimečných stavů . 20
4.4.8. Jadro - modul pro správu ostatních modulů 21
4.4.9. Porty - modul pro práci s paralelním portem 21
4.4.10. GUI - modul grafického uživatelského rozhraní 22
4.4.11. Vypocty - modul pro provádění výpočtů 24
4.4.12. Generator - modul pro generování dat 24
4.5. Paralelní port a knihovna inpout32.dll 24
4.5.1. Paralelní port 24
4.5.2. Přístup k I/O portům v systému Windows 26
4.6. Testování aplikace 27

5. Uživatelská dokumentace **29**
5.1. Instalace a požadavky 29
5.2. Hlavní okno aplikace 30
5.3. Manipulace s grafy 33
5.4. Datové operátory 34

5.5. Import dat 37
5.6. Nastavení aplikace 39
5.7. Otázky a odpovědi 40

Závěr **41**

Conclusions **42**

Reference **43**

Seznam obrázků

1. Schéma solární soustavy 4
2. Plochý kapalinový kolektor 5
3. Vakuový trubicový kolektor 5
4. Zjednodušený model ideální architektury 7
5. Reálný model architektury 8
6. Schéma databáze 13
7. Zapojení teplotních čidel 26
8. Chybové hlášení při pokusu o přímý přístup k portu 27
9. Úvodní obrazovka aplikace 30
10. Hlavní okno programu 31
11. Hlavní okno programu - záložka Přehled 32
12. Zobrazení stavu počasí v systémové liště 32
13. Ovladač grafu 33
14. Datový operátor 35
15. Zobrazení výsledku operace s daty 36
16. Dotaz na zobrazení složky s exportovaným souborem 37
17. Výběr souboru k importu 37
18. Průvodce importem excelového souboru 38

Seznam tabulek

1. Příklad předpon pro pojmenování některých kontrolních prvků . . 10

1. Úvod

1.1. Motivace

Spotřeba elektrické energie jak celosvětově, tak v České republice neustále narůstá. Když si uvědomíme hlavní faktory tohoto růstu, zjistíme, že v budoucnosti nemůžeme čekat pokles spotřeby. Mezi tyto faktory patří zvyšování životního standardu (potřeba vytápění větších bytů, bazénů apod.), vyšší hygienické a zdravotní standardy (díky přísnějším normám ve zdravotnictví spotřebují nové operační sály o 35 - 40 % více el. energie), zvyšující se doprava (rozvoj ekologické hromadné dopravy), rozvoj IT technologií (rozšiřování sítí, např. nahrazení mechanických pokladen digitálními) a další.

V dnešní době je vidět snaha o zvyšování podílu využití alternativních zdrojů energie, ke kterým bezesporu patří i využití solární energie. Zatím sice nejsou solární technologie plně konkurence schopné díky faktorům, jako je dlouhá doba návratnosti, nutný záložní zdroj energie, nestálý sluneční svit apod., ale i přesto najdeme stále více rodinných domů s instalovanými solárními panely, které jsou součástí celých solárních soustav.

Na povrch Země dopadá sluneční záření o energii 180 tisíc terawattů, což je dva tisíce krát více, než potřebuje celá biosféra a čtrnáct tisíc krát více, než spotřebovává celé lidstvo (Celková spotřeba energie je dnes asi 13 TW). Tato energie je navíc nevyčerpatelná (tedy pokud si vystačíme se zásobami na 7 miliard let). Průměrná denní spotřeba energie na jednoho člověka je přibližně 2 kW [1].

Na jeden metr čtverečný (položený vodorovně) u nás dopadne za jeden rok zhruba tisíc kilowatthodin slunečního záření. Solární energie se zde využívá především k ohřevu TUV, ohřívání vody v bazénech a přitápění. I tak lze však radikálně snížit náklady na energie. K tomu je potřeba solární soustavu správně nastavit a mít vhodný nástroj k jejímu řízení a sledování. Tímto nástrojem by mohla být dále popisovaná aplikace Solar Monitoring.

1.2. Cíle práce

Aplikace by měla poskytnout přehled o činnosti solární soustavy a následně zpracovávat získaná data. Základem je zpracování údajů z teplotních čidel, které se ukládají do databáze, jsou předkládány uživateli ve formě grafů a následně mohou být exportovány do několika běžných formátů. V době nepřítomnosti uživatele jej aplikace může upozornit formou e-mailu na různé situace, jako je překročení bezpečných teplot a podobně. Předpokládá se, že aplikace bude pokud možno v nepřetržitém provozu řádově měsíce až roky. To klade nároky na bezpečné ukládání velkého množství dat.

2. Zadání bakalářské práce

2.1. Specifikace zadání

Vytvořit softwarovou aplikaci umožňující sledovat provoz solárního systému včetně regulace základních prvků. Systém má průběžně monitorovat teplotu vody v kolektoru a v zásobníku a monitorovat chod čerpadla. Získaná data zobrazí uživateli v podobě grafů. Aplikace bude umožňovat provádět analýzu získaných dat (výpočty průměrné teploty za určité období, nalezení extrémních teplot, zobrazení teplot v určitém teplotním rozmezí) a exportovat výsledky do xml, textového souboru popřípadě excelového souboru.

2.2. Požadavky na program

- Možnost nastavení parametrů solárního systému (průtok čerpadla, typ izolace zásobníku, tloušťka izolace, měrná tepelná vodivost izolace, hustota a měrná tepelná kapacita cirkulující kapaliny)
- Možnost nastavení četnosti snímání teplot z čidel
- Možnost nastavení rozdílu teplot pro sepnutí a vypnutí čerpadla
- Načítání dat pomocí LPT/import ze souboru
- Grafické znázornění sledovaných veličin
- Analýza získaných dat (selekce dat podle kritérií, výpočty extrémů, lokálních extrémů, průměru atd.)
- Help s popisem všech funkcí aplikace
- Programátorská dokumentace
- Uživatelská dokumentace

3. Problematika solárních systémů

3.1. Úvod do problematiky

Cílem solárních systémů je přeměna světelného záření na teplo. Tato přeměna se nazývá fototermální přeměna a může probíhat buď pasivně nebo aktivně. Pasivním systémem může být celá budova (pasivní heliotechnická budova) nebo jen některé z jejích částí. Transport energie se děje pouze přirozenou cestou (energetická fasáda, Trombeho stěna, zimní zahrady apod.). Aktivní solární systémy se skládají z prvků, které lze na budovy montovat dodatečně. Energii transportuje rozvodný systém, který je naplněn vhodným teplonosným médiem.

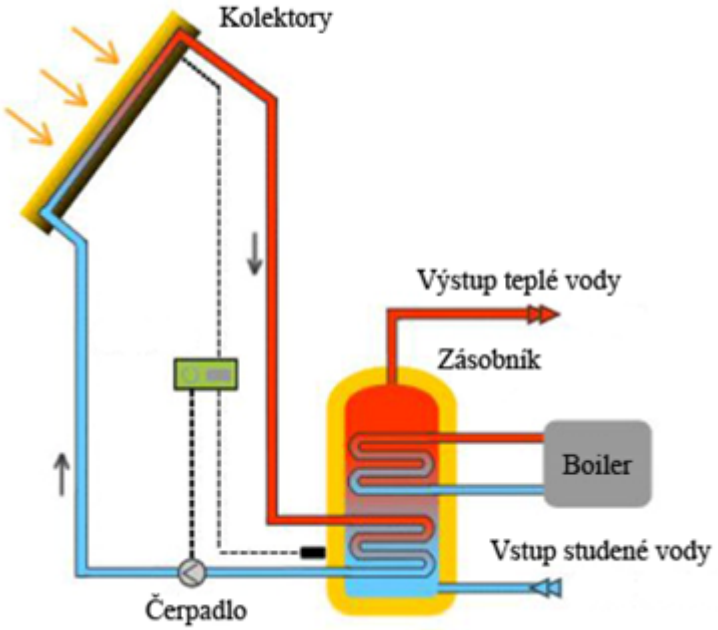
3.2. Hlavní prvky solární soustavy a základní rozdělení

V následujícím seznamu jsou hlavní prvky soulární soustavy:

- Solární absorbér s příslušenstvím - kolektor, nosná konstrukce
- Zásobník teplé užitkové vody
- Průtokové čerpadlo
- Odvzdušňovací ventil
- Manometr
- Pojistný ventil
- Expanzní nádoba
- Regulátor

Aktivní solární soustavy můžeme rozdělit podle několika kritérií. Jedním z nich je i způsob provozu solární soustavy z hlediska teplonosné látky:

- Standardní soustava - Jedná se o soustavu s vysokým průtokem, která se používá pro celoroční přípravu TUV, přitápění objektu popř. ohřev bazénové vody. Nevýhodou je pomalý ohřev zásobníku.
- Low-flow soustava - Její použití je shodné s předchozím typem. Výhodou je vyšší výstupní teplota z kolektorů. Nevýhodou je nižší účinnost kolektorů.
- Drain-back soustava - Soustava s opakovaným vyprazdňováním kolektorů. Používá se pro sezónní přípravu TUV. Nevýhodou je potřeba použití speciálních kolektorů.



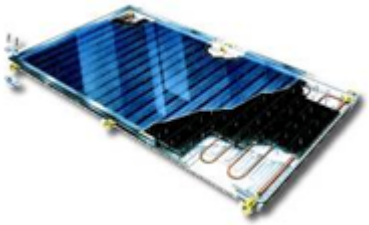
Obrázek 1. Schéma solární soustavy

Pokud bude dále v textu zmíněna soulární soustava nebo solární systém, předpokládá se, že jde o standardní solární soustavu.

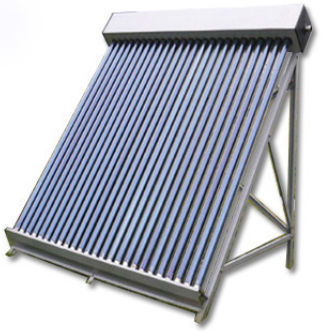
Nejdůležitějsím prvkem solární soustavy a také nejvíce namáhaným jsou kolektory. Ty musí plnit dvě důležité funkce. Absorbovat co nejvíce slunečního záření, které na něj dopadne (absorpce u nejlepších kolektorů až 95%), a efektivně předat získané teplo teplonosné látce při co nejmenších ztrátách. I kolektory můžeme rozdělit do více skupin. Zde je uveden jen stručný přehled:

- Ploché kapalinové kolektory - Nízké „vany“ uzavřené solárním sklem. Na dně vany je izolace (minerální plsť), nad níž je umístěn absorbér s nalisovanými trubkami s teplovodným médiem (voda, nemrznoucí směs). Trubky mohou být uspořádány jako meandr nebo jako lyra.
- Trubicové kolektory
- Vakuové trubicové kolektory - Absorbér je umístěný ve vakuované trubici. Výhodou je minimalizace tepelných ztrát. Absorbér může být plochý s koaxiální trubicí nebo vložená trubka s namontovanou vnitřní trubicí.
- Ploché vakuové

- Koncentrační kolektory - Využívají známého efektu dutých zrcadel k soustředění slunečního záření na menší absorpční plochu a to nejlépe na vakuovou trubici. Zachytávají i difúzní složky záření.



Obrázek 2. Plochý kapalinový kolektor



Obrázek 3. Vakuový trubicový kolektor

Pokud bude dále v textu zmíněn kolektor, předpokládá se, že jde o plochý kapalinový kolektor.

3.3. Konkrétní zadání a řešení

Solární soustava, pro kterou je aplikace vyvíjena, se skládá ze dvou okruhů. První je uzavřený okruh naplněný teplonosnou kapalinou (v tomto případě voda), která potrubím koluje dolů do zásobníku. Při tomto přenosu dochází k tepelným ztrátám mezi kapalinou a stěnami potrubí. V zásobníku dojde k ohřevu vody a teplonosná kapalina putuje zpět do kolektoru, kde bude opět ohřáta. Druhý okruh je otevřený okruh naplněný vodou, která je přiváděna z vodovodní sítě do zásobníku, kde se vytváří zásoba vody. Tato voda se po ohřátí využívá v domácnosti jako teplá užitková voda. Při nepříznivých podmínkách je teplá voda přihřívána záložním boilerem.

K monitorování teplot slouží pětice čidel:

- výstup z kolektoru
- vstup do zásobníku
- výstup ze zásobníku
- uvnitř zásobníku
- čidlo teploty vzduchu v místnosti se zásobníkem

Řízeným prvkem je spínač čerpadla, který se může nacházet ve stavu **zapnuto** nebo **vypnuto**. Pro potřeby prezentace této práce budou zapojena dvě čidla reálně a ostatní budou simulována načítáním dat ze souboru nebo generováním pseudonáhodných teplot.

4. Implementace

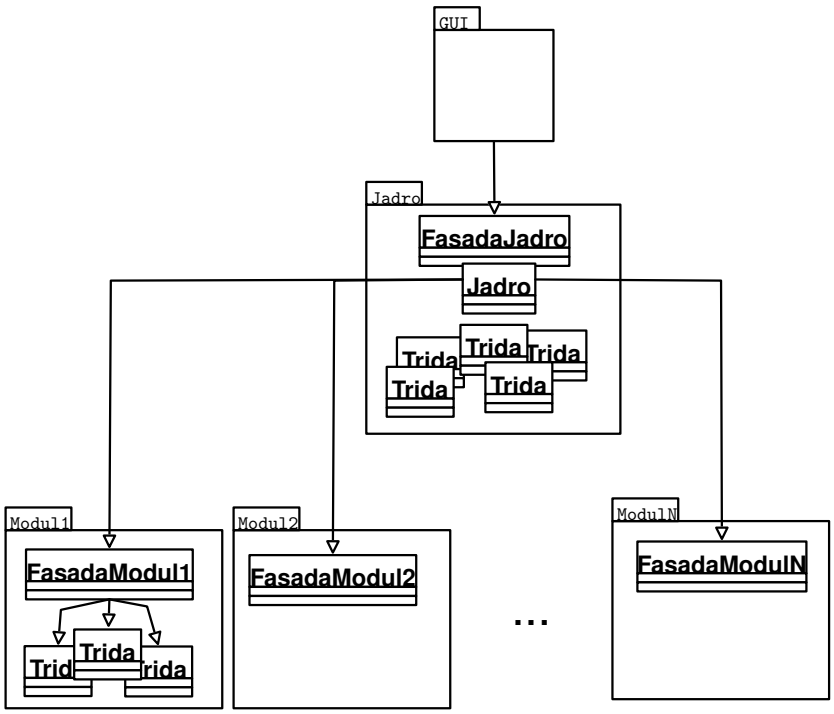
4.1. Použité technologie a programy

Původně byla aplikace implementována v prostředi .NET Framework 2.0. Postupem času se přešlo přes Framework 3.0 k verzi 3.5. Jako hlavní programovací jazyk byl zvolen C# 3.0. Protože aplikace vyžaduje perzistentní uložení velkého množství dat, bylo nutné myslet na nějaký databázový systém. Pro jednoduchost použití bez nutnosti složitého nastavování byla využita databáze Microsoft Access, která měla být později nahrazena. Uvažovalo se o databázovém systému MySql nebo Microsoft Sql Server. Nicméně databáze Microsoft Access (dále jen databáze) se ukázala jako dostačující i pro objemy dat, které aplikace potřebuje uchovávat, a tak je použita i ve finální verzi. K databázi přistupujeme přes rozhraní OLE DB a zprostředkovatelem databáze je Microsoft JET 4.0 OLE DB Provider. Prezentační vrstva aplikace je vytvořena pomocí Windows Forms, což je grafické rozhraní, které je součástí .NET Frameworku. Jako hlavní vývojové prostředí bylo použito Microsoft Visual Studio 2008.

4.2. Návrh architektury aplikace

Na počátku byla snaha o navržení modulárního systému, ve kterým by bylo minimalizováno množství vazeb mezi jednotlivými moduly, a každý modul by měl co nejomezenější informace o svém okolí a sobě samém. Takový systém by se velmi snadno rozšiřoval a změna v jednom modulu by neovlivnila zbytek systému. V praxi se tento model nepodařilo zcela úplně dodržet, ale architektura nese několik společných rysů.

Celá aplikace je rozdělena do dvanácti modulů (třinácti, pokud počítáme projekt instalátoru aplikace za samostatný modul), kde každý modul řešíme pomocí návrhového vzoru fasáda (*facade*). Tento vzor velmi usnadňoval práci v pokročilém stadiu projektu, protože jsme nemuseli pracovat s obrovským počtem tříd a u každého projektu jsme se soustředili jen na fasádu. Tím je také skryta samotná implementace a můžeme se soustředit na logickou část projektu. Další návrhový vzor využitý v architektuře je jedináček (*singleton*). Je použit pro jádra některých modulů (viz kapitola 4.3.). Pokud je modul A využíván jinými moduly, pak každý z těchto modulů vytváří vlastní objekt fasády modulu A, ale všechny fasády se odkazují na tutéž instanci jádra modulu A.

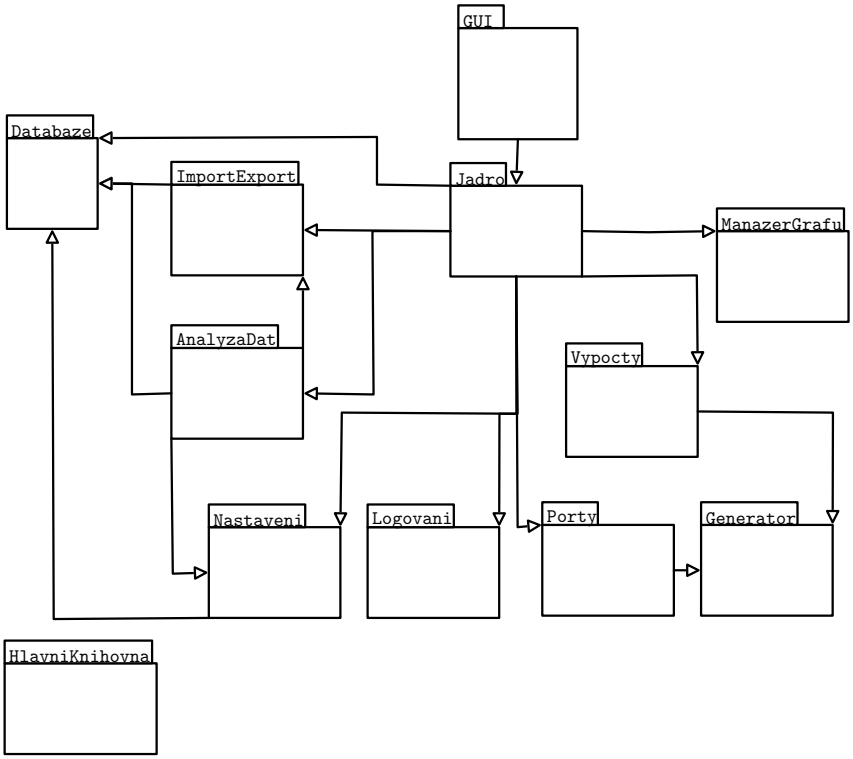


Obrázek 4. Zjednodušený model ideální architektury

Nejprve popíšeme ideální architekturu, ke které jsme se snažili přiblížit. Aplikaci můžeme rozdělit na několik vrstev. Nejvýše v hiearchii by se nacházela prezentační vrstva, která neprovádí žádné vlastní výpočty, a umožňuje komunikaci s uživatelem. Tato vrstva může komunikovat pouze s vrstvou nižší, kterou představuje správce modulů a ostatní moduly. Správce modulů by sloužil jako mezičlánek mezi prezentační vrstou a ostatními moduly, a zajišťoval by vzájemnou komunikaci modulů. Tyto moduly pod správcem by nevěděly nic o svém okolí (o žádném jiném modulu). Pokud by potřebovaly jakákoliv data z jiných modulů, vyslaly by požadavek a čekaly by na jeho vyřízení. Tento požadavek by zpracoval správce modulů, který by volajícímu modulu dodal požadovaná data. Správce modulů by byl tak jediný, kdo by měl informaci o všech ostatních modulech pod sebou. Prezentační vrstva by přímo komunikovala pouze se správcem. Další vrstvou by byla datová vrstva, kterou by představovala databáze a modul pro přímou komunikaci s paralelním portem. Navenek by každý modul zpřístupňoval pouze třídu fasády.

U výsledné architektury se nepodařilo moduly zcela odstínit a proto je míra provázání vyšší. Stále však můžeme celou architekturu abstraktně rozdělit do vrstev. Nejvýše opět leží prezentační vrstva zastoupená modulem `GUI`. Tento modul komunikuje se správcem (toho představuje modul `Jadro`), který umožňuje přístup k dalším modulům. Zcela speciální postavení má modul `HlavniKnihovna`,

který dodává jednoduché obecné statické metody používáné skrz všechny vrstvy (a ve všech modulech) v celé aplikaci. Pro zjednodušení diagramu reálné architektury na obrázku 5. jsou vynechány všechny vazby na hlavní knihovnu. Komunikace mezi moduly je implementováná pomocí událostí (v případě komunikace z nižší vrstvy do vyšší) a přímého volání metod z fasády (v případě komunikace z vyšší vrstvy do nižší). Každý modul bude samostatně popsán níže.



Obrázek 5. Reálný model architektury

Na obrázku 5. představuje každá šipka směr přímé komunikace mezi moduly. S modulem `Databaze` tak přímo komunikují moduly `Jadro`, `Nastaveni`, `ImportExport` a `AnalyzaDat`. Ideální architektury by šlo v tomto případě dosáhnout tak, že by moduly `Nastaveni`, `ImportExport` a `AnalyzaDat` při potřebě dat z modulu `Databaze` vyslaly požadavek (komunikace směrem nahoru, tedy pomocí události), na který by reagoval modul `Jadro`, který by měl jako jediný přímý přístup k modulu `Databaze`, a získaná data by přímo předal modulu, který vyslal požadavek. S rostoucím počtem modulů však tento přístup příliš zvyšoval režii a bylo rozhodnuto, že klíčové moduly mohou komunikovat přímo, ale vždy jen přes své fasády. Tím se zamezilo ještě užšímu provázání. Takto vznikl kompromis, který udržoval míru složitosti v rozumných mezích.

4.3. Použité konvence

Finální aplikace obsahuje necelých sto tisíc řádků kódu, z toho zhruba polovina představuje automaticky vygenerovaný kód k formulářům. Komentáře představují pět procent celkového kódu, dokumentace zhruba dvacetpět procent kódu. Celkové množství tříd dosáhlo počtu dvěstěpět. Kvůli velkému rozsahu projektu byly použity následující konvence, které čerpají z [3]:

- Modul (project) - Názvy modulů jsou jednoslovné, maximálně dvouslovné, stručné. Každé slovo začíná velkým písmenem (např. `HlavniKnihovna`, `Logovani`)
- Třída (class) - Názvy tříd začínají na písmeno `c`. Názvy mohou být víceslovné, ale vždy se snažíme počet slov minimalizovat se současným zachováním srozumitelnosti. Každé slovo začíná velkým písmenem. Pro pojmenování tříd jsou použita podstatná jména (např. `cFiltraceDat`, `cOperaceNadDb`).
- Metoda (method) - Názvy metod začínají velkým písmenem a důraz je kladen na srozumitelnost. Cílem bylo obsáhnout v názvu metody její funkci. Proto jsou názvy metod někdy delší, ale získáme tím jednodušší orientaci v kódu, protože nemusíme studovat implementaci metody (např.`VytvorGraf`, `PridejKrivceData`, `VratSchemaTabulky`).
- Struktura (struct) - Struktury začínají na písmeno `s`. Názvy mohou být víceslovné (např. `sMezeFiltrovani`, `sRamecDat`, `sDataGrafu`).
- Výčtový typ (enum) - Výčtové typy začínají na předponu `enm` a mohou být víceslovné. Každé slovo začíná velkým písmenem (např. `enmOperaceSGrafy`, `enmCidla`, `enmTypChyby`).
- Delegát (delegate) - Delegáty, neboli reference na metody, začínají na písmeno `d`. Názvy mohou být víceslovné (např. `dOdectenaNovaData`, `dZmenaStavuSpinace`).
- Událost (event) - Události začínají na písmeno `u` a mohou být víceslovné (např. `uChybaCteniDat`, `uKontrolaTeploty`).
- Grafický uživatelský prvek (UserControl) - Grafické uživatelské prvky začínají předponou `UC` a mohou být víceslovné (např. `UCZasobnik`, `UCInfoPrumer`).

Dalším způsobem, jak udržet kód čitelný, byla konvence pojmenování proměnných. Kontrolní prvky (Controls) a některé další prvky (např. seznamy) obsahují předponu, která je tvořena zkrácením názvu typu daného prvku a vynecháním samohlásek. Názvy mohou být víceslovné a každé slovo začíná

velkým písmenem. Pokud se jedná o proměnnou některého základního typu (*int*, *bool*, *double*, *string* apod.), předponu nepíšeme. Příklad některých předpon můžeme vidět v tabulce 1.

datový typ	předpona
System.Collections.Generic.List	lst
System.Windows.Forms.Button	btn
System.Windows.Forms.CheckBox	chb
System.Windows.Forms.ComboBox	cb
System.Windows.Forms.Label	lb
System.Windows.Forms.ListBox	lstb
System.Windows.Forms.PictureBox	pb
System.Windows.Forms.MenuStrip	ms
System.Windows.Forms.RadioButton	rb

Tabulka 1. Příklad předpon pro pojmenování některých kontrolních prvků

Posledním důležitým prvkem, který usnadňuje orientaci v kódu, je dodržení následujících pravidel:

- Nový modul musí být logický celek, s jasně definovaným účelem a přesně vymezenými kompetencemi. Pokud při návrhu není jasné, které třídy do modulu zahrnout, a které již ne, je třeba návrh přebudovat.
- Veřejná (*public*) třída modulu je pouze třída fasáda (konvence pojmenování `cFasada`[`Název modulu`]). Ostatní moduly využívají pouze metody ve fasádě.
- Fasáda neposkytuje žádnou vlastní funkcionalitu, pouze umožňuje přístup k zapouzdřeným metodám, vlastnostem apod.
- Každý modul obsahuje třídu `cJadro`[`Název modulu`], která obsahuje nejdůležitější metody modulu. Pokud se jedná o modul, který přímo komunikuje s jinými moduly, jsou instance fasád těchto modulů vytvářeny právě v jádře.
- Všechny konstanty použité v rámci modulu jsou udržovány v souboru `cKonstanty`. Konstanty jsou dostupné pouze pro daný modul.
- Všechny struktury a výčtové typy modulu jsou udržovány v souboru `cStruktury`. Dostupnost viz předchozí bod.

Dodržení všech pravidel je samozřejmě ideální stav, kterému jsme se chtěli co nejvíce přiblížit.

4.4. Popis jednotlivých modulů

V této kapitole bude následovat popis jednotlivých modulů, některé jejich důležité prvky a vzájemná komunikace.

4.4.1. HlavniKnihovna - modul poskytující obecně použitelné metody

Jak bylo řečeno výše, modul `HlavniKnihovna` má speciální postavení mezi ostatními moduly. Nemá vlastní fasádu a tvoří spíše logický celek uchovávající statické třídy a jejich metody, které jsou všeobecně využitelné v celé aplikaci. Dále jsou v tomto modulu definovány obecné struktury, výčtové typy a definice delegátů, jejichž použití přesahuje rámec jednotlivých modulů.

Mezi nejdůležitější třídy hlavní knihovny patří `cPomocneMetody`, `cPraceSXml`, `cStruktury`, `cDelegati` a poté celá sada tříd, které představují rozšíření `EventArgs` a slouží k uchování informací o argumentech specifických událostí.

Některé struktury z modulu `HlavniKnihovna`:

- `sRamecDat` - Uchovává informace o rozsahu časových dat. Obsahuje horní a spodní mez časového rozsahu, které jsou uložené ve formátu *OLE Automation Date*. Tento formát je implementován jako číslo s pohyblivou desetinnou čárkou, jehož hodnota je počet dní od půlnoci 30. prosince 1899. Tato struktura obsahuje metody pro vrácení rozsahu aktuálního dne, týdne, měsíce, apod..

- `sKrivkaSDaty` - Uchovává informace o typu křivky grafu a jejích datech, která jsou uložena zvlášť jako seznam hodnot osy X a osy Y.

- `sZasobnik` - Uchovává informace o nastavení zásobníku vody. Nastavení je uloženo v databázi a při požadavku aktuálního nastavení je uloženo do této struktury. Ve zbytku aplikace se již pracuje jen se strukturou.

Některé výčtové typy z modulu `HlavniKnihovna`:

- `enmCetnostBodu` - Vyjadřuje četnost zobrazovaných bodů grafu. Body můžeme zobrazovat všechny, nebo jen s četností po jedné hodině, šesti hodinách, dvanácti hodinách, dni, týdnu nebo měsíci.

- `enmTabulkyDb` - Rozlišuje tabulky uložené v databázi. Hodnoty tohoto výčtového typu jsou v modulu `Databaze` převedeny na řetězec s reálným názvem tabulky. Tyto řetězce jsou privátními konstantami modulu pro práci s databází a v celé aplikaci tak pracujeme pouze s hodnotami výčtového typu. Nemůže se tak stát, že bychom použili jinou tabulku, než kterou máme nadefinovanou pomocí výčtového typu.

- `enmTypOperaceData` - Vyjadřuje operace, které je možné provádět s naměřenými daty. Jedná se o selekci, výpočet průměru, maxima, minima, lokálního maxima a lokálního minima.

Některé metody z třídy `cPomocneMetody`:

- `bool JeDostupnyExcel()` - Zjistí, zda je nainstalována aplikace Microsoft Excel. Při absenci Excelu zakáže možnosti pro import a export excelových souborů.
- `RadioButton VratVybranyRadioButton(ControlCollection)` - Ze zadané kolekce kontrolních grafických prvků vrátí *RadioButton*, který je ve stavu *Checked*. Pokud žádný takový prvek v kolekci neexistuje, vrací *null*.
- `bool JeSouborPristupnyProZapis(string)` - Zjistí, zda soubor na zadané cestě existuje a má nastavena práva pro zápis.
- `bool ObsahujeVazbuGrafu(ZedGraphControl, out sVazbyGrafu)` - Testovací metoda, která zjistí, zda zadaný graf obsahuje vazbu grafu, a v kladném případě tuto vazbu vrátí ve výstupním parametru.

Pokud chce některý modul používat metody, struktury, výčtové typy a další prvky hlavní knihovny, musí mít hlavní knihovnu nareferencovánu. Metody pak voláme následujícím způsobem:

```
bool jeExcel = HlavniKnihovna.cPomocneMetody.JeDostupnyExcel();
```

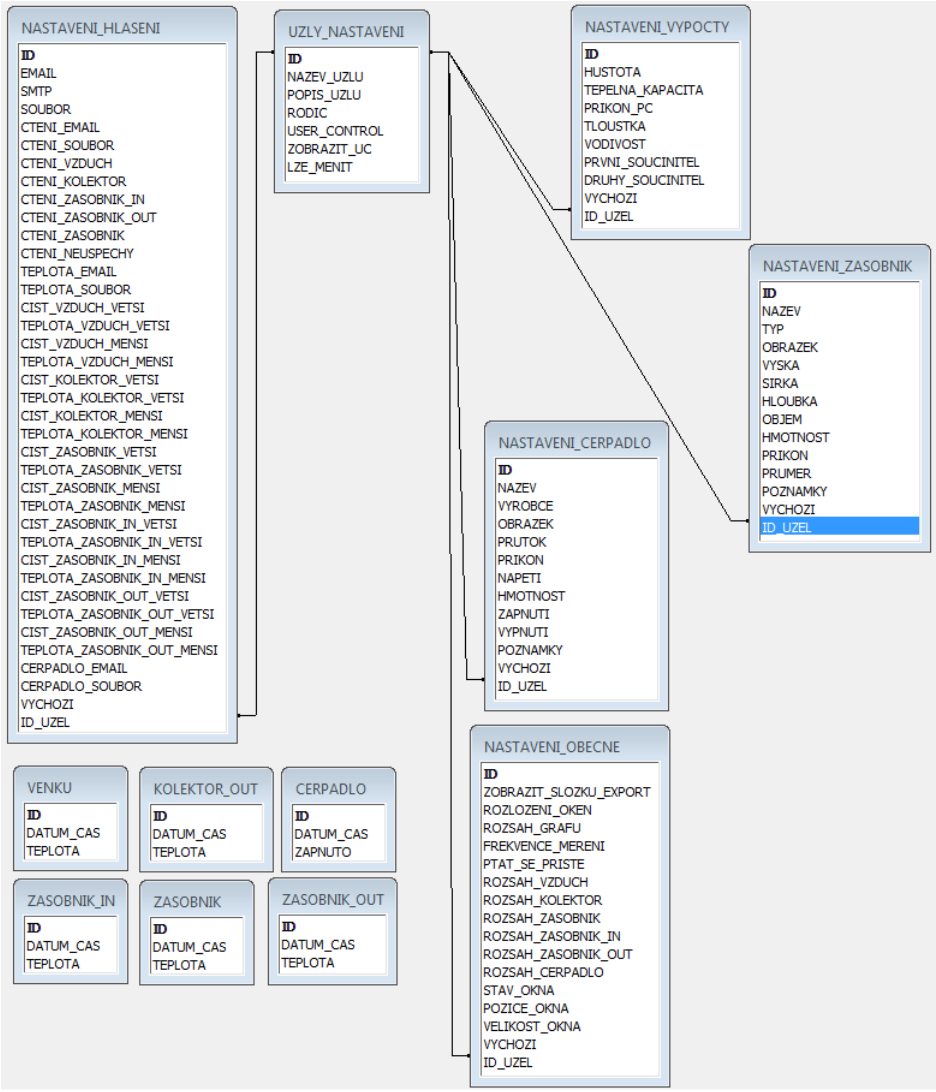
Struktury a výčtové typy používáme následujícím způsobem:

```
HlavniKnihovna.enmTabulkyDb tab =
HlavniKnihovna.enmTabulkyDb.zasobnikIn;
```

4.4.2. Databaze - modul pro přímou komunikaci s databází

Tento modul má jako jediný přímý přístup k databázi. Jak bylo zmíněno výše, aplikace využívá *Microsoft Access Jet Database Engine* k přístupu k databázi vytvořené pomocí aplikace Microsoft Access. Soubor databáze (s příponou *.mdb*) je při instalaci nakopírován do příslušného umístění na pevném disku jako skrytý soubor. Nepředpokládá se jakákoliv ruční manipulace s tímto souborem. Pokud aplikace nenajde soubor na očekávaném místě, informuje uživatele chybovým hlášením a nebude fungovat tak, jak je zamýšleno. Pokud by soubor na daném umístění existoval, ale byla by změněna jeho vnitřní struktura, aplikace se bude chovat neočekávaně.

Databáze má velmi jednoduchou strukturu s minimem relací mezi tabulkami. V tabulkách máme uloženy informace o načtených datech z čidel (jedna tabulka pro každé čidlo) a dále nastavení aplikace a solární soustavy. Schéma databáze můžeme vidět na obrázku 6..



Obrázek 6. Schéma databáze

Tabulky `VENKU`, `KOLEKTOR_OUT`, `ZASOBNIK_IN`, `ZASOBNIK` a `ZASOBNIK_OUT` mají tři sloupce. Primárním klíčem je první sloupec s automaticky generovaným identifikátorem. Následuje sloupec s údajem o datumu a čase měření. Poslední sloupec obsahuje hodnotu naměřené teploty ve stupních Celsia. Tabulka `CERPADLO` je velice podobná předchozím s tím rozdílem, že poslední sloupec je typu *boolean* a uchovává informaci o sepnutí spínače čerpadla. Klasicky nabývá dvou hodnot *true/false*. Tyto tabulky nemají žádné relace.

Tabulka `UZLY_NASTAVENI` uchovává informace o jednotlivých uzlech stromu nastavení. Přes sloupec `ID`, který je primárním klíčem (tak jako ve všech ostatních tabulkách), je provázána s ostatními tabulkami nastavení. Podrobněji se budeme

nastavení věnovat v kapitole 4.4.3..

Jádro modulu `Databaze` (dále v této kapitole jen jádro), třída `cJadroDatabaze`, je navržena podle návrhového vzoru jedináček (*singleton*). Přímo k instanci jádra může přistupovat pouze fasáda modulu, třída `cFasadaDatabaze` (dále v této kapitole jen fasáda). V konstruktoru fasády zkontrolujeme, zda bylo jádro inicializováno, a v záporném případě jej inicializujeme. Inicializací je myšleno nastavení připojovacího řetězce (*connect string*) k databázi.

Další důležité třídy jsou `cFiltraceDat` a `cOperaceNadDb`. První jmenovaná poskytuje metody pro filtrování získaných dat podle kritérií, vytváření kritérií, podle zadaných parametrů apod.. Druhá jmenovaná třída provádí základní operace nad databází, jako je vkládání, úpravy, selekce a odstraňování záznamů. V jádře jsou pak tyto metody skládány do složitějších specifických konstrukcí. Ve třídě s konstantami jsou uloženy reálné názvy tabulek, tak jak jsou nazvány přímo v databázi.

Tento modul je naprogramován tak, aby bylo možné vyměnit zdroj dat s co nejmenším zásahem do existujícího kódu.

4.4.3. Nastaveni - modul zajišťující veškeré nastavení

Modul `Nastaveni` je stejně jako předchozí modul navržen podle návrhového vzoru fasáda a jádro modulu je podle vzoru jedináček. Vnitřní uspořádání je taktéž podobné (třídy `cFasadaNastaveni`, `cJadroNastaveni`, `cKonstanty`). U tohoto modulu došlo k porušení třívrstvé architektury, protože dodává vlastní grafické uživatelské prvky pro zobrazení nastavení a umožňuje uživateli práci s nastavením. Při zpětném hodnocení kódu je toto nesprávný krok, a příště by byly veškeré grafické prvky soustředěny v jednom modulu (v našem případě modul `GUI`).

Pokud některý modul požaduje hodnoty nastavení, vytvoří si objekt fasády, a zavolá metodu, která vrátí strukturu s hodnotami vztahujícími se ke konkrétnímu nastavení. Pokud například požadujeme obecné nastavení aplikace, zavoláme metodu fasády `sObecneNastaveni VratVychoziObecneNastaveni()`. Zmíněný modul `GUI` zobrazuje na požadavek uživatele formulář s hlavním oknem nastavení (`cHlavniOknoNastaveni`), který získá přes fasádu pomocí metody `cHlavniOknoNastaveni VratHlavniOknoNastaveni()`. Jediné nastavení, které chceme ukládat programově, je obecné nastavení aplikace. K tomu slouží metoda `void UlozObecneNastaveni(sObecneNastaveni)`. U zbývajícího nastavení očekáváme, že jej bude ukládat jen uživatel.

Hlavní okno nastavení sestává ze dvou nejdůležitějších částí: stromového zobrazení uzlů nastavení a uživatelských grafických kontrolních prvků (*user controls*), ve kterých je zobrazeno vlastní nastavení. Všechny tyto kontrolní prvky dědí z jednoho předchůdce, kterým je `UCPredek`. Tím získávají následující metody:

- `abstract void UlozNastaveni()` - Provede uložení aktuálně zadaných hodnot do databáze.
- `abstract object VratNastaveni()` - Vrátí hodnoty aktuálně zobrazené v grafickém kontrolním prvku.
- `abstract void VynulovatNastaveni()` - Vymaže hodnoty všech položek v graf. kontrolním prvku.
- `abstract void Reset()` - Nastaví hodnoty v graf. kontrolním prvku na hodnoty, které jsou uloženy v databázi.
- `abstract bool SynchronizujNastaveni(DataRow)` - Pokusí se synchronizovat své nastavení podle zadaného záznamu z databáze.
- `void PridejHandleryUdalosti(params System.Delegate[])` - Přiřadí k příslušným událostem jejich obsluhu.
- `void OdeberHandleryUdalosti(params System.Delegate[])` - Odebere obsluhu příslušným událostem.

U tohoto modulu nastal jeden problém, se kterým si Visual Studio zatím neumí poradit. A tím jsou abstraktní formuláře. Designer Visual Studia neumí zobrazovat formuláře (resp. uživatelské kontrolní prvky), které jsou poděděné z abstraktních formulářů. Při zamyšlení se nad tímto problémem však dojdeme k závěru, že koncepce abstraktních formulářů je špatná. Pokud chceme všem formulářům dodat stejné metody, máme několik jiných možností, jak toho docílit. Jednou z nich je možnost využít rozhraní (*interface*). Při hodnocení kódu dostala implementace abstraktních formulářů záporné body a v budoucnu by tato špatná technika nebyla použita.

V následujícím seznamu je popis jednotlivých sloupců tabulky `UZLY_NASTAVENI`.

- `ID` - Primární klíč. Jedinečný identifikátor.
- `NAZEV_UZLU` - Programový název uzlu ve stromovém zobrazení nastavení.
- `POPIS_UZLU` - Název uzlu, tak jak je zobrazen ve stromovém zobrazení nastavení.
- `RODIC` - Název rodičovského uzlu, pokud je daný uzel zanořen. Odkazuje se na sloupec `NAZEV_UZLU`.
- `USER_CONTROL` - Uživatelský grafický prvek, který se zobrazí po výběru daného uzlu.

- `ZOBRAZIT_UC` - Sloupec, který vyjadřuje příznak, zda chceme daný grafický prvek zobrazovat.
- `LZE_MENIT` - Sloupec, který vyjadřuje příznak, zda může uživatel daný uzel přejmenovat nebo odstranit.

Nastavení jednotlivých uživatelských prvků, které představují např. nastavení solární soustavy (zásobníku vody, čerpadla), nebo obecné nastavení aplikace, je uloženo v samostatných tabulkách (viz schéma databáze na obrázku 6.). Při inicializaci nebo obnovení (refresh) nastavení projdeme nejdříve tabulku uzlů, a pokud daný záznam ve sloupci `USER_CONTROL` obsahuje hodnotu, znamená to, že daný uzel nastavení zobrazuje nějaký grafický prvek. Podle typu grafického prvku vyhledáme tabulku s nastavením a díky relaci mezi sloupci `UZLY_NASTAVENI.ID` a `[Příslušná tabulka].UZEL_ID` vrátíme záznam, který uchovává nastavení daného graf. uživatelského prvku.

4.4.4. AnalyzaDat - modul provádějící operace nad získanými daty

Jak už název modulu napovídá, slouží k práci s načtenými daty. Dokáže počítat průměrné teploty za určité období, zobrazovat data, která splňují zvolené podmínky, hledat maximální a minimální teploty apod.. U tohoto modulu, stejně jako u modulu v kapitole 4.4.3., došlo k porušení třívrstvé architektury a kromě logiky dodává i své vlastní grafické komponenty. Při zpětném hodnocení kódu bylo toto zhodnoceno jako chybný krok. Jádro modulu je řešeno podle návrhového vzoru jedináček a fasáda tohoto modulu je vytvořena pouze modulem pro správu ostatních modulů (modulem `Jadro`). Základním prvkem modulu `AnalyzaDat` je uživatelský grafický prvek `UCDatovyOperator`, který je pro přehlednost rozdělen do více souborů s těmito názvy:

- `UCDatovyOperator.cs` - Zde je definován vzhled graf. prvku a základní metody.
- `UCDatovyOperatorExtrem.cs` - Logika pro výpočty extrémů (maxima, minima).
- `UCDatovyOperatorPrumer.cs` - Logika pro výpočet průměrných teplot.
- `UCDatovyOperatorSelece.cs` - Logika pro selekci dat podle kritérií.

V jádře modulu (`cJadroAnalyzaDat`) máme metodu `UCDatovyOperator VytvorDatovyOperator(string, HlavniKnihovna.enmTypyGrafu)`, která vytvoří datový operátor (dále v této kapitole budeme datové operátory zkráceně nazývat jen operátory) zadaného jména, který se bude vztahovat ke grafu, jehož typ je určen druhým parametrem. Všechny operátory vytváříme při inicializaci jádra ve fasádě a jiný modul nemůže vytvořit další operátory. K tomuto modulu náleží i formulář pro zobrazení výsledku požadované operace.

Protože objekty operátorů, grafů (viz kapitola ??) a ovladačů grafů o sobě navzájem neví, bylo nutné navrhnout jejich vzájemnou komunikaci. Do hry tak vstupují tři moduly: `AnalyzaDat` s datovými operátory, `ManazerGrafu` s objekty grafů (viz kapitola 4.4.5.) a `Jadro` s ovladači grafů (viz kapitola 4.4.8.). Pokud uživatel změní zobrazovaný rozsah dat, se kterými se má pracovat v operátoru, je nutné změnit rozsahy i ve výše zmíněných objektech. V modulu `Jadro` máme metodu pro inicializaci operátorů (`cJadro.InicializujDatoveOperatory()`). V této metodě přiřadíme datovým operátorům obsluhy událostí, mimo jiné i události vyvolané při změně rozsahu. Tato obsluha je v modulu `Jadro` a zde je již dostatek informací pro nastavení rozsahů v ostatních zainteresovaných objektech.

Následuje popis operací nad daty.

- **Selekce** - Selekce dat probíhá na základě kritérií, která zadá uživatel. Prvním kritériem je aktuální zobrazovaný rozsah dat v grafu. Selekce se vztahuje jen k viditelným datům. Další dvě kritéria ovlivňují data na ose teplot. Uživatel může zadat až dvě omezení (větší než , menší než, menší rovno než, větší rovno než, rovno), mezi kterými je vztah „a zároveň“ (AND). To znamená, že vyhovující data musí splňovat zároveň všechna kritéria.

- **Průměr** - Výpočet průměru může probíhat nad všemi aktuálně zobrazenými daty, nebo jen nad daty naměřenými ve vybrané hodiny. Vyhovující data jsou ta, jejichž teplota je rovna vypočítané průměrné teplotě. Volitelně je možné zobrazit data, která se liší od průměrné teploty maximálně o zvolenou hodnotu (například odchylka jednoho stupně Celsia).

- **Extrémy** - Na zvoleném rozsahu najde absolutní maxima či minima. Volitelně lze zvolit lokální extrémy, kde rozsah, na kterém jsou hledány, specifikuje uživatel. Lze zvolit rozsah jedné hodiny, dne, týdne, měsíce a roku.

Výsledek operace je možné importovat do textového souboru, souboru aplikace Microsoft Office Excel, nebo do formátu XML. Více informací k exportu a importu dat v kapitole 4.4.6..

4.4.5. ManazerGrafu - modul pro práci s grafy

Modul zajišťuje veškerou logiku s grafy v aplikaci. Protože v době počátečních prací na aplikaci nebyla v .NET Frameworku žádná komponenta pro zobrazování grafů, bylo nutné se rozhodnout, zda bude vytvořena komponenta vlastní, nebo se využije nějaká již vytvořená a otestovaná uživateli.

Nakonec dostala přednost výborná komponenta ZedGraph (viz el. publikace [4]), která je šířena pod licencí LGPL. Tato licence dovoluje používat knihovny i v aplikacích, které nemají licenci LGPL. Tato aplikace pouze

připojuje projekt Zedgraph a nemění jej, proto se stává „dílem, které používá knihovnu“. Domovská stránka projektu ZedGraph je na adrese http://zedgraph.org/ a úvodní podrobný návod od autora najdeme na adrese http://www.codeproject.com/kb/graphics/zedgraph.aspx.

Komponenta ZedGraph umožňuje vytvářet spojnicové grafy, koláčové, sloupcové horizontální i vertikální, bodové, japonské svíčkové grafy a mnoho dalších. V této aplikaci si pro znázornění naměřených teplot vystačíme se základními spojnicovými grafy.

I když je možné grafy jednoduše přetáhnout v designeru Visual Studia na formuláře, vytváříme si objekty grafů ručně v jádře modulu `cJadroManazerGrafu`, pomocí metody `VytvorGraf`. Tato metoda má celkem dvanáct parametrů, pomocí kterých specifikujeme všechny důležité vlastnosti vytvářeného grafu. Jádro obsahuje množství dalších metod, z nichž jmenujme metody `ZmenHorniMezOsy`, `ZmenDolniMezOsy`, `PridejKrivceData`, `OdstranBodyKrivky`, `VratKrivkuGrafu`, jejichž názvy jsou dost výmluvné.

Další třídou v tomto modulu je třída `cUpravaOsy` pro práci s osami grafu. Zde zmíníme metodu `NastavAVratJednotkyOsyX`. Tato metoda podle aktuálního rozsahu časové osy x nastaví příslušné jednotky. Časovou osu tak lze přiblížit až na rozmezí jednotlivých vteřin a naopak oddálit na rozmezí několika let. Díky této metodě pak budeme vždy zobrazovat jednotky vhodné pro daný rozsah (nepředpokládá se, že by uživatel potřeboval zobrazovat měřená data z většího rozsahu než několik desetiletí.).

Grafy jsou využity pro zobrazení získaných dat z čidel (naměřené teploty a sepnutí čerpadla) a také při zobrazení výsledku operace s daty. Každý graf si udržuje ve vlastnosti `Tag` strukturu `sVazbyGrafu`, ve které jsou uchovány další informace o grafu. Mezi tyto informace patří:

- Typ grafu - Proměnná výčtového typu `enmTypyGrafu`, která jednoznačně určuje, o jaký graf jde (např. graf s daty z čidla ze zásobníku).
- Rozsah načtených dat - Proměnná typu `sRamecDat`, která vymezuje rozsah všech dat uložených v grafu.
- Seznam přihlášených obsluh událostí.
- Seznam obsluh událostí, které jsou dočasně potlačeny.

Pokud potřebujeme v aplikaci pracovat s nějakým grafem, většinou nepracujeme přímo s objektem grafu, ale s jeho typem (viz výčtový typ `enmTypyGrafu`).

4.4.6. ImportExport - modul zajišťující import a export dat

Zde jsou soustředěny metody pro import a export dat z a do několika různých formátů. Předpokládá se, že se naměřená data dostanou do aplikace pouze dvěmi cestami. První cesta je naměřením hodnot na čidlech a druhou je právě import

dat. Uživatel má možnost importovat data ze souboru aplikace Microsoft Excel (ve formátu *xls*) nebo z textového souboru (ve formátu *txt*). Export dat je možný do souboru aplikace Microsoft Office Excel (dále jej v této kapitole budeme označovat jako excelový soubor), do textového souboru a do formátu XML. Zobrazený graf lze uložit jako obrázek v několika formátech (*emf*, *png*, *gif*, *jpg*, *tif*, *bmp*), ale tato funkcionalita nesouvisí s modulem `ImportExport`.

U excelového souboru není uživatel nijak vázán strukturou dat v souboru, a pokud jsou data na jednom listu, nezáleží například na pořadí sloupců či formátu dat. Uživatel může také některé sloupce úplně ignorovat, nebo importovat data až od určitého řádku. Pokud jsou data na více listech, je nutné provést import pro každý list zvlášť.

U textového souboru uživatel určí oddělovací znak jednotlivých hodnot a také může specifikovat pořadí těchto hodnot. Stejně jako u excelového souboru může importovat data až od určitého řádku, nebo některé hodnoty zcela ignorovat. Toto nastavení importu je však součástí modulu `GUI`, viz kapitola 4.4.10..

Import z textového souboru provede metoda `ImportujTxtData` z třídy `cImportExportTxt`. Import probíhá v následujících krocích:

1. Ověření, že soubor existuje na zadané cestě a je přístupný pro čtení.
2. Rozdělení souboru na jednotlivé řádky.
3. Zpracování všech řádků od prvního požadovaného.
   (a) Rozdělení řádku na jednotlivé hodnoty podle zadaného oddělovacího řetězce.
   (b) Zjištění významu jednotlivých hodnot podle zadané struktury souboru.
   (c) Převedení textových hodnot na požadované typy.
4. Uložení získaných dat do databáze

Import excelového souboru se může zdát komplikovanější. Komunikace s aplikací Excel probíhá díky COM (*Component Object Model*), což je technologie (nebo standard), která umožňuje softwarovým komponentám vzájemně komunikovat. Takovou komponentou může být například aplikace Excel. Referencováním příslušných knihoven se nám tak do rukou dostává nástroj, díky kterému můžeme pohodlně z C# kódu pracovat například s jednotlivými listy nebo buňkami v excelovém sešitu. V kódu nesmíme zapomenout připojit knihovnu `Microsoft.Office.Interop.Excel`. Při práci s excelovou aplikací nesmíme zapomenout na uvolnění zdrojů. To obnáší například zavření excelového sešitu, uzavření aplikace (objekt typu `Excel.Application`) a jejich uvolnění z paměti.

Pokud by nedošlo k této proceduře, ve správci procesů by zůstavalo „viset“ několik instancí procesu *excel.exe*.

Modul pro import a export dat obsahuje třídy pro import a export do požadovaných formátů, třídu tvořící fasádu modulu, jádro modulu a také třídu, která zprostředkovává převod dat mezi různými datovými typy (`cTransformaceDat`).

4.4.7. Logovani - modul pro zaznamenávání výjimečných stavů

Modul pro zaznamenávání výjimečných stavů aplikace, různých chybových stavů a nestandardních situací byl do aplikace implementován jako poslední. Což se záhy ukázalo jako obrovská chyba, protože do chybového stavu se může dostat kterýkoliv modul, a je tedy nutné vyřešit komunikaci mezi moduly tak, aby nedošlo k narušení zapouzdření. Navíc často potřebujeme, aby se o takovém stavu dozvěděl uživatel a mohl nějakým způsobem reagovat na vzniklou situaci, nebo aby o ní byl alespoň informován. Proto se musí informace dostat až do prezentační vrstvy.

Jádro tohoto modulu je opět navrženo podle návrhového vzoru jedináček a celý modul klasicky podle vzoru fasáda. Objekt fasády vytváří pouze správce modulů. Ten také získá od modulu `GUI` delegáta metody, kterou volá ve chvíli, kdy chceme zobrazit informaci o chybovém stavu uživateli. Ostaní moduly získají delegáta na metodu ze správce modulů, kterou volají ve chvíli, kdy nastane chybový stav.

Posloupnost akcí při chybovém stavu v některém z modulů (ne v modulu `Jadro`, `GUI`, nebo `Logovani`) je následující:

1. Zavolá se metoda, kterou získala fasáda modulu jako argument od modulu `Jadro` (popřípadě od jiného modulu, který ji získal tranzitivně).
2. Modul `Jadro` zavolá metodu modulu `Logovani`, která provede zapsání do logovacího souboru (jedná se o metodu `ZpracujChybovyStav`).
3. Modul `Jadro` zavolá metodu, kterou dostal od modulu `GUI`, díky které se dostane informace o výjimečném stavu až k uživateli.

Zapsání ve skutečnosti proběhne do dvou souborů. Prvním je uživatelský log, kde je informace srozumitelná pro uživatele, jehož umístění si uživatel zvolí. Druhým je interní log, kde jsou uloženy všechny potřebné informace k nalezení příčiny chyby. Tyto soubory jsou ve formátu XML. Jeden záznam o chybovém stavu obsahuje následující položky:

- Modul - Název knihovny
- Trida - Název třídy

- Metoda - Název metody
- InterniZprava - Zpráva pro programátora
- Typ - Typ chybového stavu
- Datum - Datum a čas, kdy došlo k chybě
- UzivatelskaZprava - Zpráva o chybě pro uživatele

4.4.8. Jadro - modul pro správu ostatních modulů

Tento modul je někdy nazýván také jako správce modulů. Zajišťuje komunikaci mezi objekty, které na sebe „nevidí“. Tvoří prostředníka mezi modulem `GUI` a ostatními moduly. Grafické uživatelské prostředí tak volá jen metody ve fasádě správce modulů. Jádro tohoto modulu je navrženo podle návrhového vzoru jedináček. Objekt fasády modulu vytváříme pouze v nadřazeném modulu `GUI`. Ve třídě `cJadro` správce modulů vytváříme objekty fasád všech ostatních modulů.

Další třídy modulu jsou `cPraceSDb`, `cPraceSGrafy` a `cPraceSPorty`, které pouze vytahují metody z příslušných fasád ostatních modulů. Některé z těchto metod jsou přístupné přes fasádu správce modulů a může je tak využít jádro `GUI`.

Ve správci modulů je také definovaný uživatelský prvek pro ovládání grafů. Tento prvek nabízí uživateli možnost operovat s grafem. Uživatel tak může měnit tloušťku křivky, barvu křivky, symboly zobrazených bodů, posunovat graf, nastavovat četnost bodů a jiné. Začlenění ovladače do správce modulů bylo opět oceněno zápornými body při zpětném hodnocení kódu.

Při inicializaci jádra tohoto modulu také spouštíme časovače (*Timer*) pro odečítání dat z portu.

4.4.9. Porty - modul pro práci s paralelním portem

Dalo by se říci, že toto je významově nejdůležitější modul. Poskytuje metody pro zápis a čtění z paralelního portu. Bez této funkcionality by aplikace sloužila pro pouhé zobrazování databázových dat v grafech. Základem je knihovna `inpout32.dll`. Z této knihovny využijeme metodu `Out32`, která zašle data na paralelní port, a `Inp32`, která přečte data na paralelním portu. Více o paralelním portu a knihovně `inpout32.dll` najdeme v kapitole 4.5.. Modul sestává ze tří tříd:

- `cPristupKPortuAPI` - zaobalovací třída pro přístup k metodám z knihovny `inpout32.dll`
- `cJadroPorty` - jádro modulu
- `cFasadaPorty` - fasáda modulu

V jádru jsou definovány metody pro čtení dat z jednotlivých čidel, kontrolní metody, které zjistí, zda byla data úspěšně načtena a další. Tento modul obsahuje také dva časovače. První časovač se spustí každou minutu a způsobí kontrolu teplot na čidlech. Pokud jsou splněny nastavené podmínky, dojde k zapnutí nebo vypnutí čerpadla. Druhý časovač slouží k periodickému ukládání načtených hodnot do databáze. Frekvenci ukládání může uživatel měnit. Protože nechceme, aby měl tento modul přímý přístup k modulu pro práci s datbází, je po každém tiknutí časovače vyvolána událost, na kterou reaguje správce modulů. Ten se postará o režii kolem kontroly a ukládání dat.

4.4.10. GUI - modul grafického uživatelského rozhraní

Modul `GUI` představuje prezentační vrstvu. Obsahuje všechny formuláře, nabídky a další grafické prvky, se kterými se uživatel setká (když pomineme grafické prvky zmíněné v kapitolách 4.4.3., 4.4.4. a 4.4.8.). Nenalezneme zde třídu představující fasádu modulu, protože stojí nejvýše v hiearchii modulů, a jediný modul, který komunikuje s `GUI`, je `Jadro`. Ten však leží v nižší vrstvě, a proto komunikuje jen nepřímo prostřednictvím událostí. Modul můžeme rozdělit na tři hlavní části:

- Třída `cJadroGUI`, která tvoří jádro tohoto modulu
- Hlavní okno aplikace s přidruženými uživatelskými graf. prvky (úvodní obrazovka, předpověď počasí, a další)
- Formuláře a grafické prvky pro průvodce importem dat

V jádře modulu vytváříme objekt fasády správce modulů (viz kapitola 4.4.8.), skrz kterého máme dostupnou funkcionalitu některých dalších modulů. Pokud například cheme při zavření aplikace uložit velikost a umístění hlavního okna, požádáme správce o vrácení aktuálního nastavení, ve kterém nahradíme příslušné hodnoty. Následně správci oznámíme požadavek uložení nového nastavení. Veškeré podrobnosti komunikace správce s ostaními moduly (v tomto případě s modulem `Nastaveni`) nám jsou na tomto místě skryty.

Hlavní okno aplikace (třída `cGUIHlavniOkno`) má na starost správné reakce na uživatelovy požadavky a zobrazuje všechny důležité informace, které by uživatel mohl požadovat. Součástí aplikace je i velmi obecná předpověď počasí. Základem je knihovna `AnimaOnline Weather API`, kterou naleznete na adrese `http://awapi.codeplex.com/`. Při rozhodování se, jak předpověď implementovat, vstupovaly do hry následující faktory:

- Předpověď počasí není hlavní částí programu, a proto musí být co nejjednodušší na implementaci.
- Díky předchozímu požadavku tak chceme využít již existující službu.

- Protože nám stačí jen základní funkcionalita, použití existující komponenty musí být jednoduché.

Konečné rozhodování bylo mezi službou *Yahoo! Weather Feed* a *Google Weather Feed*. Vybrali jsme nakonec druhé řešení, protože první zmiňované porušilo třetí podmínku.

Přidáním příslušné reference na knihovnu `Animaonline Weather API.dll` a zaregistrováním jmeného prostoru (`using Animaonline.WeatherAPI;`) získáme přístup k důležité metodě `WeatherAPI.GetWeather(LanguageCode, string, bool)`. V prvním argumentu specifikujeme jazyk, ve kterém chceme vrácené informace zobrazit. V tomto výčtovém typu není bohužel čestina zahrnuta. Použití češtiny je možné po úpravě kódu, ale to by znamenalo menší licenční komplikace. Druhý argument určí místo, pro které chceme vrátit předpověď. Je více možností, jak specifikovat místo, ale nám stačí zadání názvu města. Posledním argumentem určíme, zda chceme teplotu vrátit v jednotkách stupňů Celsia.

Výsledkem volání této metody je objekt typu `WeatherData`, který zapouzdřuje data XML vrácená službou *Google Weather Feed*. Z tohoto objektu využijeme následující údaje:

- `temp_c` - aktuální teplotou ve zvolených jednotkách
- `conditionTODAY` - řetězec s aktuálním stavem počasí ve zvoleném jazyce (v angličtině, proto jej ve vlastní metodě překládáme)
- `highTODAY` - maximální předpovězená teplota pro dnešní den
- `lowTODAY` - minimální předpovězená teplota pro dnešní den
- `conditionTOMORROW` - řetězec s předpovězeným stavem počasí pro zítřejší den
- `highTOMORROW` - předpovězená maximální teplota pro zítřejší den
- `lowTOMORROW` - předpovězená minimální teplota pro zítřejší den
- `conditionDAY2` - řetězec s předpovězeným stavem počasí pro pozítří
- `highDAY2` - maximální předpovězená teplota pro pozítří
- `lowDAY2` - minimální předpovězená teplota pro pozítří

V metodě `VratPopisPocasi` přeložíme anglický stav počasí do češtiny a podle stavu zvolíme odpovídající obrázek a vytvoříme ikonu, kterou zobrazíme v systémové liště. Předpověď počasí se obnovuje po deseti minutách. Nutnou podmínkou pro využití předpovědi počasí je připojení k internetu.

4.4.11. Vypocty - modul pro provádění výpočtů

Tento modul byl do aplikace přidán pro možné budoucí rozšíření a v předváděné verzi aplikace (verze 2.23) nemá zatím reálné využití. Bylo však nutné počítat s tímto modulem v navrhované architektuře. Uvažuje se, že bude poskytovat metody pro různé náročnější výpočty, jako je například celkový zisk systému.

4.4.12. Generator - modul pro generování dat

Stejně jako modul z kapitoly 4.4.11. nemá mnoho využití v předváděné verzi aplikace. Původně některé jeho části sloužily k simulování načítání dat z portů generováním náhodných teplot podle určitého algoritmu. Byl však v systému ponechán pro možné budoucí rozšíření. Uvažuje se o generování různých modelů předpovědí počasí. Díky těmto modelům by bylo možné sestavit scénaře pro práci solární soustavy. Podle těchto modelů by došlo k automatickému nastavení solární soustavy tak, aby docházelo k co nejmenším ztrátám tepla ve dnech s menší sluneční aktivitou.

4.5. Paralelní port a knihovna inpout32.dll

4.5.1. Paralelní port

Informace byly v této kapitole čerpány převážně z elektronických publikací [5], [6] a [7]. Paralelní port, někdy též zkráceně označovaný LPT (*Line Printer Port*), byl poprvé uveden v osobním počítači (`Personal Computer`) představeném firmou IBM v roce 1981.

Jeho hlavní využití bylo spojováno s tiskárnami s velkým výkonem. Hlavní výhodou oproti sériovému portu byl paralelní přenos osmi bitů dat, zatímco sériový port data přenášel bit po bitu. Normován byl až v roce 1994 jako IEEE 1284. Tato specifikace určila tři typy konektorů, které nesly označení 1284-A, 1284-B a 1284-C. První zmiňovaný je konektor, který se také označuje DB25 (25 pinů ve dvou řadách), a najdeme jej na starších základních deskách. Druhý a třetí typ jsou konektory rozhraní Centronics. Pokud bude dále v textu zmíněn konektor paralelního portu, máme vždy na mysli konektor DB25.

Doporučená délka připojovacího kabelu je uváděna mezi 1,8 metru (viz [6]) a 5 metry (viz [5]) při dodržení některých zásad. Jednou z nich je doporučení, aby každá signálová linka měla v kabelu svůj zemnící vodič, se kterým vytvoří zkroucený pár (*twisted pair*). Pokud je z nějakého důvodu potřeba vést signál na delší vzdálenosti, zpravidla se to řeší vyvedením portu až k zařízení. Výstupní signály jsou definovány TTL logickou úrovní signálů (log. 1 odpovídá +3,5 V až +5 V a log. 0 hladině 0 V až +0,4 V). Proto se může na vstup přivést jen napětí

mezi +0 V až +5 V. V normě IEEE 1284 je rozlišeno pět různých režimů přenosu dat:

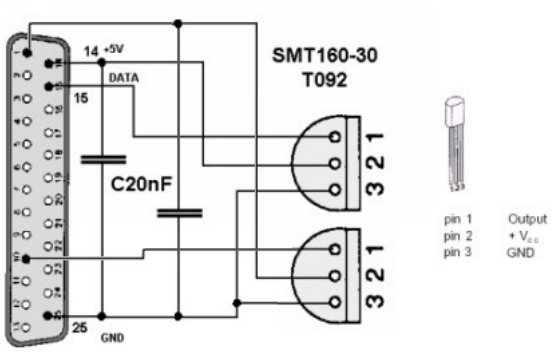
- SPP Mode - Compatibility mode (Centronics mode)
- Nibble Mode
- Byte Mode
- EPP Mode (Enchanced Parallel Port Mode)
- ECP Mode (Extended Capabilities Mode)

Výchozí by měl být vždy režim kompatibility (SPP režim), aby bylo možné komunikovat se zařízením bez jakýchkoliv úprav. Poslední tři jmenované režimy nejsou zpětně komatibilní s původními paralelními porty, a k jejich použití je potřeba hardwarově upravený port. Pro řízení přenosu v SPP režimu slouží následující signály: STROBE (vzorkování), BUSY (zaneprázdněn) a ACKNLG (potvrzení). Přenos dat probíhá v následujících krocích:

- Vložení dat na signálové vodiče (zápis dat do datového registru).
- Ohlášení připojenému zařízení přítomnost nových dat - to je provedeno aktivní úrovní signálu STROBE.
- Vybuzení signálu BUSY sestupnout hranou signálu STROBE - signál BUSY je v aktivní úrovni až do chvíle, kdy je zařízení připraveno přijmout nová data.
- Potvrzení přijetí dat zařízením vysláním signálu ACKNLG.
- Ukončení cyklu přenosu slova přechodem signálů BUSY a ACKNLG do neaktivního stavu (úroveň log. 1).

Přestože máme k portu připojena teplotní čidla, na kterých odečítáme teplotu (směr od zařízení k počítači), máme režim portu nastaven na SPP. Čidla jsou zapojena podle obrázku 7. na STATUS piny (10 a 15), což jsou výstupní piny (směr od počítače k zařízení).

Snímač teploty (čidlo) je zapojen podle jednoduchého zapojení (podle [8]), které využívá převodníku teplota/střída typ SMT 160-30, umístěného v pouzdře TO92. Součástky jsou umístěné v pouzdře, napájené přímo na vývody konektoru DB25 (25-pin D-SUB). Čidla jsou pak vyvedena pomocí ohebného kabelu mimo konektor. Spínač čerpadla je umístěn v samostatné krabiččce síťové zásuvky 230 V. Ovládací signál z paralelního portu spíná relé pomocí tranzistoru BC 548, které svým kontaktem připojune spotřebič (v našem případě čerpadlo). Tento spínač je napájen ze samostatného stejnosměrného zdroje 6V.



Obrázek 7. Zapojení teplotních čidel

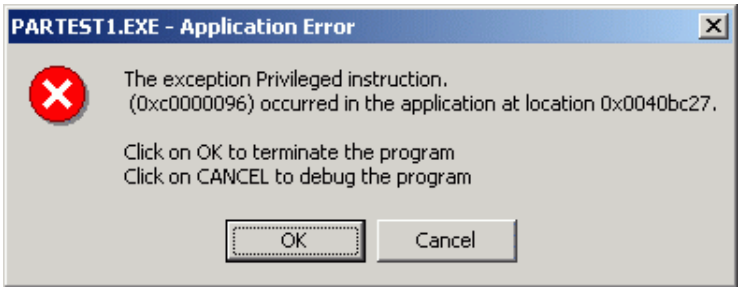
Čidlo generuje obdelníkový signál s frekvencí 1 kHz až 4 kHz v závislosti na teplotě, se střídou podle vzorce $DC = 0,32 + 0,0047 * t$, kde t je teplota ve stupních Celsia (viz [8]). Teplotní rozsah čidla udávaný výrobcem je od -45 °C do +150 °C s odchylkou od přesnosti 0,7 °C. Čidlo se tak hodí k měření v situacích, kdy nám stačí přesnost na celé stupně Celsia.

Pro výpočet střídy provádíme cyklus o dvěstě tisících opakování, ve kterém zjišťujeme, zda je na pinu, ke kterému je připojeno čidlo, jednička či nula. Abychom dosáhli větší přesnosti, před začátkem cyklu nastavíme prioritu procesu, ve kterém probíhá měření, na `Real-time`. Po skončení měření prioritu nastavím zpět na hodnotu `Normal`. Pokusem bylo zjištěno, že zvýšením počtu opakování cyklu již nedochází k většímu zpřesnění výpočtu, což je způsobeno technickými limity čidla. Stejně tak nedošlo ke zpřesnění větším počtem měření teploty a následným výpočtem průměru.

4.5.2. Přístup k I/O portům v systému Windows

U operačních systémů Windows založených na jádře NT (Windows NT, Windows 2000, Windows XP) není možný přímý přístup k I/O portům. Není to vina systému NT, ale tzv. chráněného režimu procesorů, který je u procesorů 386 a vyšších, a který právě Windows NT (a vyšší) využívají [9]. Chráněný režim poskytuje 4 úrovně přístupu k hardware, nazývané jednoduše ring 0 až ring 3. Ve skutečnosti existuje ještě tzv. SMM režim (*System Management Mode*), do kterého může procesor přejít po vyvolání SMI (*System Management Interrupt*). V úrovni ring 0 běží samotné jádro operačního systému spolu s ovladači zařízení. Úrovně ring 1 a ring 2 se nepoužívají a běžné aplikace jsou pak spouštěny v úrovni ring 3.

Pokud chce aplikace (spuštěná v úrovni ring 3) přistupovat k I/O portu, musí tak učinit přes ovladače běžící v ring 0. Přístup aplikací k ovladačům je prováděn podle povolovací masky (I/O Permission Bit Map). Když požaduje aplikace přístup k portu, procesor otestuje povolovací masku, ze které zjistí, zda



Obrázek 8. Chybové hlášení při pokusu o přímý přístup k portu

lze k portu přistoupit. U systémů před NT došlo při pokusu přístupu na port k výjimce, která byla odchycena operačním systémem a zpracována. Zpracování upravilo povolovací masku a další přístup k portu proběhl v pořádku. U systémů s jádrem NT (popřípadě s jádrem WinMin u Windows Vista a Windows 7) je vyvolaná výjimka zpracována tak, že je aplikace přerušena. O přerušení je uživatel informován chybovým hlášením na obrázku 8..

Existují dvě řešení této situace. První možností je napsání si vlastního ovladače, který by běžel na úrovni ring 0. Druhou, ovšem ne příliš doporučovanou možností, je úprava povolovací masky. V této aplikaci jsme využili již existujícího ovladače `inpout32.dll`. Tento ovladač v sobě zahrnuje ovladač běžící v režimu jádra. Jeho použití v aplikaci bylo otestováno na operačních systémech Windows XP, Windows Vista a Windows 7.

Pokud je zavolána některá z funkcí ovladače, provede se otestování verze operačního systému. Pokud se jedná o systém WIN9x, použije se funkce `_intp() nebo _outp`. Jinak se nainstaluje ovladač běžící v režimu jádra (`Hwinterface.sys`) a komunikace s portem probíhá přes jeho funkce.

4.6. Testování aplikace

Testování aplikace probíhalo ve dvou vlnách. Nejprve bylo testerům řečeno, ať se pokusí aplikaci jakýmkoliv způsobem přivést do chybového stavu. V nejlepším případě do stavu, po kterém následovalo vyvolání neošetřené výjimky. Ze začátku to nebyl vůbec těžký úkol, a tak se podařilo odladit některé velice závažné nedostatky. Po tomto hrubém testování přišly na řadu scénáře použití. Testující osoby byly rozděleny do tří skupin: programátoři, uživatelé s velmi dobrými znalostmi a uživatelé laici. Všichni dostali stejné zadání, které sestávalo z několika úkolů. Po vyplnění (mohlo být neúspěšné) byli požádáni o sepsání zkušeností s programem, co se týká uživatelské přívětivosti, jednoduchosti ovládání (popřípadě obtížnosti) apod.. Zadané úkoly byly následující:

- Nainstalovat a spustit aplikaci.
- Nastavit adresu paralelního portu.

- Nastavit soubor, kam budou ukládány chybové hlášky.
- Nastavit frekvenci odečítání teplot na co nejmenší hodnotu.
- Z přiloženého excelového souboru importovat data podle zadání.
- Nalézt lokální maximální teploty a vytvořit export do excelového souboru.
- Změnit některé vlastnosti grafu a nastavit rozsahy os na požadované rozsahy.

Po vyhodnocení byly některé vytknuté věci změněny. Bylo by určitě dobré, pokud by příště testování probíhalo častěji a byl vypracován kvalitní hodnotící dotazní.

5. Uživatelská dokumentace

V této sekci popíšeme aplikaci Solar Monitoring z uživatelského pohledu. Projdeme postupně všechny nabídky a ovládací prvky, ukážeme si, jak aplikaci správně používat a probereme některé na první pohled ne uplně jasné části.

5.1. Instalace a požadavky

Aplikace byla vyvíjena pro operační systém Windows XP a výše. Otestována byla v operačních systémech Windows XP, Windows Vista a Windows 7 (64 bitová verze). Ke svému běhu potřebuje .NET Framework 3.5. Pokud nebude na cílovém počítači nalezen, nainstaluje se spolu s aplikací. Před tím dojde k jeho stažení z internetu. Aplikace spolupracuje s programem Microsoft Office Excel. Pokud nebude na cílovém počítači nainstalován, nebude možné provádět import a export do `xls` formátu. Na další funkce aplikace to ale nemá vliv. K nainstalování aplikace jsou potřeba dva soubory: `SolarMonitoring.msi` a `SolarMonitoring.exe`. Instalace začne po spuštění souboru `SolarMonitoring.exe`. Instalace je poměrně přímočará a sestává ze tří kroků:

- V prvním okně je uživatel obeznámen s instalovaným programem.
- Pokud se rozhodne pokračovat v instalaci, může specifikovat cestu, kam bude program nainstalován. Pokud nebude měnit cestu, nainstaluje se program do složky *Program Files*, která se nachází na systémovém disku.
- V posledním kroku uživatel potvrdí zadané údaje (cestu) a program bude nainstalován.

Pokud uživatel nezvolí jinak, je aplikace nainstalována na umístění `C:\Program Files\Roman Loník\SolarMonitoring\`. Program se spouští souborem `SolarMonitoring.exe`. Do této složky jsou při instalaci nakopírovány knihovny (soubory s příponou *dll*), které jsou však zkopírovány jako skryté soubory. Do složky `ProgramData\SolarMonitoring` jsou nakopírovány další soubory. Jedná se o soubor databáze (`SolarMonitoring.mdb`) a soubory logů (`interniLog.txt`). Je zde také vytvořena adresářová struktura pro ukládání exportovaných dat.

5.2. Hlavní okno aplikace

Při spuštění aplikace se jako první zobrazí úvodní okno (viz obrázek 9.), které zmizí po inicializaci potřebných součástí aplikace a je nahrazeno hlavním oknem aplikace.

Obrázek 9. Úvodní obrazovka aplikace

Postupně jsou inicializovány tyto součásti:

- Hlavní okno aplikace
- Databáze
- Uživatelské rozhranní
- Správce modulů
- Datové operátory
- Ovladače grafů
- Grafy
- Předpověď počasí

Hlavní okno aplikace sestává z pěti důležitých součástí, které jsou označeny čísly na obrázku 10..

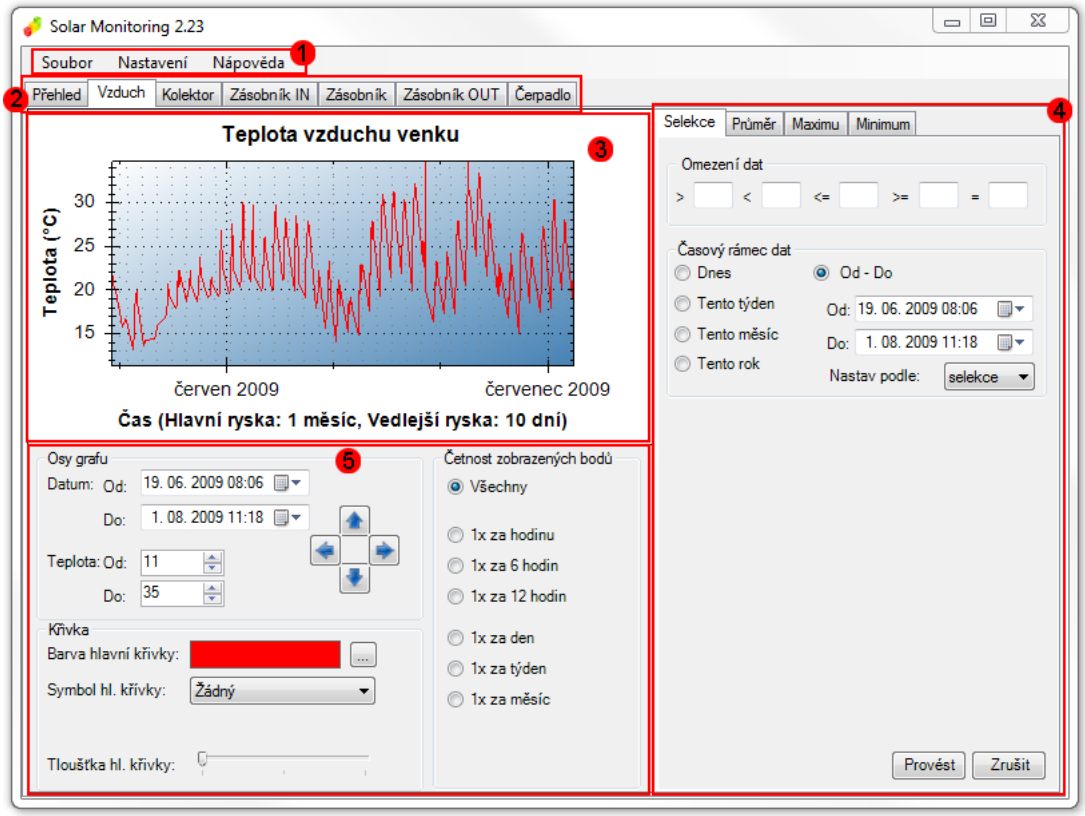
1. Hlavní menu - Z hlavního menu můžeme provést pět činností: importovat data do aplikace, zavřít aplikaci, otevřít nastavení, zobrazit nápovědu a zobrazit informace o programu.

2. Karta záložek - První záložka zobrazuje souhrnné informace a bude popsána později. Zbývajících šest záložek se vztahuje k jednotlivým čidlům a spínači čerpadla. Každá z těchto šesti záložek má vlastní graf, ovladač grafu a datový operátor.

3. Datový operátor - Datový operátor slouží k výpočtům nad daty a následnému exportu výsledků.

4. Graf - V grafu jsou zobrazena data uložená v databázi. Podrobněji se grafům budeme věnovat v sekci popisující ovladač grafu.

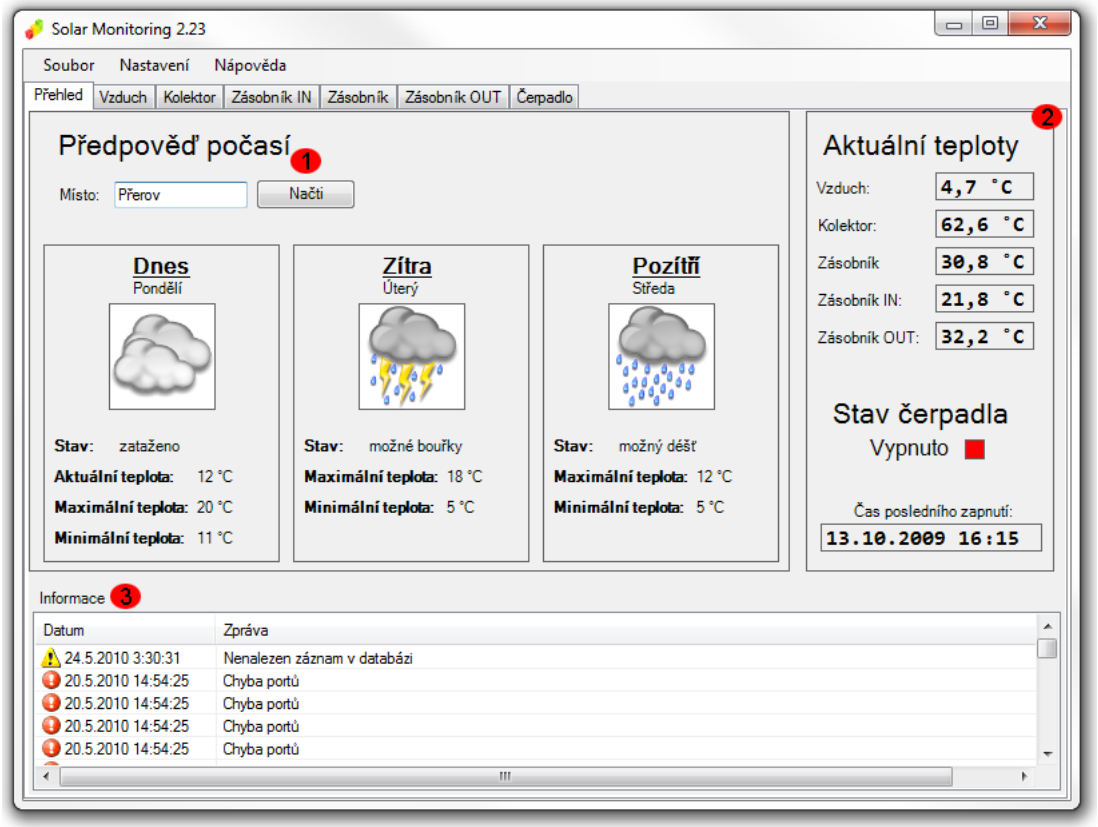
5. Ovladač grafu - Umožňuje nastavovat některé vlastnosti grafu, jako je barva křivky, symbol bodů apod..



Obrázek 10. Hlavní okno programu

Na obrázku 11. je hlavní okno s aktivní záložkou *Přehled*. Na této záložce najdeme informaci o stavu počasí pro zvolené místo, poslední naměřené teploty na čidlech, stav čerpadla a čas posledního zapnutí čerpadla. Ve spodní části okna se nachází informace o chybových stavech aplikace. Tyto informace jsou

ukládány do souboru uživatelského logu, jehož umístění můžeme změnit v nastavení. Konkrétně musíme zvolit položku *Nastavení aplikace - Hlášení a upozornění - Soubor*.



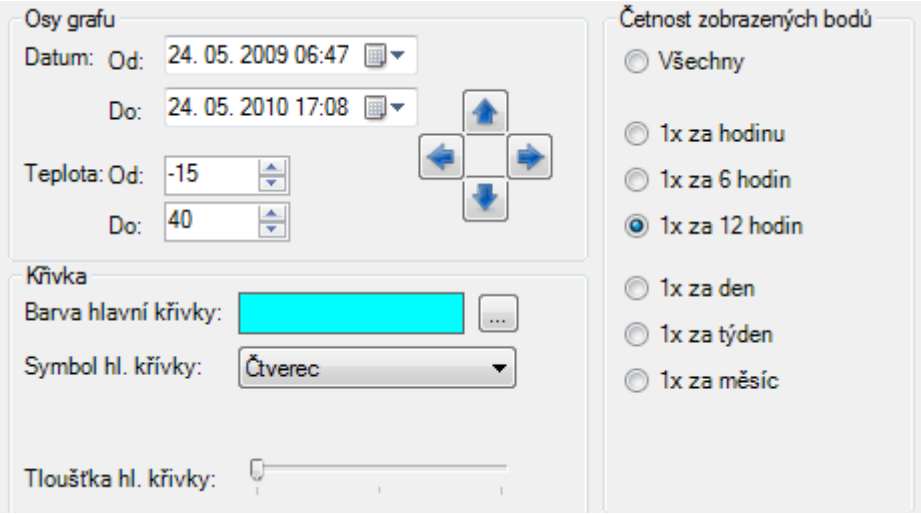
Obrázek 11. Hlavní okno programu - záložka Přehled

Aktuální stav počasí můžeme vidět v systémové liště (viz obrázek 12.). Kliknutím pravým tlačítkem myši na ikonu aplikace v systémové liště vyvoláme nabídku, kde máme možnost ukončit aplikaci, spustit průvodce importem dat, nebo se přepnout na některou záložku s informacemi o vybraném čidle (popřípadě spínači čerpadla). Dvojklik levým tlačítkem aplikaci minimalizuje, nebo aktivuje.

Obrázek 12. Zobrazení stavu počasí v systémové liště

5.3. Manipulace s grafy

V aplikaci se nachází celkem šest grafů. Prvních pět se váže k teplotním čidlům, šestý ke spínači čerpadla. Grafy můžeme ovládat přímo, nebo pomocí ovladače, který se nachází vždy pod grafem. Následující popis je společný pro všechny grafy. Jediná výjimka nastává u grafu zobrazujícího informace o spínání čerpadla. U tohoto grafu je možné měnit rozsah jen na časové ose X. Osa Y vyjadřuje údaj o naměřené teplotě (u grafu čerpadla nese dvě hodnoty, zapnuto či vypnuto).



Obrázek 13. Ovladač grafu

Pokud chceme volně manipulovat osami grafu (to znamená konstantně měnit jejich rozsah), stiskneme nad grafem prostřední tlačítko myši a pohybujeme se požadovaným směrem. Po opětovném uvolnění tlačítka nastane přenastavení údajů o rozsahu os. Tento způsob manipulace slouží k rychlému přesunu, kdy nepotřebujeme absolutní přesnost. Pokud potřebujeme osy nastavit na přesné hodnoty, provedeme to v ovladači grafu (viz obrázek 13.). Časovou osu nastavíme vepsáním časového údaje (pozor na neplatné hodnoty, které mohou vzniknout při pokusu o nastavení spodní meze na vyšší hodnotu, než má horní mez), nebo rozkliknutím nabídky vlevo. Po rozkliknutí se objeví kalendář, ve kterém můžeme pohodlně vybrat datum. Teplotní osu nastavíme vepsáním požadované teploty, nebo klikáním na příslušné prvky. Čtyři směrová tlačítka slouží k pohybu vertikálně nebo horizontálně.

Další možností je přiblížení nebo oddálení grafu. To provedeme stisknutím levého tlačítka nad grafem a současným tažením myši. Objeví se čárkovaný obdélník, který vymezuje přibližovanou oblast. Druhou možností je použití

kolečka myši jedním nebo druhým směrem. Pokud se chceme vrátit ke stavu před přiblížením (resp. oddálením), stiskneme nad grafem pravé tlačítko myši a ve zobrazené nabídce vybereme položku *Zpět přiblížení*. Pokud chceme zrušit všechna přiblížení, vybereme ze stejné nabídky položku *Zpět vše*.

Na ovladači grafu můžeme měnit i další vlastnosti grafu. Barvu křivky změníme pomocí dialogu, který se zobrazí po stisknutí tlačítka vedle barevného panelu. Pokud se chceme rychle vrátit k výchozí červené barvě, provedeme to levým dvojklikem na barevný panel. Pod ním se nachází prvek pro výběr symbolů jednotlivých bodů. Na křivce nemusíme zobrazovat symbol žádný. Úplně dole se nachází prvek pro ovlivnění tloušťky křivky. Na výběr jsou tři možnosti. Volba *Četnost zobrazovaných bodů* najde uplatnění na pomalejších sestavách. Umožňuje zobrazovat jen některé body z celého rozsahu. Ovšem výpočty jsou prováděny nad všemi body, takže se nemusíme bát zkreslených výsledků. Ovladač grafu má dva možné vzhledy podle velikosti hlavního okna. Liší se jen rozestavěním kontrolních prvků.

Kontextová nabídka grafu (stisknutí pravého tlačítka myši nad grafem) nabízí tyto možnosti:

- Zkopírovat - Zkopíruje obrázek grafu do schránky.
- Uložit obrázek jako - Otevře okno pro uložení grafu jako obrázku. Na výběr jsou tyto formáty: emf, png, gif, jpg, tif, bmp.
- Zobrazit hodnoty - Zobrazí popisky hodnot. Stačí podržet kurzor myši nad křivkou.
- Zpět přiblížení - Vrátí graf do stavu před posledním přiblížením.
- Zpět vše - Vrátí graf do původního stavu.

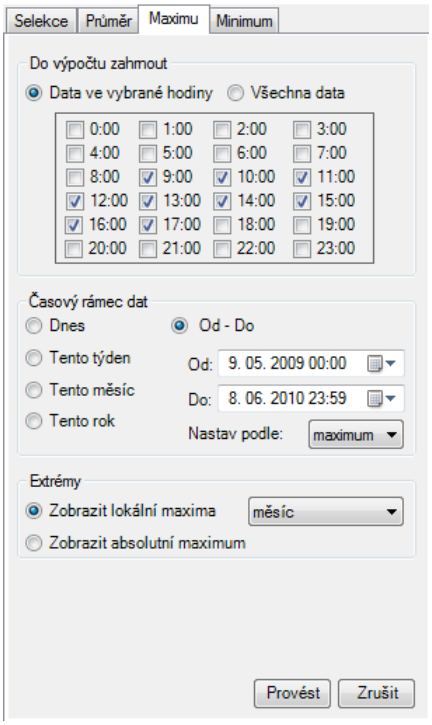
Grafy je možné přiblížit až na úroveň jednotlivých vteřin nebo oddálit na rozmezí několika desetiletí. Aktuální jednotky jsou zobrazeny pod časovou osou. Jednotky na ose Y jsou vždy stejné, u grafu spínání čerpadla vyjadřují stav zapnuto nebo vypnuto.

5.4. Datové operátory

Datové operátory slouží pro práci se získanými daty. Nabízí nám možnost několika operací:

- Selekce - Výběr dat podle zadaných kritérií.
- Průměr - Výpočet průměrné teploty v zadaném období.
- Extrémy - Nalezení extrémních hodnot (absolutních nebo lokálních.)

Společnou položkou na všech záložkách operací je výběr časového rámce dat. Data můžeme časově omezit na aktuální den, týden, měsíc, rok, nebo můžeme specifikovat vlastní rozmezí. Možnost `Nastav podle` umožňuje nastavit časový rozsah na aktuální záložce operace podle rozsahu jiné operace. Tlačítko `Provést`, které je na všech záložkách operací, provede aktuální operaci se zadanými parametry a zobrazí výsledek. Výsledné okno bude popsáno níže.



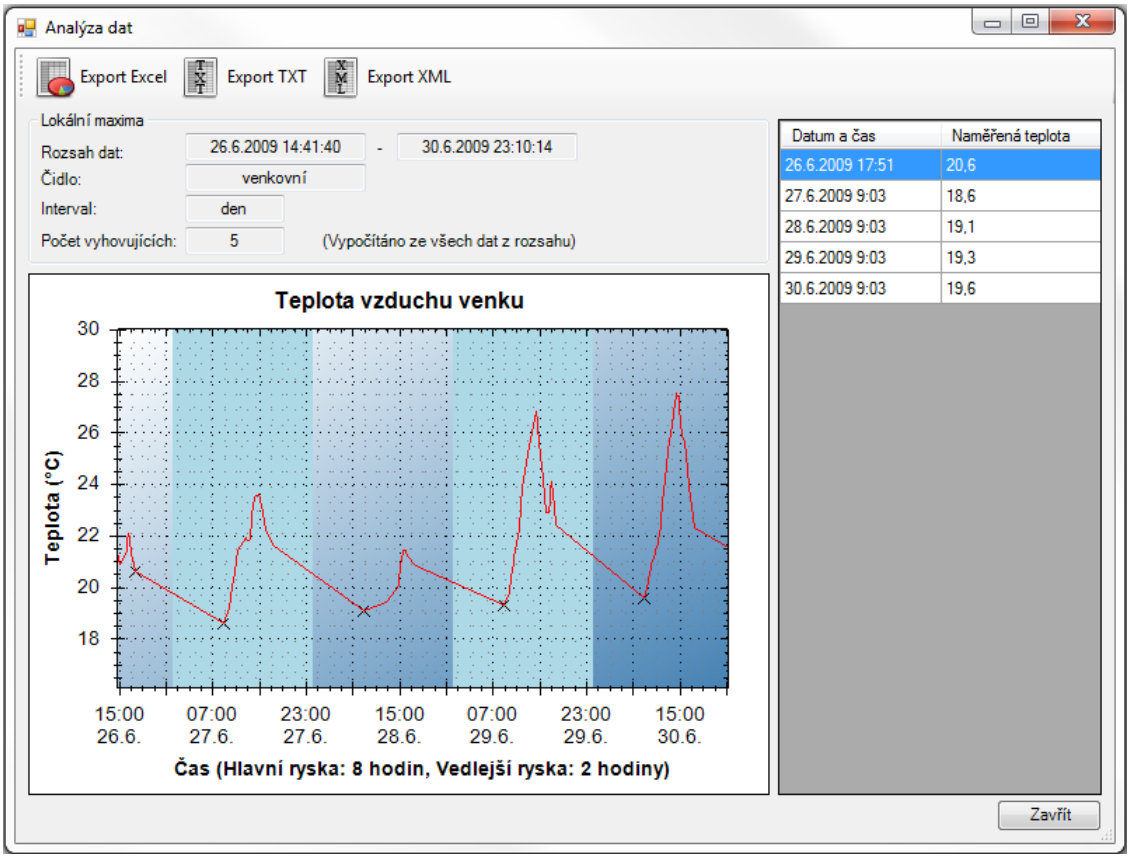
Obrázek 14. Datový operátor

U selekce můžete vybrat až dvě kritéria omezující selektovaná data. Vyhovující data musí splňovat všechna kritéria zároveň.

Operace výpočtu průměru umožňuje počítat průměr ze všech dostupných dat ve zvoleném časovém úseku, nebo do výpočtu zahrne jen data naměřená v požadované hodiny. Tak můžeme vypočítat zvlášť denní a noční průměr apod.. Data, která budou zobrazena jako vyhovující, musí mít teplotu shodnou s průměrnou. Máme na výběr i možnost zobrazit data, která se liší maximálně o zadanou hodnotu (v obou směrech, průměrná teplota pak tvoří střed intervalu).

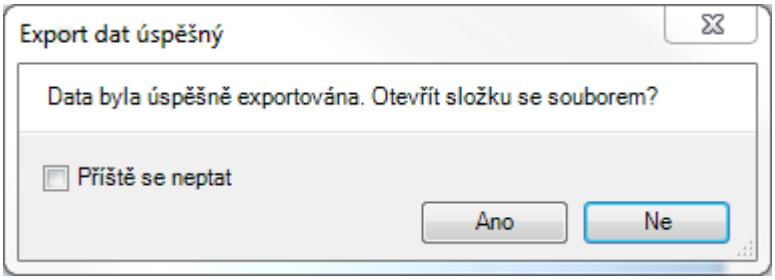
U extrémů je stejná volba výběru dat, která budou zahrnuta do výpočtu, jako u průměru. Dále můžeme zvolit hledání lokálního extrému a omezit lokální interval. Toto nastavení je shodné u operace maxima i minima.

Okno s výsledkem vidíme na obrázku 15.. Největší plochu okna zaujímá graf, na kterém jsou zobrazena data vyhovující podmínkám operace. Nad grafem máme informace o prováděné operaci, nad kterými jsou tlačítka pro export výsledku



Obrázek 15. Zobrazení výsledku operace s daty

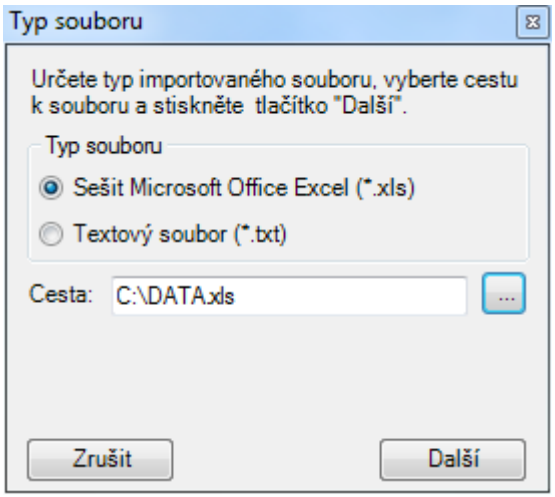
do některého z podporovaných formátů. Pokud není na počítači nainstalován program Microsoft Office Excel, není možné provádět export do formátu `xls`. Napravo od grafu leží panel s tabulkou vyhovujících dat. Při exportu se nejprve zobrazí dialog pro výběr umístění exportovaného souboru. Po uložení se objeví okno (viz obrázek 16.). Po kliknutí na možnost `Ano` se otevře složka s vybraným exportovaným souborem.



Obrázek 16. Dotaz na zobrazení složky s exportovaným souborem

5.5. Import dat

Průvodce importem dat je dostupný třemi možnostmi. Pomocí klávesové zkratky `CTRL + I`, z hlavního menu `Soubor - Import Dat` nebo ze systémové lišty. Zvolením jedné z těchto možností zobrazíme okno pro výběr souboru (viz obrázek 17.). Na výběr je možné importovat data z textového souboru nebo ze souboru aplikace Microsoft Office Excel, pokud je tato aplikace nainstalována.



Obrázek 17. Výběr souboru k importu

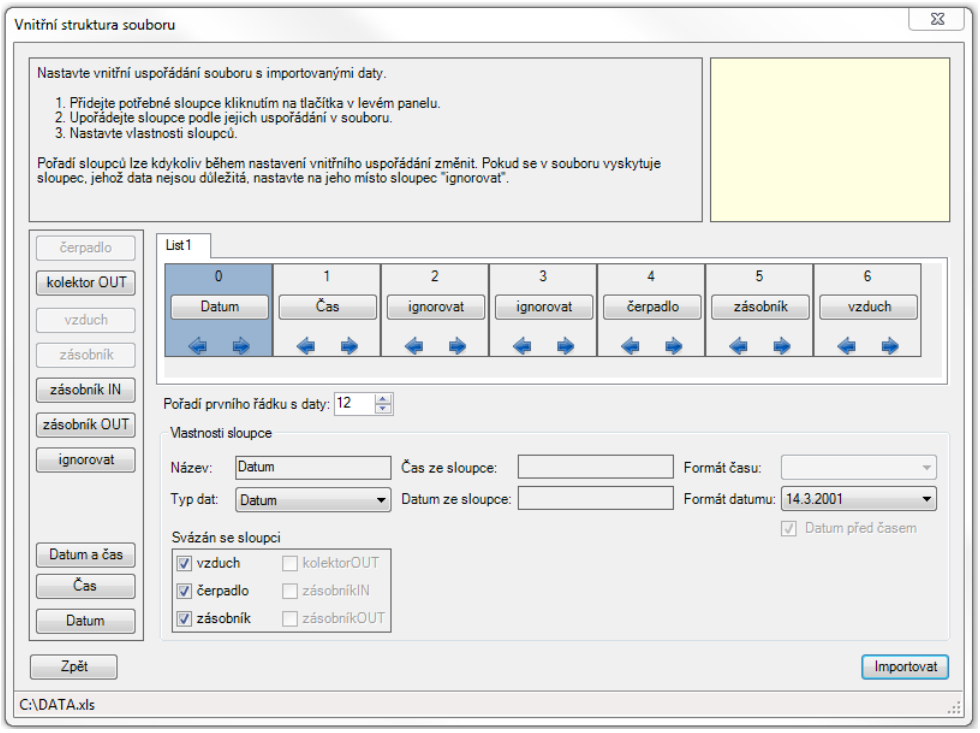
Na další obrazovce můžeme nastavit vnitřní strukturu souboru. V levém menu máme na výběr tlačítka s typy jednotlivých čidel a navíc tlačítka pro sloupce s časovými hodnotami. Aby byl import povolen, musí mít každý sloupec čidla (popřípadě spínače čerpadla) přiřazen sloupec s časovým údajem. Kliknutím na tlačítko příslušného typu sloupce umístíme tento sloupec na list. Pořadí listů můžeme měnit buď klávesovými šipkami, nebo stisknutím šipky na daném sloupci. Pokud chceme sloupec odstranit, opět klikneme na jeho tlačítko.

U časových sloupců (`Datum a čas`, `Datum`, `Čas`) můžeme zvolit formát.

Pokud jsou časové údaje v jiném formátu, než je v nabídce, nebude import úspěšný. Svázání konkrétního sloupce s časovým sloupcem provedeme následujícím způsobem:

1. Zvolíme příslušný časový sloupec
2. Ve spodní nabídce `Svázán se sloupci` zaškrtneme příslušný sloupec čidla

Dvojklikem do nabídky `Svázán se sloupci` vybereme nebo odznačíme všechny sloupce. Pokud chceme data importovat až od určitého řádku, nastavíme položku `Pořadí prvního řádku s daty` na příslušnou hodnotu.



Obrázek 18. Průvodce importem excelového souboru

Na obrázku 18. máme nastaven import následujícím způsobem:

- Umístění importovaného souboru je ve spodním stavovém řádku (`C:\DATA.xls`).
- Importujeme hodnoty čidel `zásobník`, `vzduch` a `spínače čerpadla`.
- Časové údaje tyto sloupce získávají z prvních dvou sloupců (`Datum`, `Čas`).
- První importovaný řádek je dvanáctý řádek.
- Sloupce mezi údajem o čase a hodnotami sepnutí čerpadla ignorujeme.

5.6. Nastavení aplikace

K nastavení aplikace se dostaneme pomocí klávesové zkratky `CTRL + N` nebo z horního menu (`Nastavení`). V levé části okna vidíme strom nastavení, v pravé části se zobrazuje aktuálně vybraná položka nastavení. Nastavení obsahuje čtyři základní položky v kategoriích `Nastavení aplikace` a `Nastavení solární soustavy`:

- Hlášení a upozornění - V této části nastavujeme podmínky, za jakých nás aplikace upozorňuje. Upozornění může být zasláno na e-mail, nebo uloženo do souboru. Hlášeny mohou být tři druhy událostí. Chyba čtení dat na některém z čidel. Můžeme specifikovat, která čidla chceme sledovat (nemusíme sledovat žádné) a po kolika neúspěšných čteních je hlášení zasláno či uloženo. Druhou událostí je dosažení kritické teploty. Opět můžeme vybrat jen některá čidla ke sledování a dále můžeme zvolit, zda chceme hlídat jen dosažení teplot směrem nahoru nebo dolů. Poslední možností je sledování spínání čerpadla.

- Obecné nastavení - Zde nastavujeme obecné nastavení, které se týká celé aplikace. Nejdůležitější je položka `Adresa portu`. Jedná se o adresu paralelního portu, a pokud nebude správně nastavena, nebude fungovat odečítání teplot z teplotních čidel ani spínání čerpadla. Adresa se zadává v hexadecimálním tvaru. Standardně má paralelní port označovaný jako LPT1 adresu 0x378 a paralelní port označovaný jako LPT2 adresu 0x278. Adresu můžeme zadat bez bez počátečního řetězce `0x`. Aktuální adresu paralelního portu najdeme přes nabídku `Start - Ovládací panely - Systém - Správce zařízení - Porty (COM & LPT) - Paralelní port - Vlastnosti - Prostředky`. Z této nabídky nás zajímá počáční adresa rozsahu, kterou zadáme jako hodnotu položky `Adresa portu`.

  Další položkou je frekvence měření teplot. Minimální frekvence je pět minut, maximální je jeden den. Tlačítko `Restart` provede restartování odpočítávání do dalšího měření. To způsobí okamžité přečtení teplot. Poslední tři položky na této kartě nastavení nastavují zapamatování si rozsahů grafů při ukončení aplikace, zapamatování si velikosti a pozice hlavního okna a zakázání či povolení zobrazení dotazu na otevření složky při exportu dat.

- Nový zásobník - Na této kartě nastavujeme vlastnosti zásobníku vody. Můžeme si uložit nastavení několika různých zásobníků. Výchozí pak může být nastaven vždy jen jeden. Zásobník určíme jako výchozí pravým kliknutím myši na název zásobníku v levém stromu nastavení a výběrem položky `Nastavit výchozí`.

- Nové čerpadlo - Na této kartě nastavujeme vlastnosti čerpadla vody. Důležité jsou položky `Teplota zapnutí` a `Teplota vypnutí`. Určují, při

jakém rozdílu teplot dojde k zapnutí nebo vypnutí čerpadla. K zapnutí čerpadla dojde, pokud je rozdíl teplot na čidlech v kolektoru a zásobníku větší, než nastavená hodnota. K vypnutí čerpadla dojde ve chvíli, kdy je rozdíl teplot na vstupu a na výstupu ze zásobníku menší, než nastavená hodnota. Pokud uživatel nevytvoří nastavení čerpadla, jsou hodnoty pro zapnutí a vypnutí čerpadla nastaveny na 10 °C a 2 °C.

5.7. Otázky a odpovědi

1. *Jak nainstaluji aplikaci?* K nainstalování jsou potřebné dva soubory: `SolarMonitoring.exe` a `SolarMonitoring.msi`. Spuštěním kteréhokoliv z nich spustíte průvodce instalací.

2. *Jak aplikaci spustím?* V nabídce `Start - Programy - Solar Monitoring` zvolíte soubor SolarMonitoring a spustíte.

3. *Spustil jsem aplikaci, ale v grafem nevidím žádná data. Co dělám špatně?* Po instalaci je databáze načtených dat prázdná, proto není co zobrazovat v grafech. Můžete importovat data z textového souboru nebo souboru Microsoft Office Excel.

4. *Z čidel se nenačítají žádná data. Co musím nastavit?* Na kartě nastavení `Obecné nastavení` je potřeba nastavit adresu paralelního portu. Tu lze nalézt v nabídce `Start - Ovládací panely - Systém - Správce zařízení - Porty (COM & LPT) - Paralelní port - Vlastnosti - Prostředky`. Pro vyzkoušení odečítání teploty stiskněte tlačítko `Restartovat` na kartě *Obecné nastavení*. Způsobí okamžité přečtení teplot.

5. *Nikde nevidím možnost exportovat zobrazená data v grafu. Jak provedu export?* V levé části okna s grafem se nachází tzv. datový operátor. Pokud chceme vybrat data z daného rozsahu, stiskneme tlačítko `Provést` na kartě `Selekce` datového operátoru. Objeví se okno s výsledkem selekce, kde se v horním menu nachází tlačítka pro export.

6. *Chci zobrazit data z prvního měsíce roku 2010, která se liší od růměrné teploty maximálně o jeden stupeň. Jak to udělám?* V datovém operátoru vybereme kartu operace `Průměr`, nastavíme rozsah od 1.1.2010 do 31.1.2010 a zvolíme položku `S odchylkou maximálně`, kde nastavíme jeden stupeň Celsia. Poté stiskneme tlačítko `Provést`.

Závěr

Cílem práce bylo vytvořit aplikaci, která bude sloužit k monitorování a řízení provozu solární soustavy. První část textu se věnuje základním pojmům z oblasti solárních systémů, na kterou navazuje popis vlastní implementace a použití. Při vývoji a následném testování byla reálná solární soustava nahrazena dvěma čidly připojenými k paralelnímu portu počítače a jednoduchým obvodem se spínačem a spotřebičem, který simuluje připojení průtokového čerpadla.

Protože paralelní port není v .NET Frameworku přímo podporován a dnes se s ním příliš nesetkáme, bylo nutné vyřešit komunikaci s portem i pod operačními systémy, které využívají chráněný režim (OS s jádrem NT a WinMin). Tento problém se podařilo překonat díky externím ovladačům pro komunikaci s I/O porty. Další externí knihovna pomohla vyřešit problém s absencí komponenty pro práci s grafy, která je dostupná až v .NET Frameworku verze 4.

Ačkoliv se aplikace snaží pokrýt všechny zadané požadavky, slouží spíše jako odrazový můstek pro vytvoření obecné řídící aplikace, kterou by bylo možné přizpůsobit jakékoliv solární soustavě. Během vývoje se vyskytlo mnoho nápadů na její rozšíření, a proto by mohla posloužit jako výchozí bod k vytvoření nové aplikace, kterou by mohli využít majitelé solárních systémů.

Conclusions

The goal of this work was to create an application that will be used to monitor and control the operation of solar system. The first part deals with basic concepts in the field of solar systems, which is connected to a description of implementation and use of application. During the development and subsequent testing was the real solar system replaced by two sensors attached to the parallel port on the computer and a simple circuit with a switch and an appliance that simulates flow pump connection.

Since the parallel port is not in .NET Framework directly supported, and today we do not meet with it too, it was necessary to solve the communication of port and operating systems which use protected mode (OS with core NT and WinMin). This problem has been overcome by external drivers for communications with the I/O ports. Other external library helps to solve the problem of lack of components for working with charts, which are only available in .NET Framework 4.

Although the application is trying to cover all the specified requirements, it serves more as a springboard for a general control application, which could be adapted to any solar system. During the evolution there have been many ideas for its extension and therefore could serve as a starting point to create new applications that could benefit owners of solar systems.

Reference

[1] Kleczek, Josip *Slunce a jeho energie*
Elektronická publikace, 2004

[2] Klímová Silva, Ševčíková Lenka *Pasivní solární systémy*
Elektronická publikace, 2009

[3] McConnell, Steve. *Code Complete 2nd Edition*. Harper Perennial, New York, 2004 ISBN: 978-0735619678

[4] *ZedGraph: A flexible charting library for .NET*
Elektronická publikace, 2003.

[5] Olmr, Vít *Paralelní port - LPT (IEEE 1284)*
Elektronická publikace, 2005

[6] Peacock, Craig *Interfacing the Standard Parallel Port*
Elektronická publikace, 2005

[7] Tišnovský, Pavel *Interfacing the Standard Parallel Port*
Elektronická publikace, 2008

[8] Ištok, Milan *Nejjednodušší on-line teploměr*
Elektronická publikace, 2003

[9] Mrázek, Oldřich *Přímý přístup na I/O porty počítače*
Elektronická publikace, 2004

[10] *Animaonline Weather API*
Elektronická publikace, 2009.