ČVUT Praha
Fakulta elektrotechnická
Katedra řídící techniky

Diplomová práce

Návrh a implementace softwaru pro vizualizaci telemetrických dat pro malý proudový motor

Bc. Miroslav Hájek

vedoucí práce: Ing. Ondřej Špinka

Praha 2009

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Miroslav Hájek**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Návrh a implementace softwaru pro vizualizaci telemetrických dat pro malý proudový motor**

Pokyny pro vypracování:

1. Navrhněte a implementujte komunikační protokol pro přenos měřených dat z řídicí jednotky malého proudového motoru.
2. Navrhněte a implementujte software pro záznam a vizualizaci naměřených dat.
3. Vytvořte softwarový emulátor řídicí jednotky turbíny a Vámi vytvořený softwarový produkt důkladně otestujte s použitím tohoto emulátoru.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Ondřej Špinka

Platnost zadání: do konce zimního semestru 2008/09

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 15. 1. 2009

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……

……
Miroslav Hájek

Věnováno Markétce

Abstrakt

Diplomová práce se zabývá vývojem a implementací softwarového vybavení pro řídicí jednotku malého proudového motoru (turbíny) pro model letadla.

V první části se věnuje vývoji vlastního komunikačního protokolu, formátem posílaných dat a specifikací paketů. V druhé části práce popisuje tvorbu a implementaci softwarových klientů v jazyce C#, jak pro osobní počítač tak i pro mobilní zařízení typu Pocket PC. Pro testování byl vyvinut program emulující chování řídicí jednotky, pomocí kterého byly ověřeny základní funkce a programy byly laděny z výkonnostního hlediska.

Klíčová slova

vizualizace, C#, Pocet PC, komunikace, protokol, malý proudový motor, turbína

Abstract

This thesis deals with development and implementation of telemetry and vizualization software, used for data processing and communication with a small jet engine control unit.

The first part of this work introduces the communication protocol, used for command and data transfers to and from the control unit. The second part describes design and implementation of a vizualization software clients for PC and pocket PC platforms, using the C# framework. A simulation program, emulating the behaviour of the jet engine cotrol unit, was also implemented and used for testing.

Keywords

visualization, C#, Pocket PC, communication, protocol, small jet engine, turbine

Obsah

Seznam tabulek

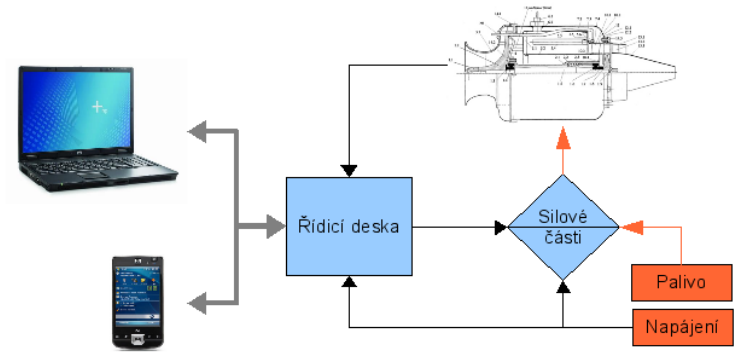
Seznam ilustrací

1. Úvod

Cílem práce je navrhnout a realizovat softwarové vybavení pro osobní počítač a mobilní zařízení typu Pocket PC[1], pomocí kterého by bylo možné komunikovat s řídicí jednotkou, diagnostikovat a řídit proudový motor pro model letadla.

Diplomová práce volně navazuje na bakalářskou práci stejného autora [1] a na bakalářskou práci Michala Voseckého – Návrh a realizace řídicí jednotky malého proudového motoru [4].

První část práce je věnována návrhu komunikačního protokolu. Zde je rozpracován formát a typy jednotlivých zpráv. Hlavní část diplomové práce se zabývá návrhem a implementací samotného softwarového produktu. Byly vytvořeny dva programy. Jeden určený pro osobní počítače s operačním systémem Windows a druhý pro mobilní zařízení s platformou Windows CE. Oba programy pak byly testovány pomocí serveru emulujícího chování řídicí jednotky, který byl pro toto použití speciálně vytvořen.



Ilustrace 1: Schéma zapojení

Ilustrace 2: Model nadzvukového proudového cvičného letounu Northrop T-38 Talon

1 Pocket PC (zkráceně též P/PC nebo PPC) je hardwarová specifikace pro kapesní počítače (PDA) s operačním systémem Microsoft Windows Mobile. Mohou být schopny používat alternativní operační systém, jako NetBSD nebo Linux.

2. Rozbor zadání

2.1. Použité termíny

API

Aplication Programming Interface [9]. Jedná se o rozhraní aplikace, které umožňuje využívat funkce aplikace v jiných aplikacích. Je to sbírka procedur, funkcí a tříd, které může programátor využít.

Framework

Softwarová struktura sloužící jako podpora při programování a vývoji jiných projektů. Obsahuje podpůrné programy, API, vzory a doporučení pro vývoj.

DLL

Dynamic Linking Library. Při linkování programu s dynamickou knihovnou se do výsledného spustitelného souboru ukládají pouze tabulky odkazů na symboly definované v dynamické knihovně. Pro chod programu je pak nutno kromě vlastního programu mít na počítači nainstalovánu i příslušnou dynamickou knihovnu.

Při spouštění programu pak operační systém provádí tzv. dynamické linkování. Během tohoto procesu operační systém načítá do operační paměti jak kód vlastního programu (spustitelný soubor) tak i kód dynamické knihovny, kterou program vyžaduje ke své činnosti.

XML

Extensible Markup Language [11] je univerzální specifikace pro vytváření vlastních značkových jazyků. Umožňuje uživateli definovat vlastní elementy a celkovou strukturu dat, tak aby odpovídala požadavkům na popis daného systému.

Serializace

Serializace [3] je proces ukládání objektů na jednotky pevných disků. Jiná část, nebo dokonce úplně samostatná aplikace pak může objekt deserializovat (obnovit do původního stavu). Objekt pak bude ve stejném stavu, v jakém se nacházel v okamžiku serializace.

Sestavení (assembly)

Sestavení [3] je logická jednotka obsahující přeložený (kompilovaný) kód určený pro platformu .NET Framework.

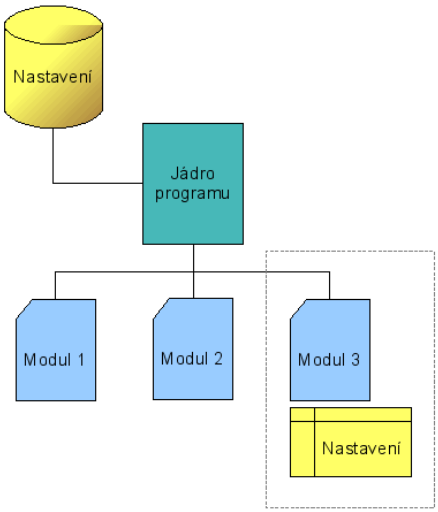
Jmenné prostory (namespaces)

Jmenné prostory [3] jsou způsobem, jak seskupit skupinu souvisejících tříd.

2.2. Analýza

Pro realizaci projektu bylo nejprve nutné vybrat komunikační port. Navrhnout a specifikovat komunikační protokol. Protokol měl splňovat požadavky na co nejmenší objem přenášených dat a určitou úroveň bezpečnosti pro dané použití.

Pro samotnou realizaci programů bylo nutné vytvořit několik základních knihoven, pomocí kterých „jádro" programu například komunikuje s řídicí jednotkou, pracuje se samotnými daty (správa, ukládání, načítaní), vykresluje grafy, zobrazuje další stavové informace řídicí desky atd. Tento modulární systém umožnil především jednodušší práci s jednotlivými požadovanými funkcemi programu, ale také se do budoucna ulehčí úpravy a další případná rozšíření.



Ilustrace 3: Model programu

Jednotlivé části, knihovny programu, jsou napojeny na již zmíněné jádro, které zpracovává příkazy od uživatele nebo řídící desky. Samotné moduly pak reagují pouze na události z jádra určené pouze jim nebo posílají žádosti jádru. Centralizovaný systém trochu ztížil modifikaci, ale ušetřil výpočetní čas. Například náročný rozklad paketů z řídicí desky probíhá pouze v jednom místě a moduly dostávají informaci určenou pouze pro ně. Modul na vykreslení grafů zase pouze žádá o určitý rozsah dat podle nastavení, nebo podle stavu interního časovače překreslení. Následně data sám zpracovává a zobrazuje jako časový průběh.

Tento návrh je použit jak u programu pro osobní počítač, tak pro Pocket PC.

2.3. Řídicí jednotka

Vývojem řídicí jednotky se zabýval Michal Vosecký ve své bakalářské práci. Pro řízení je použita deska SpejblARM s 32b mikrokontrolérem LPC2119 s implementací základních nízkoúrovňových funkcí. Firmware pro řídící jednotku je v současné době ve vývoji.

Pro testovací účely byl tak vytvořen program (7) emulující všechny potřebné funkce.

Do určité míry využívá stejných knihoven designovaných pro samotný program. Například pro komunikaci.

2.4. Vývojové prostředí

Pro tvorbu programů pro mobilní zařízení s platformou Microsoft Windows CE se využívá nejčastěji programovacího jazyka Visual C++ nebo C#.

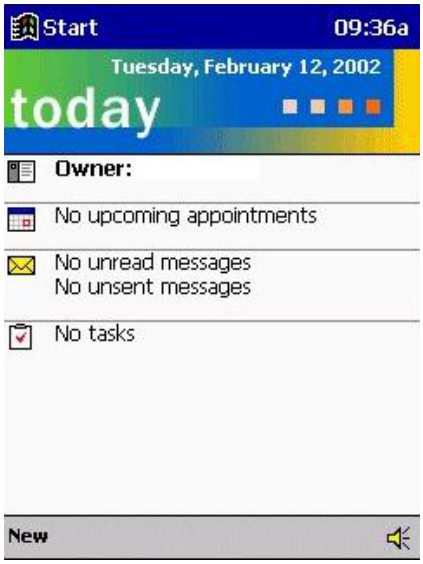
Pro tvorbu všech programů bylo vybráno Microsoft Visual Studio 2005 a jazyk C#. Programovací jazyk C# splňuje moderní nároky na objektově orientované programování a .NET Framework[2] je distribuován i jako Compact verze do mobilních zařízení[3].

Microsoft .NET Compact Framework používá některé shodné knihovny tříd jako klasický .NET Framework a také několik knihoven designovaných speciálně pro mobilní zařízení. Jako je třeba Windows CE InputPanel.

3. Specifikace prostředí Windows Mobile

3.1. Historie

6. ledna roku 2000 byla na světovém veletrhu CES (Consumer Electronics Show) představena první verze s kódovým jménem Rapier. Windows CE 3.0 byly určeny pro kapesní počítače a obsahovaly Windows Media Player a čtečku elektronických knih MS Reader s podporou vyhlazování písma ClearType. Jediné podporované rozlišení bylo QVGA[4].



Ilustrace 4: Pocket PC 2000

Mezi první firmy, které implementovaly do svých zařízení nový operační systém bylo HP a Casio. [10]

Od té doby bylo vydáno pět základních verzí. Každá modifikace, jako GSM modul, příchod chytrých telefonů (Smartphone) si totiž vyžádal novou základní verzi.

Pocket PC 2002 zavedla personifikaci „úvodní" stránky Dnes, textový editor Word, vylepšený Pocket Internet Explorer atd. Objevila se také verze Smartphone 2002. V nabídce Start jsou počínaje verzí Windows Mobile 2003 Second Edition neustále zobrazovány nejčastěji spouštěné aplikace a jsou tak uživateli snadněji přístupny. Ve verzi Pocket PC umožňuje také rotaci displeje.

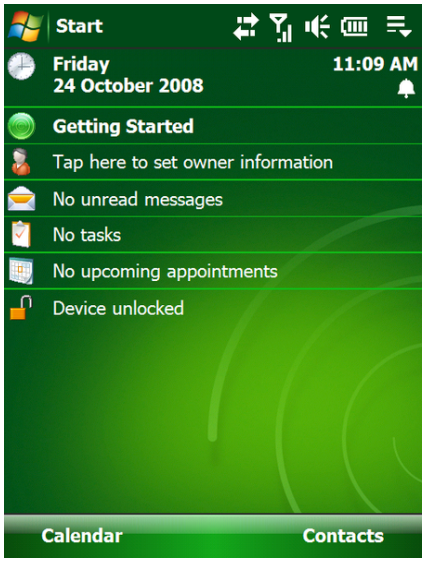
Windows Mobile 5, které byly představeny 9. května 2005 v Las Vegas pohání již

2 První verze 1.0 RTM byla vydána ve druhé polovině roku 2002, zatím poslední verze 3.5 byla vydána 25. ledna 2008
3 Oficiální stránky Windows Mobile http://www.microsoft.com/windowsmobile/
4 320x240 Quarter VGA (Video Graphics Array)

Windows CE 5.0. Kromě vylepšení a nových verzí programů, USB 2.0, přináší především změnu práce s pamětí. Data se ukládají do FlashROM a RAM je určena pouze pro běh aplikací. Nehrozí tak ztráta dat i při vybití záložní baterie.

Windows Mobile 6, známý dříve pod kódovým označením Crossbow, je nejnovější verze tohoto mobilního operačního systému, která byla představena 12. února 2007 na 3GSM World Congress 2007. Existují celkem tři verze systému: Windows Mobile 6 Standard pro Smartphony Windows Mobile 6 Professional pro PDA s integrovaným telefonem (Pocket PC Phone Edition) a Windows Mobile 6 Classic pro PDA bez telefonního modulu. Windows Mobile 6 je úzce propojen s Windows Live a Exchange 2007 a podporují až rozlišení 800x480 (WVGA).

Společnost Microsoft nyní vyvíjí nástupce Windows Mobile 6.0 (6.1) pod kódovým označením Photon, který bude uveden na trh nejdříve v roce 2009.



Ilustrace 5: Windows Mobile 6.1 Professional - obrazovka Dnes

Ilustrace 6: Windows Mobile 6.1 Standard - obrazovka Dnes

3.2. Instalace programů

Programy jsou do mobilního zařízení instalovány přes Microsoft ActiveSync (4.1.2), který obsahuje správce nainstalovaných programů. Program je vždy zkopírován do adresáře programu ActiveSync a odtud je kopírován do zařízení. Přes správce je pak možné vždy odebrat zvolený program ze zařízení a později jej zase nainstalovat, popřípadě počkat na nejbližší připojení zařízení k počítači, kdy se synchronizují i instalace programů.

Instalaci je také možné provádět pomocí balíčků CAB přímo nakopírováním do zařízení a spuštěním instalace (9.2).

4. Windows Mobile Development

Pro tvorbu programů slouží takzvané vývojové kity (*Software Development Kits*, dále jen SDKs). Jednotlivé SDK obsahují všechny potřebné knihovny a jejich součástí je i emulátor pro dané zařízení.

Jednotlivé SDK pro Windows Mobile znázorňuje následující tabulka. Od verze 6 Microsoft zavádí nové pojmenování (3.1). Dřívější řadu, například Windows Mobile 2003 pro Pocket PC Phone Edition tak nyní najdeme pod pojmenováním Windows Mobile 6 Professional SDK.

Staré pojmenování	**Nové pojmenování**	**SDK**
Windows Mobile for Smartphone	Windows Mobile Standard	Windows Mobile 6 Standard SDK
Windows Mobile for Pocket PC	Windows Mobile Classic	Windows Mobile 6 Professional SDK
Windows Mobile for Pocket PC Phone Edition	Windows Mobile Professional	

Tabulka 1: Přejmenování Windows Mobile SDKs

4.1. Minimální požadavky

Pro tvorbu programů pro zařízení Windows Mobile 6 je nutné mít minimálně Microsoft Visual Studio 2005 a Service Pack 1. Neplacená Express edice Visual Studia není podporována.

Dále je doporučováno nainstalovat nejnovější verzi Microsoft .NET Compact Frameworku, Microsoft ActiveSync. (popř. Windows XP Service Pack 2)

Všechny potřebné programy je možné stáhnout ze stránek Microsoft Download Center [5].

4.1.1. Microsoft .NET Compact Framework v2 SP2

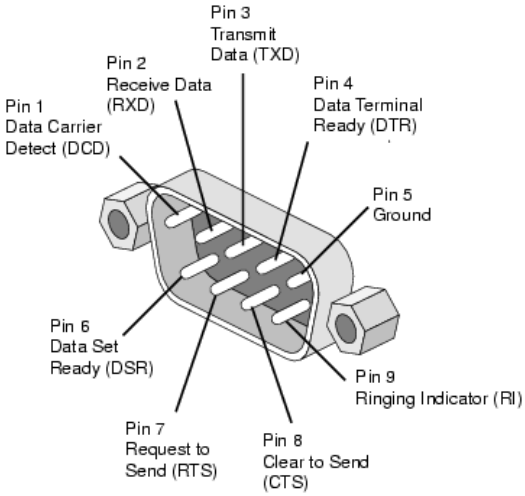
Obsahuje .NET Framework ve verzi 2. Pro instalaci balíku je nutné nejdříve odinstalovat stávající Compact Framework nainstalovaný například jako součást Visual Studia 2005. Po kompletní instalaci je nabídnuta instalace i do mobilního zařízení přes Microsoft ActiveSync.

4.1.2. Microsoft ActiveSync

Program pro synchronizaci s počítačem či serverem a instalaci nových programů do Windows Mobile. Od verze Windows Vista je ActiveSync nahrazen novým systémem Windows Mobile Device Center[5]. Aktuální verze pro rok 2009 je 6.1.

5 Windows Mobile Device Center
http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-center.mspx

5. Port



Ilustrace 7: Konektor DB-9/DE-9

K přenosu dat se využívá sériového portu, nebo portu USB.

Při zapojení přes sériovou linku tok dat není softwarově řízen, takže se k přenosu využívá pouze pinu 2 (příchozí data), pinu 3 (odchozí data) a pinu 5 (GND). Datové kanály se pouze překříží.

K zapojení přes mini-USB u Pocket PC se využívá obou sériových kanálů a tok dat obstarává USB modul a API Windows CE. Následující tabulka ukazuje rozsah nastavení portu.

Parametr	**Rozsah hodnot**
Počet datových bitů	8
Rychlost	9600 – 256000 b
Parita	žádná
Stop bit	1
Řízení toku	žádné

Tabulka 2: Nastavení portu

6. Komunikační protokol

6.1. Popis

Komunikační protokol prošel několika změnami až do současné verze, která je optimalizována pro dané potřeby a možnosti komunikujících zařízení. Minimalizovala se redundance dat starajících se o bezchybný přenos.

Komunikační protokol popisuje přenos dat, komunikaci mezi softwarovým vybavením osobního počítače, Pocket PC a řídicí jednotkou pro řízení proudového motoru. Komunikace probíhá po sériové lince RS-232 (PC), která je popřípadě emulována na PC rozhraním USB. Při použití softwaru v Pocket PC je využíván mini-USB port.

Data jsou posílána ve formátu big-endian. Na paměťové místo s nejnižší adresou se uloží nejvíce významný bajt (MSB) a za něj se ukládají ostatní bajty až po nejméně významný bajt (LSB) na konci.

Zpráva je posílána jako posloupnost znaků, kde čísla jsou reprezentována jako řetězec hexadecimálních čísel o různé délce. Jednotlivé zprávy jsou rozděleny do několika skupin. Každé skupině zpráv a databloku v nich obsažených přísluší jednoznačný jednoznakový identifikátor.

6.2. Klíčová slova

Paket, zpráva

Posílaná posloupnost znaků. Začíná identifikátorem pro začátek a končí paritním znakem.

Tělo zprávy

Část paketu obsahující počet znaků v databloku a samotný datablok.

Datablok

V hlavičce obsahuje unikátní identifikátor (id) zprávy a za ním následují samotná data ve formátu odpovídající id a typu zprávy.

6.3. Protokol

6.3.1. Obecný formát zpráv

Samotný řetězec znaků se skládá ze zprávy začínající kontrolním start bajtem, následují znaky reprezentující délku zprávy, id zprávy a samotná data. Vše je zakončeno paritním znakem. K výpočtu paritního znaku se počítá příčná parita přes všechny znaky (6.3.12).

```
<paket>
<start byte> <tělo zprávy> <XOR>
<start byte> <délka zprávy><datablok> <XOR>
<start byte> <délka zprávy><id zprávy><data> <XOR>
```

6.3.2. Použité znaky

6.3.2.a Start byte

Každá posílaná zpráva musí začínat tímto znakem. Slouží k identifikaci a na straně PC k synchronizaci komunikace.

Velikost	Znak	Popis
1	@	Úvodní znak

Tabulka 3: Start byte

6.3.2.b Délka zprávy

Určuje délku samotných dat.

Velikost	Znak	Popis
2		Počet dat v databloku včetně paritního bajtu

Tabulka 4: Dálka zprávy

6.3.2.c Id zprávy

Samotné textové řetězce jsou identifikovány podle znaku id. Tím se dělí na zprávy, příkazy a potvrzující zprávy.

Zprávy mají vlastní identifikátor za kterým následuje posloupnost dat odpovídající dané zprávě.

Příkazy začínají znakem C (Command), za kterým je znak daného příkazu. Příkazy neobsahují žádný blok dat a jejich délka je tak konstantní: 3 (do délky se počítá i paritní bajt). Následující tabulka představuje seznam všech podporovaných povelů, jejichž význam bude podrobněji popsán v kapitole 6.3.4.

Velikost	Znak	Popis
1	V	Data
1	C	Příkaz. Viz. tabulka příkazů v kapitole 6.3.4
1	R	Regulátor
1	E	Error
1	O	Ok
1	J	Provede skok o definované výšce
1	U	Rampa nahoru
1	D	Rampa dolů
1	P	Nastaví vzorkovací periodu
1	S	Otáčky

Tabulka 5: Příkazy

6.3.2.d Paritní znak

Kontrolní znak vypočtený přes celou zprávu a přidaný na konec paketu. Pro výpočet se používá funkce XOR.

Velikost	Znak	Popis
1		XOR

Tabulka 6: XOR

6.3.3. Směr zpráv / příkazů

PC	»»»»»»	««««««	Řídicí jednotka
Zpráva	**Odpověď**	**Odpověď**	**Zpráva**
			Z Data
Z Regulátor		Ok / Error	
Z Otáčky		Ok / Error	
Z Skok		Ok / Error	
Z Rampa		Ok / Error	
Z Vzorkovací perioda		Ok / Error	
P Regulátor		**Z** Regulátor	
P Program		Ok / Error	
P Manuál		Ok / Error	
P Stop		Ok / Error	
P Vzorkovací perioda		**Z** Vzorkovací perioda	

Tabulka 7: Směr zpráv / příkazů

* Z – zpráva, P – příkaz

6.3.4. Příkazy

<**kód**>

@	0	3	C	Id	XOR

Délka

3

Popis

Příkazy jsou pakety, které obsahují pouze id zprávy (znak C), po kterém následuje jeden znak reprezentující daný příkaz. Příkaz tak neobsahuje blok data.

V následující tabulce je seznam všech použitých příkazů.

Příkaz	Kód	Popis
regulátor	R	Řídicí jednotka pošle aktuální nastavení regulátoru
program	A	Předá řízení programu
manuál	B	Přepne řízení na manuál
stop	C	Zastaví turbínu
vzorkovací perioda	S	Odešle do pc nastavenou vzorkovací periodu

Tabulka 8: Tabulka příkazů

6.3.5. Zprávy

<id dat><napětí na čerpadle><otáčky><teplota>

@	2	2	V	Id	P	S	T	XOR

Délka

34

Formát

Znak	Formát		Popis
Id	int32	-	Číslo vzorku
P	int32	[mV]	Napětí na čerpadle
S	int32	[ot/min]	Otáčky rotoru
T	int32	[°C]	Teplota

Tabulka 9: Formát zpráv

Popis

Na prvním místě je úvodní znak, následuje znak, který identifikuje datablok jako data. Délka zprávy je 34 (22h). Id odpovídá číslu vzorku a jeho hodnota je inkrementována s každým následujícím vzorkem.

Posloupnost po sobě jdoucích vzorků se žádným algoritmem nekontroluje. Pokud se nějaký vzorek telemetrických dat například ztratí, nebo neodpovídá parita, nevyžaduje se opakované poslání daného vzorku. Číslo vzorku je určeno pouze pro kontrolu kontinuity naměřených dat.

6.3.6. Nastavení regulátoru

<parametry><limity>

@	C	2	R	k	w_I	w_D	b	c	N	I_L	I_H	D_L	D_H	O_L	O_H	XOR

Délka

194

Formát

Znak	Formát		Popis
k	double	-	Zesílení
w_I	double	-	Zlomová frekvence integračního členu
w_D	double	-	Zlomová frekvence derivačního členu
b	double	-	Váhový koeficient pro nelineární člen P složky
c	double	-	Váhový koeficient pro nelineární člen D složky
N	double	-	Zlomová frekvence dolní propusti pro derivační složku
I_L	double	-	Spodní omezení integrátoru
I_H	double	-	Horní omezení integrátoru
D_L	double	-	Spodní omezení derivačního členu
D_H	double	-	Horní omezení derivačního členu
O_L	double	-	Spodní omezení výstupu
O_H	double	-	Horní omezení výstupu

Tabulka 10: Nastavení regulátoru - formát dat

Popis

Formát paketu popisuje parametry regulátoru [2].

6.3.7. Vzorkovací perioda

`<násobitel>`

@	0	A	P	M	XOR

Délka

10

Formát

Znak	Formát	Popis
M	int32	Násobitel základní vzorkovací periody procesoru

Tabulka 11: Vzorkovací perioda - formát dat

Hodnoty

Násobitel	Vzorkovací perioda [ms]
1	16
2	32
3	48
4	64
...	...

Tabulka 12: Vzorkovací perioda - hodnoty

6.3.8. Otáčky

<otáčky>

@	0	A	S	X	XOR

Délka

10

Formát

Znak	Formát	Popis
X	int32	Otáčky rotoru

Tabulka 13: Otáčky - formát dat

Popis

Například pro nastavení volnoběžných otáček. Turbína přejde plynule na danou hodnotu otáček (v daných mezích).

6.3.9. Skok

<napětí>

@	0	A	J	U	XOR

Délka

10

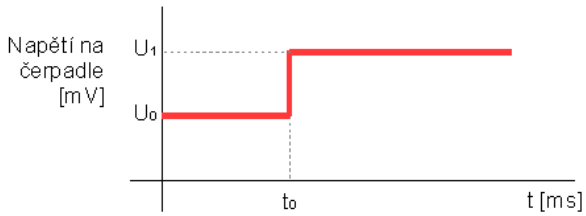
Formát

Znak	Formát	Popis
U	int32	Napětí v mV

Tabulka 14: Skok - formát dat

Popis

Po zpracování tohoto příkazu, řídicí jednotka vygeneruje napěťový skok na palivovém čerpadle a tak zvýší otáčky rotoru.



Ilustrace 8: Skok

6.3.10. Rampa nahoru a dolu

<počet kroků><výška jednoho kroku><čas jednoho kroku>

@	1	A	U	C	H	T	XOR

Délka

26

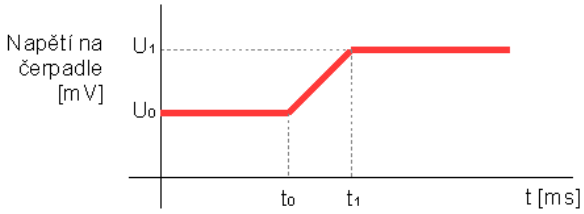
Formát

Znak	Formát	Popis
C	int32	Počet kroků
H	int32	Výška jednoho kroku [mV]
T	int32	Délka jednoho kroku [ms]

Tabulka 15: Rampa nahoru a dolu - formát dat

Popis

Po přijetí tohoto příkazu řídicí jednotka začne generovat rampu napětí na palivovém čerpadle směrem nahoru (U), popřípadě dolů (D) definované výšky a délky. Výška je dána jako součin hodnot C a H – rozdíl napětí U_1 a U_0. Druhý rozměr, délka, je dána součinem hodnot C a T – časový interval t_1 - t_0.



Ilustrace 9: Rampa nahoru

6.3.11. Potvrzující a chybové zprávy

Potvrzující zprávy se posílají:

- pouze z řídící desky

Potvrzují se:

- příkazy
- nastavení regulátoru

Při příchodu příkazu, nebo zprávy obsahující informaci o nastavení regulátoru řídicí jednotka zpracuje data a potvrdí jejich bezchybné vykonání. Z jednotky se posílá zpráva obsahující pouze id zpracovávaného příkazu.

Chybové zprávy se posílají:

- pokud nastala chyba – podle tabulky chyb

6.3.11.a Potvrzující zprávy

<id potvrzované zprávy>

@	0	3	O	Id	XOR

Délka

3

Formát

Znak	Formát	Popis
Id	char	Id zprávy, která má být potvrzena. Např. pro regulátor je Id „R“

Příklad

Pokud je vyžadována potvrzující zpráva, měla by být do určitého časového limitu doručena zpět straně, která ji vyžádala. Pokud se tak nestane, nastává opakování podle nastavení programu v pc. Pokus se několikrát opakuje a při neúspěchu je informován uživatel.

6.3.11.b Chybové zprávy

<id>

@	0	A	E	Id	XOR

Délka

10

Chybové zprávy se řídí podle tabulky chyb.

6.3.11.c Tabulka chyb

Tabulka chyb nebyla zatím specifikována jelikož řídicí jednotka je stále ve vývoji. Chybové zprávy tak zatím pouze doplňují potvrzující zprávy jako jejich opak a jejich využití ukáže až čas.

6.3.12. Zabezpečení zpráv - XOR

<zpráva><XOR>

Kontrolní příčná sudá parita se počítá jako exkluzivní OR. Vždy přes celou zprávu.

Příklad

@	2	2	V	Id	P	S	T	XOR

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 16: Funkce XOR

6.3.13. Formát dat

6.3.13.a Int32

Délka čísla *int32* je 4B. Číslo je tak zakódováno jako 8 znaků

6.3.13.b Double

Hodnota *double* má délku 8B. Hodnota je tak reprezentována jako posloupnost 16 znaků.

6.3.13.c Char

Představuje jeden libovolný znak.

6.3.14. Příklad zprávy

6.3.14.a Data

```
@22V0000000A0000157C000151E400000282k
```

Paket se dá rozložit na jednotlivé části podle příslušného formátu, který definuje čtvrtý znak

```
@ | 22 | V | 0000000A | 0000157C | 000151E4 | 00000282 | k
```

a dále dekódovat:

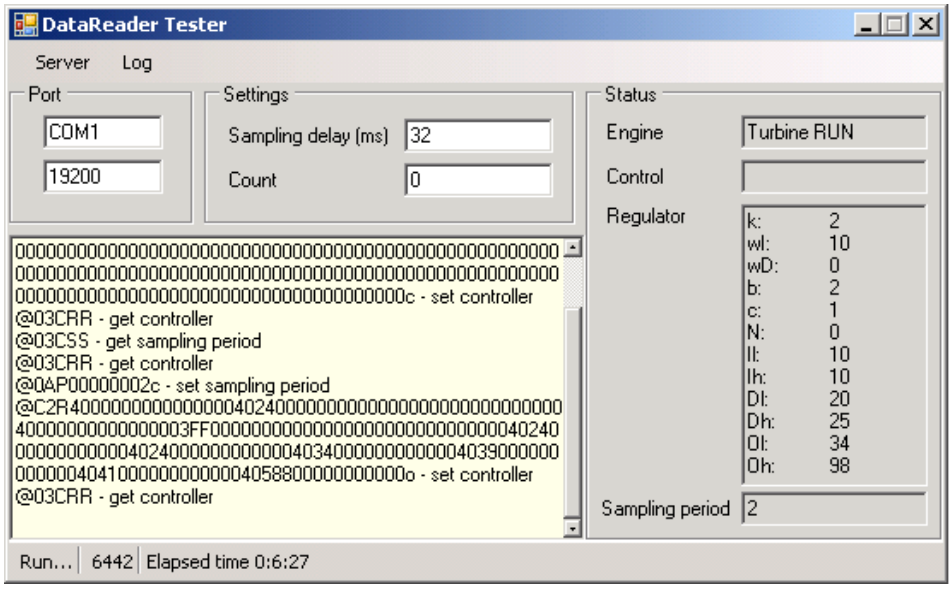
Délka:	22h odpovídá 34 znaků
Id zprávy:	V = data
Id (číslo) vzorku:	A odpovídá číslu vzorku 10
Napětí na čerpadle:	5500mV
Otáčky:	86500ot/min
Teplota:	642°C
Parita:	znak k

7. Program DataReader Tester

7.1. Popis

K testovacím účelům byl vytvořen program emulující chování řídicí desky. Zdrojové kódy a instalace je na přiloženém cd. Instalací provází průvodce.

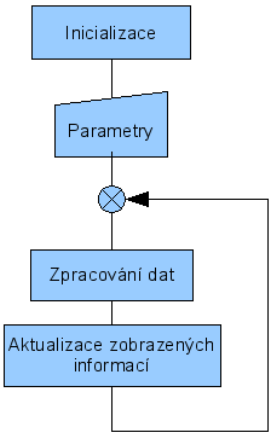
Komunikace se nastavuje v části Port, kde se vyplní jméno a přenosová rychlost portu. Hodnota *Sampling delay* vymezuje časovou prodlevu mezi jednotlivými odeslanými vzorky a pokud je nastavena hodnota *Count*, program odešle pouze požadovaný počet hodnot. V pravé části programu se nachází informační panel, který představuje aktuální nastavení řídicí desky. Ve spodní částí je pak logovací okno, kde se zobrazují přijaté pakety v textové podobě s textovým popiskem.



Ilustrace 10: DataReader Tester

7.2. Specifikace

Po inicializaci a bezchybném otevření komunikačního kanálu se vytvoří vlákno, které začne odesílat vzorky představující telemetrická data. Hodnoty otáček rotoru a teploty výfukových plynů jsou generovány jako obdélníkový průběh. Hodnota napětí je generována jako šum v rozmezí 1 až 7 voltů.



Ilustrace 11: DataReader Tester - vývojový diagram

Program dále reaguje na příchod a rozpoznání datového paketu. Po příchodu, identifikaci je vyvolána událost `DataAvailableEvent`. Data obsažená v příchozím paketu jsou zpracována. Pokud paket obsahoval žádost o data nebo se jednalo o paket, který má být potvrzen, sestaví se nový paket a vloží se mezi pakety telemetrických dat.

8. Program DataReader

Za tímto účelem bylo nutné navrhnout a implementovat níže popisované části programů, které jsou kompilovány jako dynamické knihovny (2.1), které pak spojuje spustitelný soubor sestavení DataReader.

Následující část práce je rozdělena na popis programu pro osobní počítač (DataReader) a verzi pro mobilní zařízení (DataReader6Professional).

8.1. Verze pro osobní počítač

8.1.1. Popis

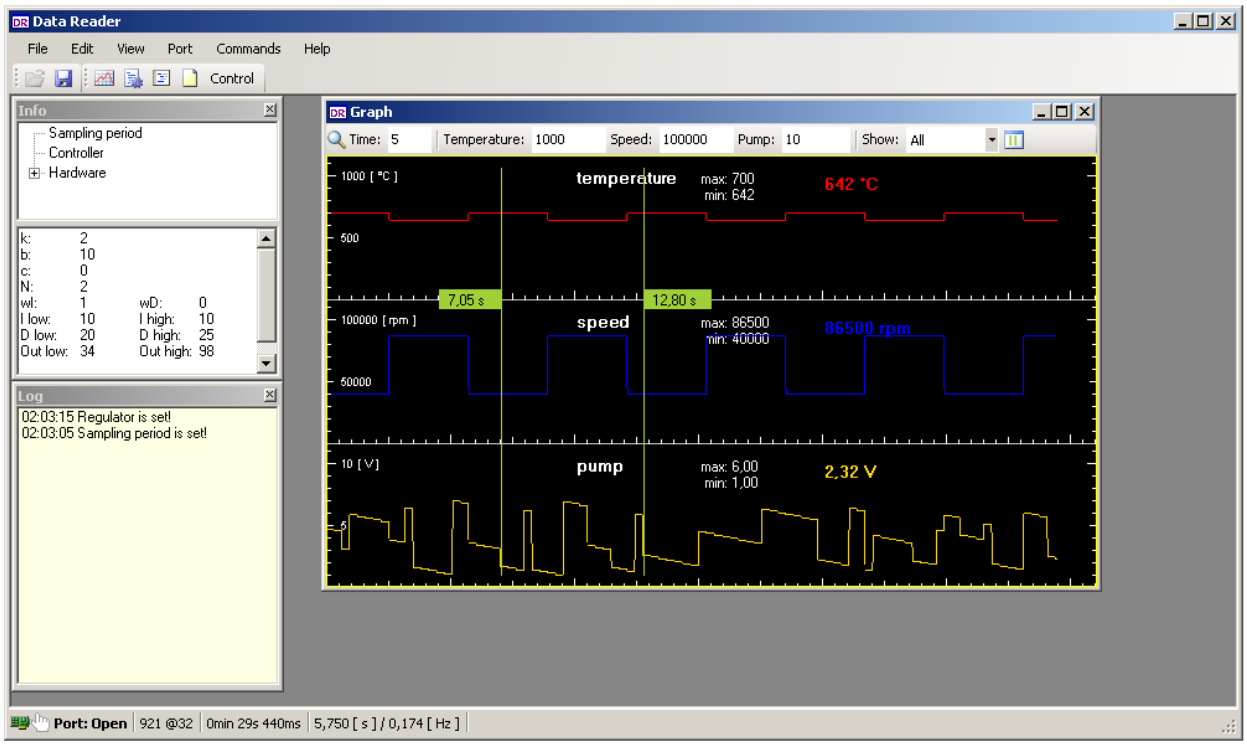
Program slouží k diagnostice a řízení turbíny. Telemetrická data jsou zobrazována na grafu, který je rozdělen na tři části. Každé části přísluší jedna přenášená veličina měřená na turbíně. Jsou to teplota výfukových plynů, otáčky rotoru a napětí na palivovém čerpadle.

Vedle těchto telemetrických dat komunikuje software s jednotkou pomocí řídicích příkazů, které umožňují aktualizovat nastavení parametrů regulátoru turbíny, intervalu odečítání dat ze senzorů, nastavovat volnoběžné otáčky a zobrazovat další stavové veličiny.

Při připojení programu a nastavení parametrů portu program automaticky detekuje komunikaci, odešle nastavovací pakety a začne pracovat.

Data jsou průběžně ukládána do souboru a poslední řada dat potřebná k vykreslení se zpracovává a zobrazuje v okně grafu. Nepotřebná data jsou z datové struktury automaticky mazána. Počet vzorků, které jsou ukládány do temp souboru, se nastavuje v hlavním nastavení programu. Tímto způsobem se udržuje minimální spotřeba operační paměti, zvyšuje se rychlost odezvy programu a zároveň se zálohují odečítané vzorky.

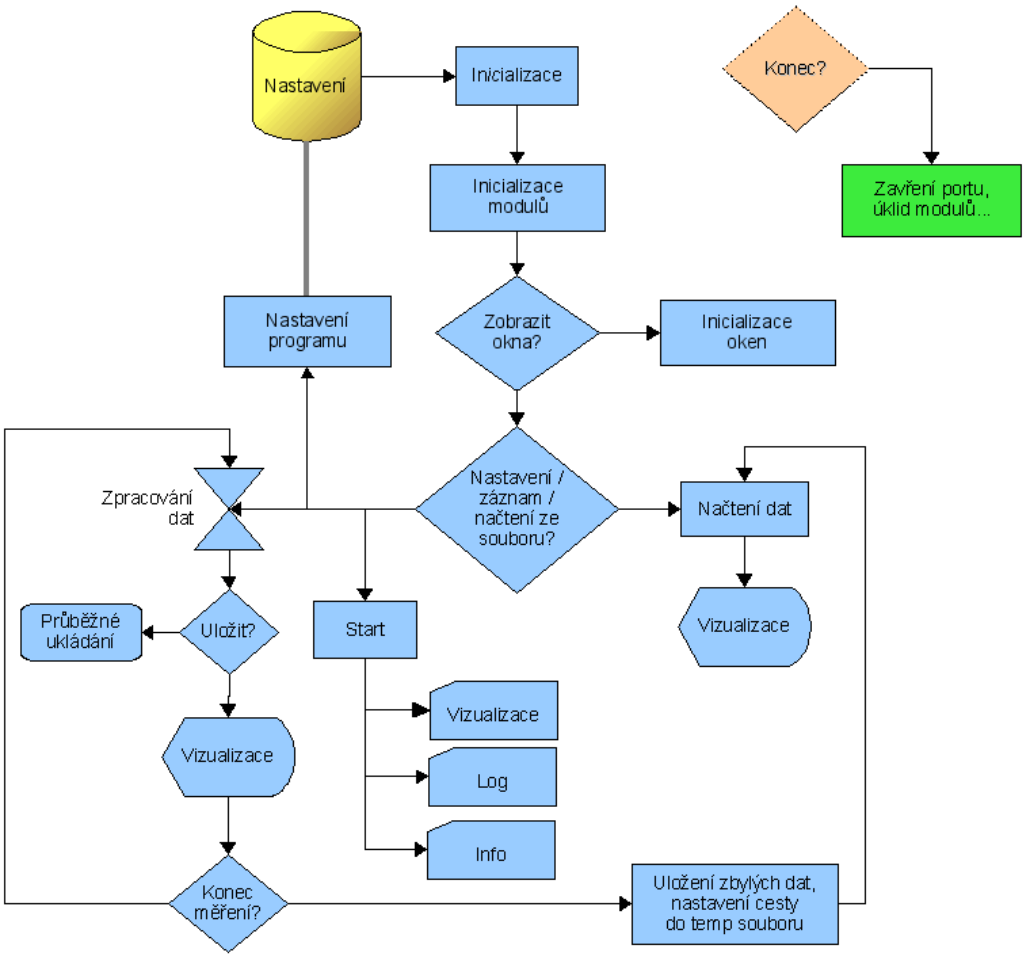
Data je možné v průběhu chodu programu ukládat do souborů, nebo program odpojit od řídicí desky a zobrazovat uložená data. K ukládání dat byl zvolen soubor typu csv [10]. Jednotlivé hodnoty jsou odděleny středníkem a každá skupina vzorků je na novém řádku. Data je tak možné zpracovávat v jakémkoliv matematickém programu typu Matlab.



Ilustrace 12: DataReader

Na obrázku 12 je program zobrazen s několika informačními okny. Hlavní okno je zároveň jádro programu a představuje rodiče pro jednotlivá dílčí okna. Některá z nich pak implementují některé z modulů (například okno *Graph*). Okno *Info* pak podle vybrané položky odesílá požadavek do řídicí desky a přijaté informace zobrazuje. Okno *Log* zobrazuje například informace o změně připojení, úspěšném nastavení regulátoru atd.

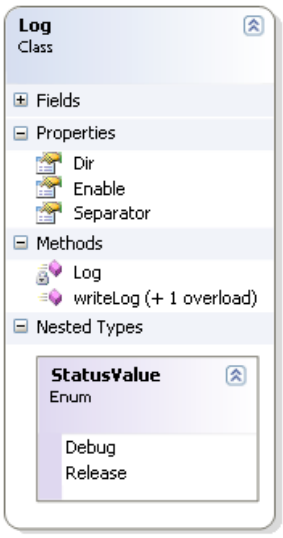
8.1.2. Specifikace



Ilustrace 13: DataReader - vývojový diagram

Následující část je věnována samotným komponentám programu. Části jsou děleny podle jmen projektů, ze kterých je složeno celé sestavení.

8.1.2.a Log4App



Ilustrace 14: Třída Log

Třída `Log` se nachází ve jmenném prostoru `Log4App` stejnojmenného projektu. Jedná se o podpůrnou třídu zajišťující zápis logů do souboru. Slouží především k identifikaci chyb v programu, které se vyskytují náhodně, nebo v místech, kde je je špatné odladit.

Pro správnou funkci objektu stačí nastavit adresář pro soubory, oddělovač mezi jednotlivým spuštěním programu a typ záznamu – Debug, nebo Release.

Soubory jsou ukládány do složky Log v adresáři projektu a jsou pojmenovávány podle aktuálního data ve formátu `RRRR.MM.DD`.

Knihovnu `Log4App` je možné využít v jakémkoliv jiném projektu.

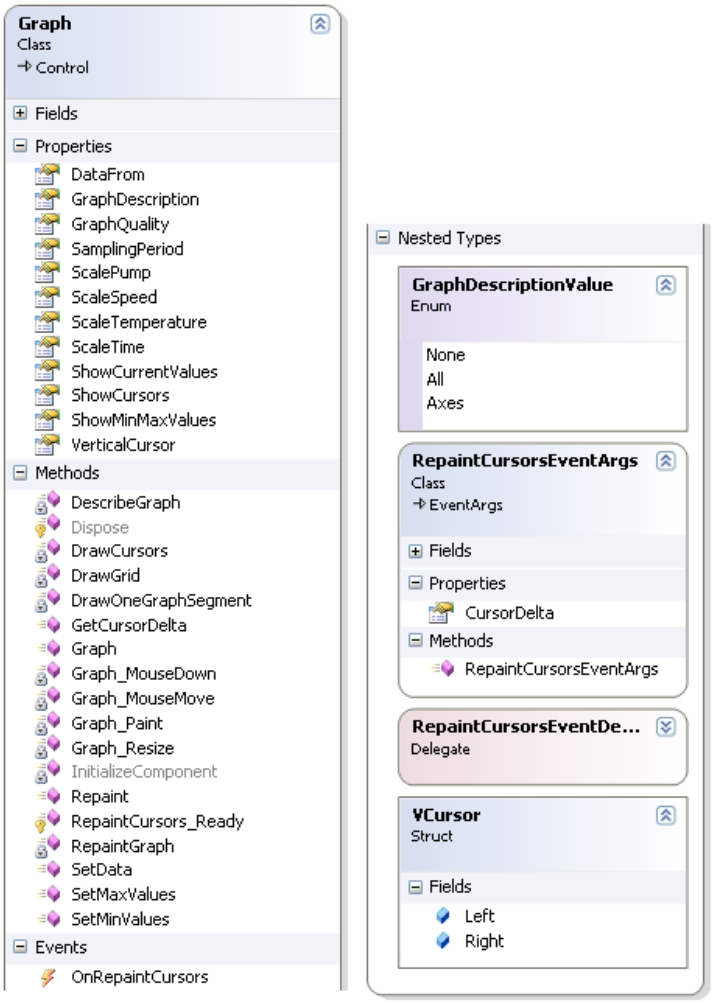
Příklad výpisu

```
2008.12.23 12:55:29 >> Repaint canvas error.
                    >> System.ObjectDisposedException: K uvolněnému objektu
nelze přistupovat.
      Název objektu: Graph
         v System.Windows.Forms.Control.CreateHandle()
         v System.Windows.Forms.Control.get_Handle()
         v System.Windows.Forms.Control.CreateGraphicsInternal()
         v System.Windows.Forms.Control.CreateGraphics()
         v DataReader.Visualization.Graphs.Graph3.Graph.Repaint() v
C:\Dokumenty\Visual Studio 2005\Projects\DataReader\Drawing\Graph.cs:řádek
593

2008.12.23 12:55:29 >> Plot graph.
                    >> System.ObjectDisposedException: K uvolněnému objektu
nelze přistupovat.
      Název objektu: Graph
         v DataReader.Visualization.Graphs.Graph3.Graph.Repaint() v
C:\Dokumenty\Visual Studio 2005\Projects\DataReader\Drawing\Graph.cs:řádek
599
         v DataReader.Forms.FormGraph.RepaintGraph() v c:\Dokumenty\Visual
Studio 2005\Projects\DataReader\DataReader\Forms\FormGraph.cs:řádek 258
```

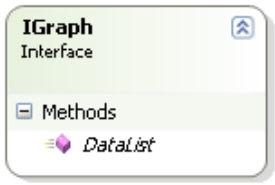
8.1.2.b Drawing

Projekt Drawing slučuje všechny komponenty spadající pod grafické prostředí. Zatím obsahuje pouze třídu `Graph`, která je potomkem třídy `Control`.

Třída `Graph` se nachází ve jmenném prostoru `DataReader.Visualization.Graphs.Graph3`. Komponentu najdeme také v záložce *Toolbox (View - Toolbox)* a některá její nastavení jsou definována tak, že jejich hodnoty je možné upravovat přímo přes panel *Properties (View – Properties Window)*.



Ilustrace 15: Třída Graph

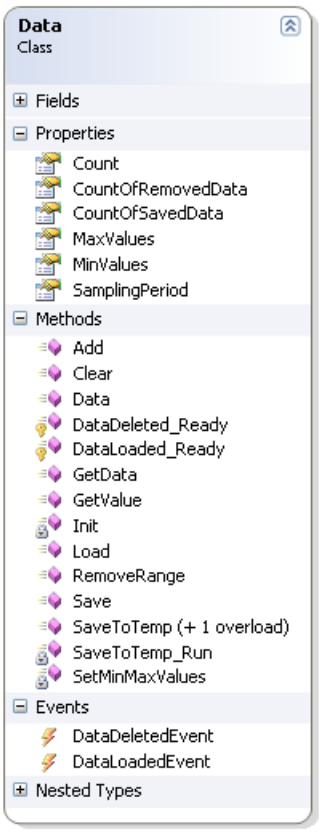


Ilustrace 16: Interface IGraph

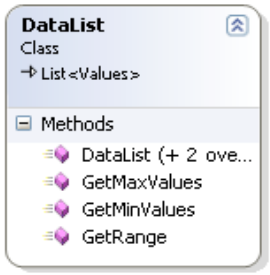
Následující kód ukazuje použití komponenty `Graph` s nastavením základních parametrů.

```
DataList data = new DataList(3);
data.Add(new Values(1000, 50000, 2500));
data.Add(new Values(1050, 50100, 2700));
data.Add(new Values(1120, 50600, 2850));

Graph graph = new DataReader.Visualization.Graphs.Graph3.Graph();
graph.ScaleSpeed = 100000;
graph.ScaleTemperature = 1000;
graph.ScalePump = 5;
graph.ScaleTime = 5;
graph.GraphQuality = System.Drawing.Drawing2D.SmoothingMode.HighSpeed;
graph.GraphDescription = Graph.GraphDescriptionValue.All;
graph.SetData(data);
graph.SetMaxValues(data.GetMaxValues());
graph.SetMinValues(data.GetMinValues());
graph.Repaint();
```

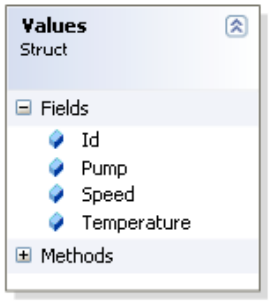
8.1.2.c DataAdapter



Ilustrace 17: Třída Data



Ilustrace 18: Třída DataList



Ilustrace 19: Struktura Values

Projekt data se skládá z tříd `Data`, `DataList` a struktury `Values` a nachází se ve jmenném prostoru `DataReader.DataAdapter`.

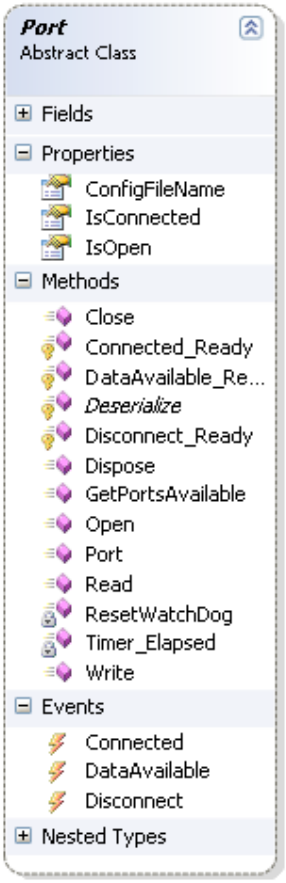
Třída `Data` zastřešuje všechny operace s daty. Všechny důležité vlastnosti (Properties), funkce (Methods) a události (Events), které objekt `Data` může generovat jsou patrné z obrázku.

Třída data obsahuje jako datový sklad třídu `DataList`, která je potomkem generické třídy `List<>`. Do kolekce je možné ukládat pouze data ve formátu `Values`. `Values` je jednoduchá struktura obsahující potřebné proměnné.

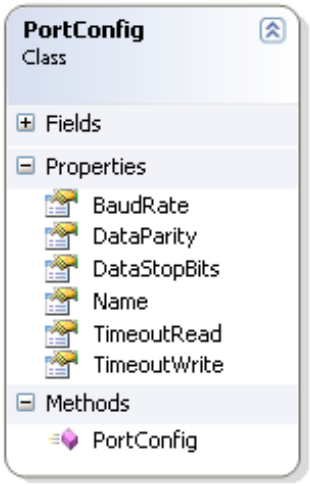
Celé datové úložiště je pak možné jednoduše rozšířit o další měřené hodnoty. Stačí upravit strukturu `Values`, modifikovat třídu `Data`, nebo definovat nové operace nad polem hodnot v třídě `DataList`.

8.1.2.d Interface

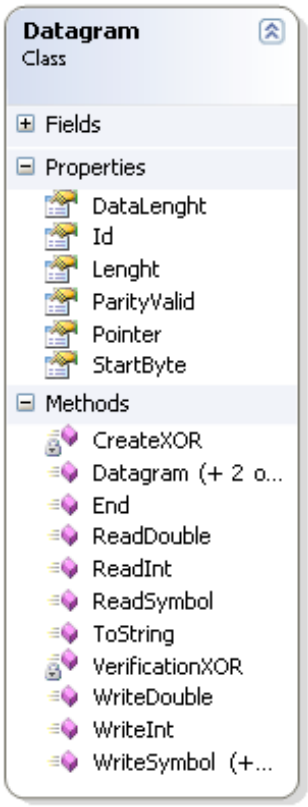
Třída obsluhující komunikaci po sériovém portu se nachází ve jmenném prostoru `DataReader.IO.PortSerial` projektu Interface.



Ilustrace 20: Třída Port



Ilustrace 21: Třída PortConfig



Ilustrace 22: Třída Datagram

Třída `Port` vedle základních funkcí jako je `Open()`, `Close()` obsahuje ještě několik dalších užitečných funkcí, které jsou patrné z obrázku. Funkce `Connected_Ready()` vyvolává událost `Connected`. Událost je tak vyvolána, pokud dojde ke spárování s řídicí jednotkou – detekci paketů. K události `Disconnect()` patří funkce `Disconnect_Ready`. Odpojení od desky zajišťuje navíc jeden časovač, který určuje časovou prodlevu. Ta je určena konstantou `WATCH_DOG_INTERVAL`.

Konfigurace portu je uložena v třídě `PortConfig`. Nastavení portu je serializované (2.1) do souboru definovaném v konstruktoru třídy `Port`.

```
// Default config file name
this.configFileName = @"Port.dr";
```

Nastavení portu tak zůstává uloženo a při dalším spuštění a existenci souboru je třída `Port` deserializována.

Jmenný prostor `DataReader.IO.PortSerial` obsahuje ještě třídu `Datagram`. Ta představuje data posílaná z řídící desky nebo do ní.

Třída `Datagram` obsahuje například vlastnost `Id`, což je první znak identifikující datagram. Znak je kontrolován s konstantou `START_BYTE`, jejíž úpravou je možné změnit první kontrolu datagramů. `ParityValid` nabývá hodnot `true` nebo `false`, podle toho zda vypočtená hodnota XOR odpovídá poslednímu znaku datagramu. Třída `Datagram` pak obsahuje několik funkcí pro zápis a čtení hodnot různých typů, jako je znak (char), číslo s pevnou desetinou čárkou (integer) nebo s 64b číslo double.

Každý posílaný datagram musí být ukončen. K tomu slouží funkce `End()`, která zapouzdří hodnoty – přidá na začátek počet znaků, vypočte kontrolní znak a nastaví ukazatel čtení (pointer) na hodnotu 4. Ta odpovídá místu, kde začíná první znak první hodnoty. Toto nastavení ukazatele je důležité pouze k dodržení specifikace protokolu a například pro zpětnou kontrolu (následné čtení hodnot).

Funkce `ToString()` vrátí datagram v textové podobě.

V třídě datagram jsou definovány všechny potřebné konstanty.

```
public const char START_BYTE = '@';

public const char DATA = 'V';
public const char COMMAND = 'C';
public const char REGULATOR = 'R';
public const char OK = 'O';
public const char ERROR = 'E';

public const char JUMP = 'J';
public const char RAMP_UP = 'U';
public const char RAMP_DOWN = 'D';
public const char SAMPLING_PERIOD = 'P';
public const char SPEED = 'S';

public const char COMMAND_ID_REGULATOR = 'R';
public const char COMMAND_ID_PROGRAM = 'A';
public const char COMMAND_ID_MANUAL = 'B';
public const char COMMAND_ID_STOP = 'C';
public const char COMMAND_ID_SAMPLING_PERIOD = 'S';
```

Následující příklady ukazují použití třídy `Port` a `Datagram`.

Nastavení vzorkovací periody. Poslání příkazu do desky.

```
int samplingPeriodMultiplier = 3;

Datagram datagram = new Datagram();
datagram.WriteSymbol(Datagram.COMMAND);
datagram.WriteSymbol(Datagram.COMMAND_ID_SAMPLING_PERIOD);
datagram.WriteInt(samplingPeriodMultiplier);
datagram.End();

this.port.Write(datagram);
```

Žádost o zaslání nastavené vzorkovací periody v řídicí desce.

```
Datagram datagram = new Datagram();
datagram.WriteSymbol(Datagram.COMMAND);
```

```
datagram.WriteSymbol(Datagram.COMMAND_ID_SAMPLING_PERIOD);
datagram.End();

this.port.Write(datagram);
```

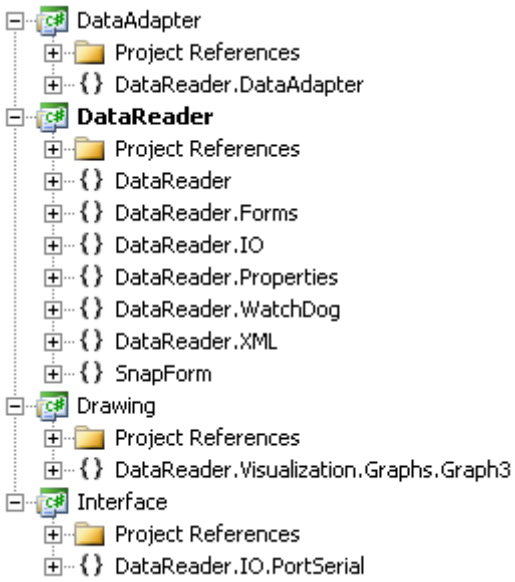
a následné čtení odpovědi:

```
/// <summary>
/// Port_s the data available.
/// </summary>
/// <param name="sender">The sender.</param>
/// <param name="e">The e.</param>
void Port_DataAvailable(object sender, Port.DataAvailableEventArgs e)
{
	switch (e.Data.Id)
	{
		case Datagram.SAMPLING_PERIOD:
		int samplingPeriodMultiplier = data.ReadInt();
		// ...
		break;

		case Datagram.REGULATOR:
		// New code here
		break;

		case Datagram.ERROR:
		// New code here
		break;

		case Datagram.OK:
		// New code here
		break;

		// Next case...

		default:
		break;
	}
}
```

8.1.2.e DataReader

DataReader je projekt, který vytváří vlastní spustitelný exe soubor programu a dynamicky linkuje všechny výše zmíněné komponenty. Zpracovává hlavní rutiny týkající se komunikace po sériové lince, manipulace s daty, nastavení, zobrazování a předávání informací do dalších oken.

Projekt je rozdělen do několika hlavních jmenných prostorů, ve kterých se nachází například všechna zobrazovaná okna, rozhraní pro port atd.

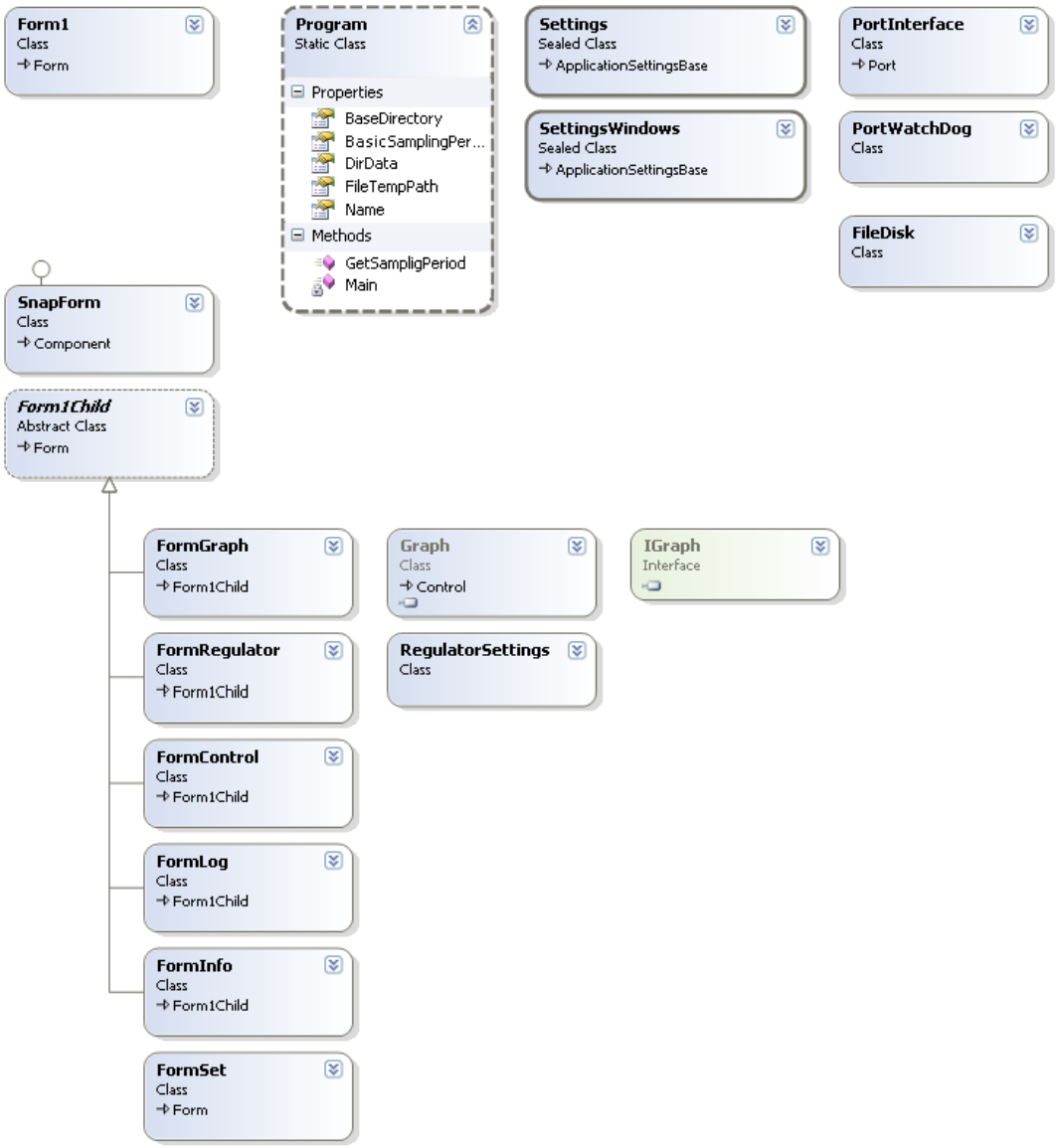


Ilustrace 23: Sestavení DataReader - jmenné prostory

Nastavení programu se ukládá do `DataReader/Properties/Settings.settings` a vlastnosti oken se ukládají separátně do `DataReader/Properties/SettingsWindows.settings`.

Nastavení portu se deserializuje do složky programu. V místě instalace se nachází i složka pro logy a je zde i soubor pro zálohování naměřených dat.

Ve jmenném prostoru SnapForm se nachází třída zajišťující magnetický efekt přichytávání oken [12].



Ilustrace 24: DataReader - class diagram

8.1.3. Formát dat

Data jsou ukládána do souboru typu csv. Jako oddělovač je použit středník. První je čas odměru, číslo vzorku, podle kterého je možné kontrolovat kontinuitu dat, následuje napětí na palivovém čerpadle v milivoltech, teplota (°C) a otáčky rotoru turbíny za minutu.

Příklad

```
96,00;33734;10000;700;40000
128,00;33735;10000;700;40000
160,00;33736;10000;700;40000
192,00;33737;10000;700;40000
```

8.1.4. Výkon

8.1.4.a Zadání

Program byl testován z hlediska spotřeby operační paměti a času procesoru.

K testování byl zvolen notebook, stolní počítač s následujícími parametry a byla použita finální release verze.

Parametry notebooku

Procesor	Intel® DualCore 2
Typové označení procesoru	T5600
Frekvence	1,83GHz, dynamicky přepínatelná od 987MHz
Operační paměť	1GB
Grafická karta	Mobile Intel® 945GM Express Chipset Family, sdílení operační paměť
Operační systém	Microsoft Windows XP Professional Servis Pack 3
Verze .NET Framework	2.0 Servis Pack 1

Parametry stolního počítače

Procesor	AMD Athlon
Typové označení procesoru	-
Frekvence	1,20GHz
Grafická karta	nVIDIA GeForce2 MX/MX 400 64MB
Operační paměť	512GB
Operační systém	Microsoft Windows XP Professional Servis Pack 2
Verze .NET Framework	3.0

Nastavení programu

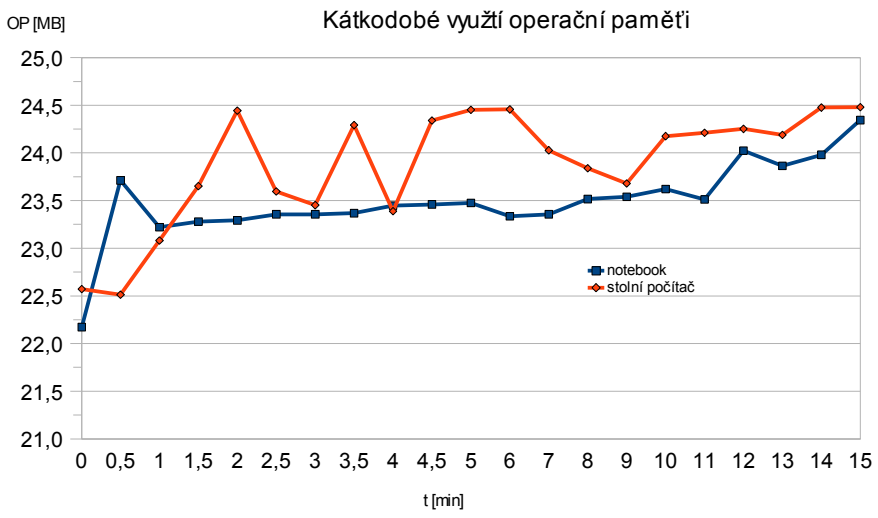
Přenosová rychlost	19200
Parita	žádná
Stop bity	1
Vzorkovací perioda	32 ms
Ukládání na disk	po 100 vzorcích
Překreslován grafu po	100 ms
Kvalita grafu	vysoká

8.1.4.b Výsledky

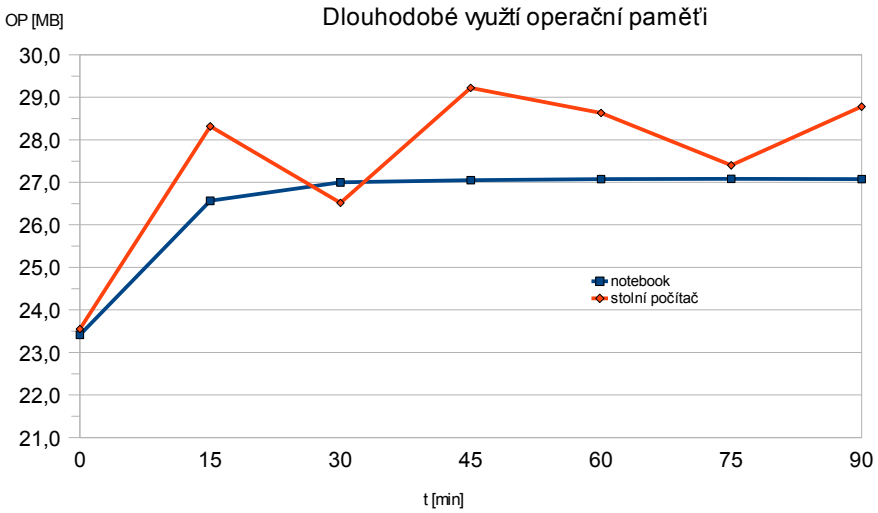
Při testování program vykazoval větší spotřebu operační paměti na stolním počítači, kde ale také znatelně kolísala. Jelikož se rozdíl pohyboval v řádu jednoho MB, lze jej pominout. Spotřeba operační paměti stále měla mírně vzrůstající charakter, byl tedy proveden nový test zaměřený na delší časový úsek.

Program spuštěný na přenosném počítači znatelně méně zatěžoval procesor. Procesor pracoval na konstantních 6%, oproti tomu procesor stolního počítače pracoval v rozmezí 27% až 34% s občasnými špičkovými hodnotami kolem 45%.

Za celý patnáctiminutový test bylo přeneseno přibližně 19250 paketů s telemetrickými daty. Dlouhodobý test následně potvrdil stabilní spotřebu operační paměti.



Ilustrace 25: Krátkodobé využití operační paměti



Ilustrace 26: Dlouhodobé využití operační paměti

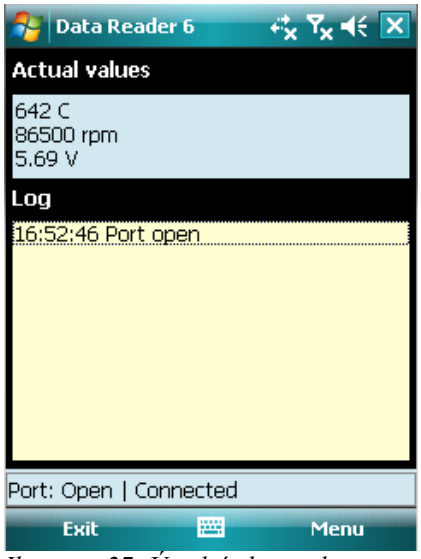
8.2. Verze pro mobilní zařízení

8.2.1. Popis

Program slouží k diagnostice turbíny a k základnímu ovládání. Program je oproti verzi pro osobní počítač ochuzen o některé funkce a je designován pro použití v terénu. Programem je tedy možno jednoduše a rychle nastavit požadované parametry regulátoru, vypnout turbínu nebo odečítat telemetrická data. Data nejsou žádným způsobem ukládána pro pozdější zpracování, ani je není možno v programu načíst a zobrazit.

Program vychází ze stejné koncepce jako je verze pro osobní počítač. Do určité míry bylo dodrženo pojmenování a jmenné prostory, díky tomu je při znalosti funkce jednoho z programů je následná orientace v druhém jednodušší.

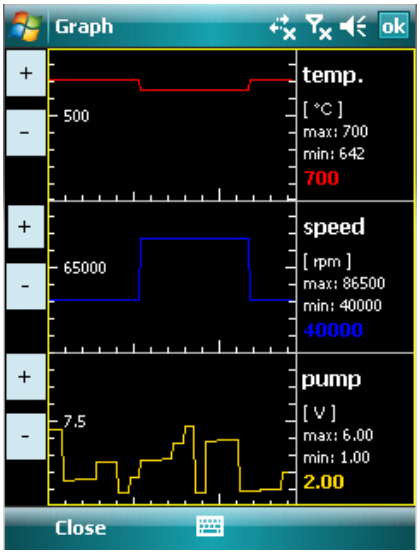
Program je rozdělen do několika částí, obrazovek. Hlavní obrazovka (Ilustrace 27), která se zobrazí po spuštění, slouží k aktuálnímu odečítání telemetrických veličin, čtení informačních textových zpráv (logů) a zobrazuje aktuální stav portu a řídicí desky. Pomocí položek v hlavním menu je možné se následně přepnout do dalších částí programu. Položka *Graph* (Ilustrace 28) slouží k vykreslování orientačního grafu s aktuálními hodnotami. Položka *Controller* slouží k odeslání – nastavení parametrů regulátoru. Pomocí položky *Port Open* je možno otevřít USB port a spustit tak komunikaci s řídicí jednotkou. Komunikace začíná vždy odesláním datagramu s nastavením vzorkovací periody do řídicí desky.



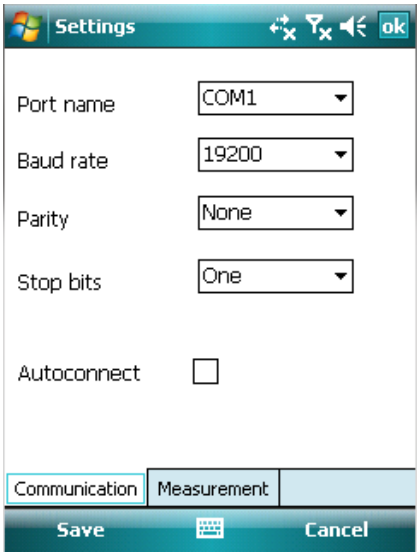
Ilustrace 27: Úvodní obrazovka

Nastavení portu a celého programu se nachází pod položkou *Settings* (Ilustrace 29).

Mechanizmus ukládání nastavení programu, který je použit u verze pro osobní počítač .NET Compact Framework nepodporuje. Proto byl vytvořen vlastní systém, tak aby mezi jednotlivými spuštěními programu nebylo nutné opětovně nastavovat určité parametry. Nastavení, jako i chybové logy programu se ukládají do složky Temp v souborovém systému Windows Mobile.



Ilustrace 28: Graph

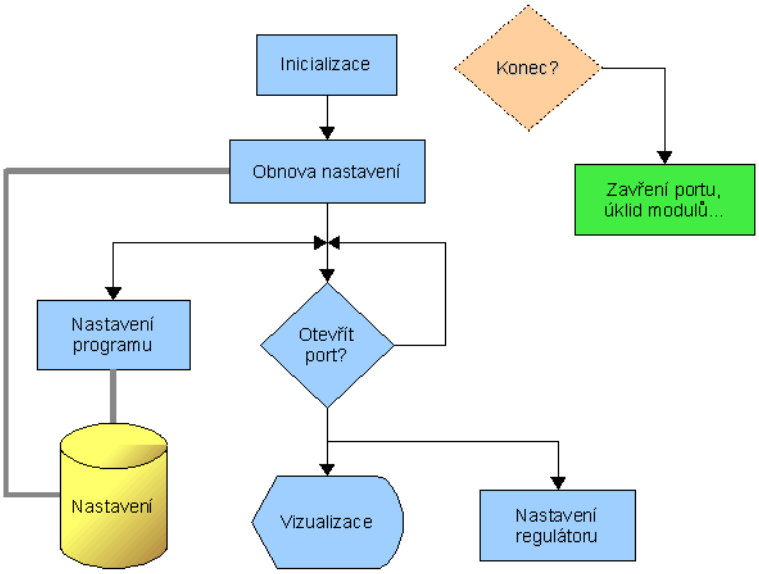


Ilustrace 29: Settings

Všechna nastavení se ukládají do cesty \Temp\Data Reader\[6] a soubory s log informacemi \Temp\Data Reader\Log\.

8.2.2. Specifikace

Jmenné prostory, třídy a celá funkcionalita byla navrhnuta s důrazem na specifika Microsoft .NET Compact Frameworku.



Ilustrace 30: DataReader6Professional - vývojový diagram

Při vývoji bylo dbáno na dodržení stejného pojmenování funkcí a tříd. Pod stejným jménem v třídě stejného jména, která se nachází v sestavení pro osobní počítač nalezneme funkci, která dělá navenek to samé. Jediný rozdíl může být ve formátu, popřípadě počtu

6 Kořenový adresář v systému Windows Mobile je \.

vstupních proměnných. Samotné manipulace a výpočty vně funkce pak pracují za použití jiných prostředků Frameworku.

Jako příklad je uveden přepočet čísla double do hexadecimální textové podoby projektu Interface (8.1.2), třídy `Datagram`. (Třída `Datagram` slouží k manipulaci paketů posílaných do řídicí desky a zpět.)

Microsoft Framework

```
/// <summary>
/// Reads the double.
/// </summary>
/// <returns></returns>
public double ReadDouble()
{
	try
	{
		string hex = this.message.ToString()
                                        .Substring(this.pointer, 16);
		this.pointer += 16;

		return BitConverter.Int64BitsToDouble(long.Parse(hex,
                                System.Globalization.NumberStyles.HexNumber));
	}
	catch (System.Exception)
	{
		return 0;
	}
}
```

Microsoft Compact Framework

```
/// <summary>
/// Reads the double.
/// </summary>
/// <returns></returns>
public double ReadDouble()
{
	try
	{
		string hex = this.message.ToString()
                                        .Substring(this.pointer, 16);
		this.pointer += 16;
		double d = 0;

		for (int n = hex.Length - 1; n >= 0; n--)
		{
			d += System.Uri.FromHex(hex[n])
                                        * Math.Pow(16, hex.Length - 1 - n);
		}
		return d;
	}
	catch (System.Exception)
	{
		return 0;
	}
}
```

Jelikož se projekt Drawing znatelně liší od verze, která je použita v sestavení pro

osobní počítač, je v následujícím odstavci podrobněji popsána. Ostatní projekty se podstatně neliší, a proto nebudou zmíněny. Pro popis tak postačí již výše zobrazené class diagramy.

8.2.2.a Drawing

Ilustrace 31: Třída TextValues

V projektu Drawing se oproti programu pro osobní počítač nachází i ovládací prvek `ActualValues`, potomek `UserControl`, který zobrazuje aktuální hodnoty telemetrických dat a je použit v hlavním oknu programu.

V třídě `Graph` stojí nejvíce za zmínku funkce `DrawOneGraphSegment()`, která kreslí jednu část výsledného grafu. Vstupní parametry pak určují pole dat, barvu, měřítko, a hlavně pozici a celkový počet. Výsledný graf tedy může být složen z libovolného počtu dílčích částí. Následující část kódu přibližuje použití a blíže popisuje vstupní parametry.

```
/**
 * The Global GDI+ Graphics
 */
Graphics g;

/**
 * Count of points
 */
int count = GetCoutOfDataToPaint(5);

/**
 * Create new points, x -y coordinates
 */
Point[] pointsS1 = new Point[count];
Point[] pointsS2 = new Point[count];

pointsS1[0] = new Point(10, 12);
pointsS1[1] = new Point(11, 14);
// ...

/// <summary>
/// Draws the graph segment.
/// </summary
/// <param name="points">The points.</param>
/// <param name="graphics">The graphics.</param>
/// <param name="graphId">The graph id.</param>
/// <param name="graphCount">The graph count.</param>
/// <param name="graphName">Name of the graph.</param>
/// <param name="graphScale">The graph scale.</param>
/// <param name="color">The color.</param>
DrawOneGraphSegment(pointsS1, g, 1, 2, 1.2, Color.Red);
DrawOneGraphSegment(pointsS2, g, 2, 2, 1.2, Color.Azure);
```

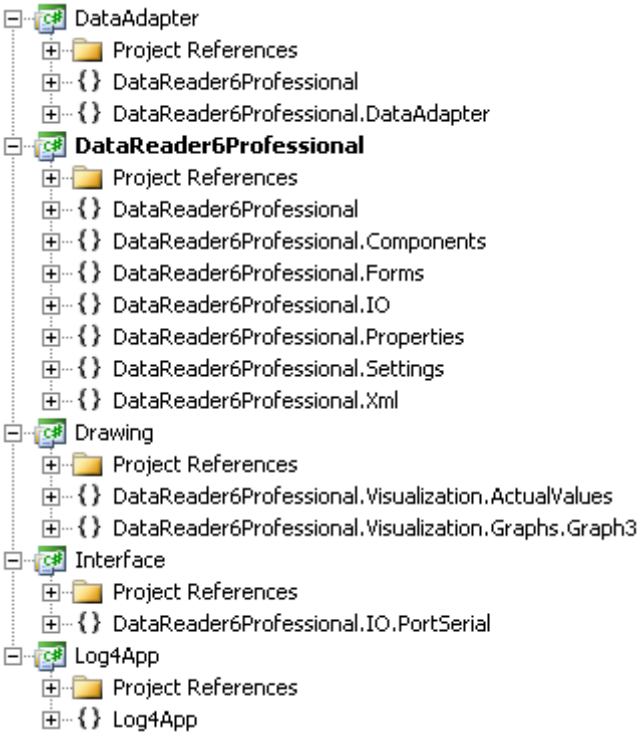
`Graphics g` je globální instance třídy, pomocí které se vykresluje – kreslící plátno. Inicializace zde není blíže rozepsána.

K vykreslení dvou grafů jsou následně vytvořena dvě pole s hodnotami

reprezentujícími souřadnice x a y. K maximálnímu počtu potřebných bodů je použita funkce `GetCoutOfDataToPaint()`, která vrací celočíselnou hodnotu počtu bodů, které zaplní celý prostor grafu. Jako vstupní parametr se zadává měřítko x-ové souřadníce.

Vstupními parametry funkce `DrawOneGraphSegment()` je pak pole hodnot, kreslící plocha, číslo grafu, celkový počet grafů které se zobrazují, měřítko osy y pro daný graf a jeho barva.

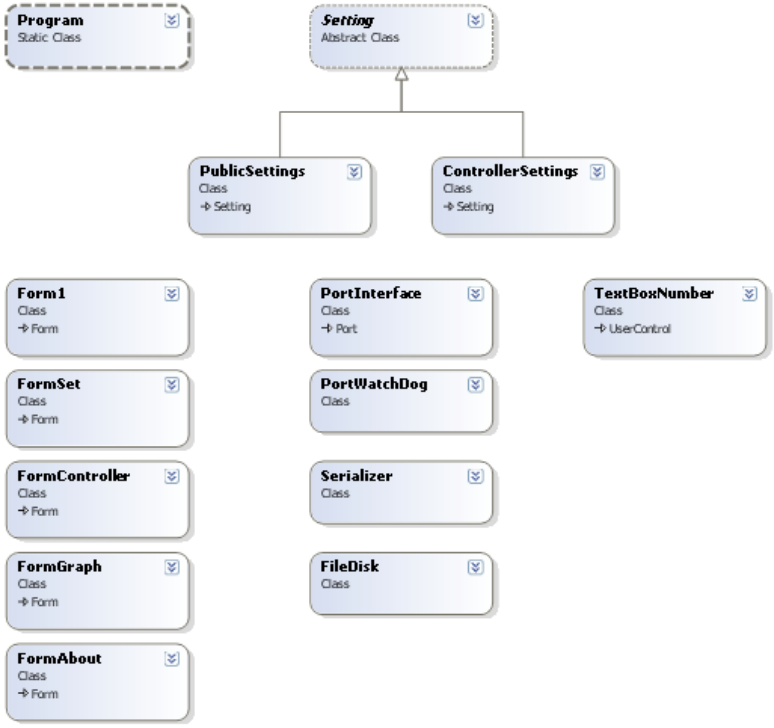
8.2.2.b DataReader6Professional



Ilustrace 32: Sestavení DataReader6Professional - jmenné prostory

Projekt DataReader6Professional seskupuje všechny ostatní projekty a vytváří vlastní spustitelný soubor, obdobně jako u verze pro osobní počítač (Ilustrace 32).

Z předchozí ilustrace jsou patrné jednotlivé projekty a jmenné prostory v nich, následující diagram reprezentuje projekt jako skupinu tříd a jejich propojení.



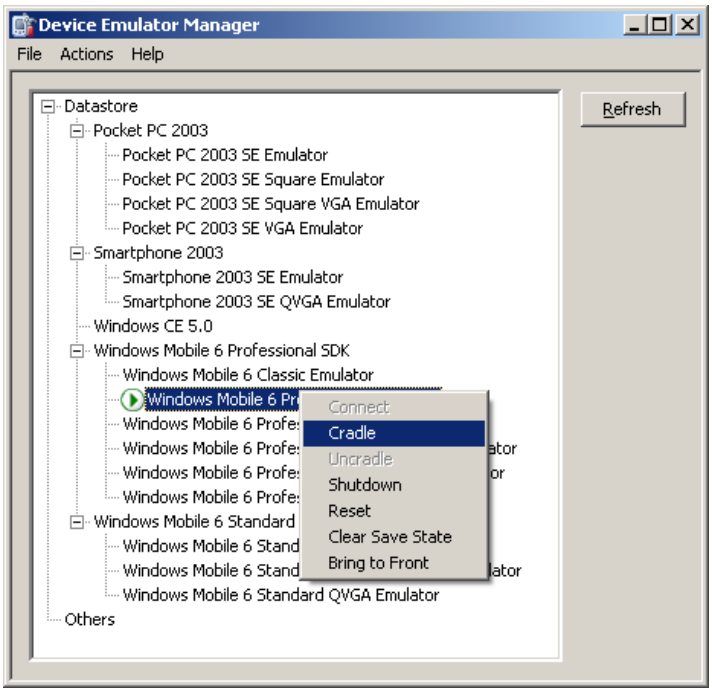
Ilustrace 33: DataReader6Professional - class diagram

8.2.3. Vývoj, debugging

Následující odstavec popisuje ladění programů ve vývojovém prostředí Microsoft Visual Studio 2005 za pomocí Device Emulator.

8.2.3.a Emulátor

Manažer emulátorů se nachází v menu pod položkou *Tools*. Po spuštění nabídne manažer instalované emulátory.



Ilustrace 34: Device Emulator Manager

Connect

Položkou *Connect* v kontextové nabídce se vybraný emulátor spustí.

Crable

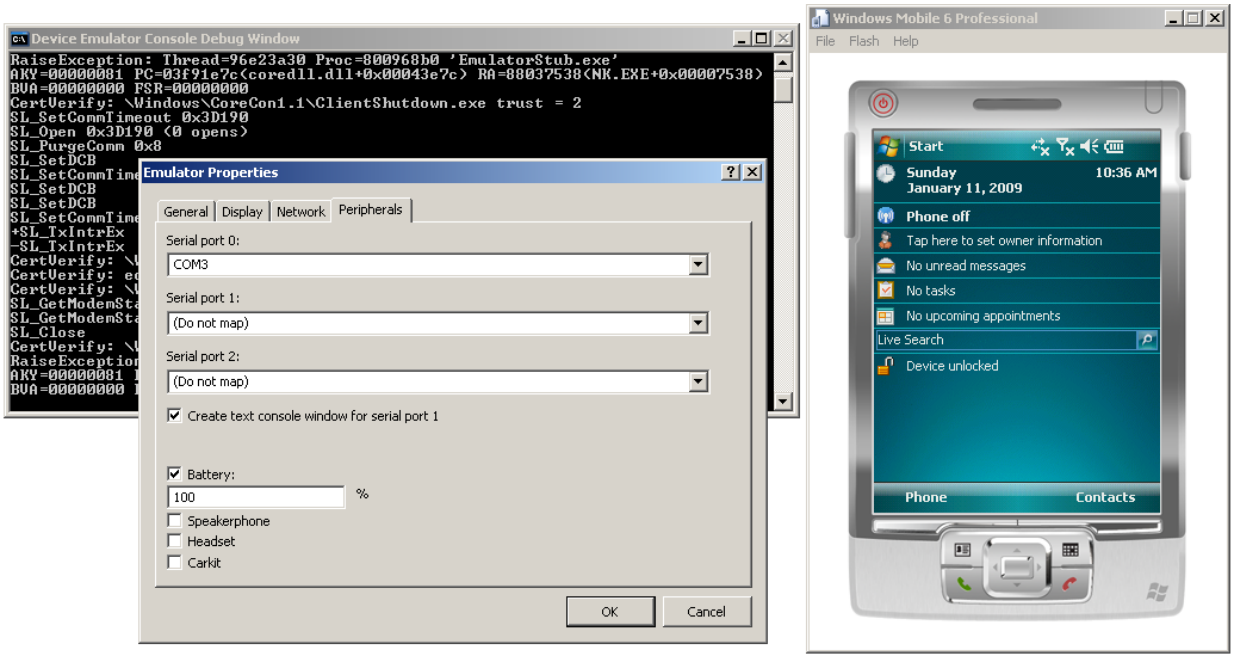
Položkou *Crable* se emulátor připojí k nainstalovanému programu Microsoft ActiveSync. Souborový systém je pak možné procházet například za pomocí pluginu wincecf (11.4) do TotalCommanderu[7], nebo přes průzkumník programu ActiveSync.

8.2.3.b Port

Následující návod popisuje připojení emulátoru k fyzickému sériovému portu systému Windows.

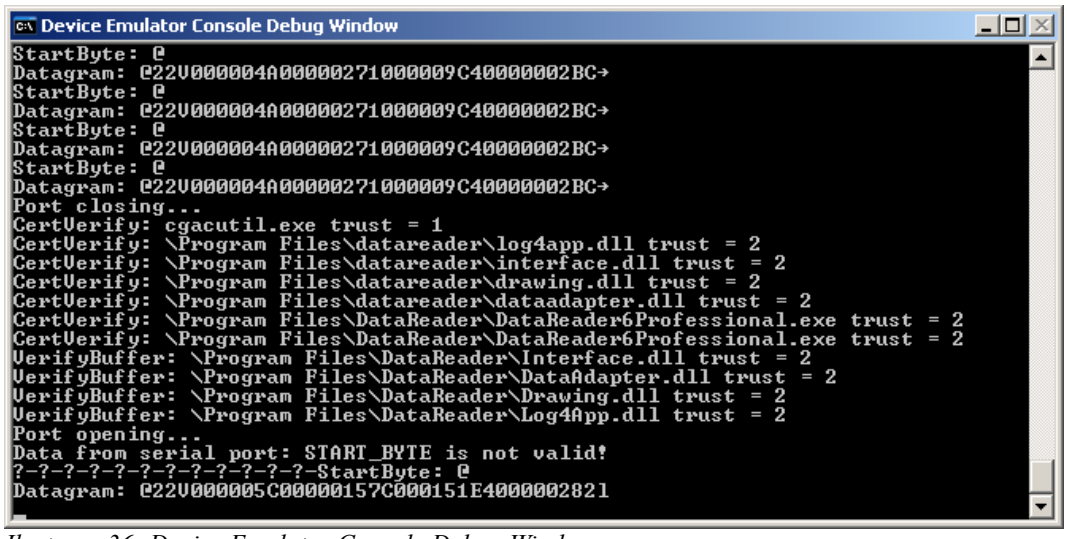
Konfigurace emulátoru se nachází v nabídce *File*. Po spuštění vybereme položku *Peripherals*, kde vybereme jméno portu a popřípadě zaškrtneme *Create text console window for serial port 1* pro debugovací konzoli.

7 Souborový manažer Total Commander http://www.totalcommander.cz/



Ilustrace 35: Emulátor Windows Mobile 6 Professional - nastavení portu

Tím se přiřadil fyzický port k portu emulátoru. V samotném programu je pak námi vybraný USB (sériový) port pojmenovaný jako COM1.



Ilustrace 36: Device Emulator Console Debug Window

Na obrázku 36 jsou vidět posílaná data, nové spuštění programu, otevření portu a následná synchronizace s řídicí jednotkou a první zasynchronizovaný datagram.

9. Instalační manuál

9.1. Verze pro osobní počítač

Instalace pro osobní počítač je připravena jako instalační balíček Microsoft Windows Installer (MSI) a spustitelný soubor exe. Instalací programu provází klasický průvodce, kde je možné zvolit cílovou složku. Instalátor zkopíruje všechny potřebné soubory a přidá zástupce do nabídky Start a na Plochu.

Poslední verze programu je na přiloženém CD ve složce Release\Data Reader (11.2).

9.2. Verze pro mobilní zařízení

Pro instalaci programu DataReader6Professional je nutné, aby mobilní zařízení splňovalo tyto minimální požadavky:

	Minimální	**Doporučené**
Procesor	200MHz	400MHz
Operační paměť	16MB	16MB
Paměť	Instalační balíček cca 35kB	
	Instalace 90kB	

Tabulka 17: Požadavky pro instalaci

Pro instalaci na mobilní zařízení s prostředím Windows Mobile je program připraven jako CAB balíček. Ten obsahuje všechny potřebné informace k instalaci a soubory, podobně jako MSI balíček.

Soubor Setup.CAB zkopírujeme do mobilního zařízení a spustíme. Při instalaci se pouze vybere místo instalace – zařízení nebo paměťová karta. Pokud je program nainstalován v přímo zařízení, program bude dle výkonu Pocket PC pracovat rychleji.

Po úspěšné instalaci najdeme program v nabídce *Programs* a samotné soubory programu v

\Program Files\Data Reader\

nebo

\Storage Card\Program Files\Data Reader\

Do složky \Temp\Data Reader\Log\ program zaznamenává log informace.

10. Zhodnocení

Diplomová práce navazuje na bakalářskou práci *Identifikace proudového motoru pro model letadla* stejného autora. Od původního záměru testovat a rozšířit matematický model a navržené regulátory se odvrací směrem k vývoji ovládacího softwaru.

Cílem práce tak bylo navrhnout uživatelsky přívětivé softwarové vybavení, pomocí kterého by bylo možné měřit telemetrická data turbíny a nastavovat parametry řídicí desky.

Za tímto účelem byl specifikován komunikační protokol pro přenos dat. Jelikož v době vývoje nebyla hotova řídicí jednotka, byl také vytvořen softwarový server emulující do potřebné hloubky chování řídicí desky za jehož pomocí byl vývoj testován.

Pro běžného uživatele, modeláře byl také vytvořen program fungující na platformě Windows CE. Pomocí tohoto programu bude možné jednoduše nastavovat parametry a kontrolovat stav turbíny před samotným letem přímo v terénu.

Mobilní zařízení skrývají určitá úskalí a vývojová omezení, která se v průběhu práce postupně objevovala. S největší pravděpodobností tak bude muset být po zhotovení funkční verze řídicí desky modifikován komunikační protokol s ohledem na tato omezení.

Celý systém je ale navržen tak, že by jakákoliv modifikace či rozšíření neměly činit větší potíže.

Do budoucna autor počítá s rozšiřováním podporovaných funkcí celého systému a věří v brzké nasazení do běžného používání.

11. Obsah přiloženého CD

11.1. Zdrojové kódy

DataReader\
DataReaderTester\
DataReader6Professional\
Log4App\

11.2. Release

Ve složce Release se nachází poslední verze instalací programů.

Release\Data Reader\
Release\Data Reader Tester\
Release\Data Reader 6 Professional\

11.3. Dokumentace

Documentations\

11.4. Programy

Ve složce programy se nachází podpůrné programy, pluginy pro vývoj nebo testování.

Programs\Hercules\
Programs\tcmdrplugin\

12. Přílohy

Příklad serializované třídy PortConfig (8.1.2.d).

```
<?xml version="1.0" encoding="utf-8"?>
<PortConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <name>COM3</name>
      <baudRate>19200</baudRate>
      <timeoutWrite>200</timeoutWrite>
      <timeoutRead>200</timeoutRead>
      <dataStopBits>One</dataStopBits>
      <dataParity>None</dataParity>
</PortConfig>
```

Seznam použité literatury

[1] Hájek Miroslav *„Identifikace proudového motoru pro model letadla"*, http://dce.felk.cvut.cz/knihovna/diplomky.htm, 2006

[2] Astrom, K. and Hagglund, T. *„PID Controllers: Theory, Design and Tuning, 2nd edition"*, Instrument Society of America, Research Triangle Park, NC, 1995

[3] Robinson, S., Scott Allen, K., Cornes, O., Glynn, J., Greenvoss, Z., Harvey, B., Naggel, Ch., Skinner, M., Watson, K. *„C# Programujeme profesionálně"*, Computer Press, 2003, ISBN 80-251-0085-5

[4] Vosecký Michal *„Návrh a realizace řídicí jednotky malého proudového motoru "*, 2008

[5] *Microsoft Download Center*, http://www.microsoft.com/downloads/, 2009

[6] *Windows Mobile*, http://www.microsoft.com/windowsmobile/, 2009

[7] *MSDN Library*, http://msdn.microsoft.com/en-us/library/default.aspx, 2009

[8] *Download Windows Mobile 6 Standard SDK Windows Mobile 6 Professional SDK*, http://www.microsoft.com/downloads/details.aspx?familyid=06111A3A-A651-4745-88EF-3D48091A390B&displaylang=en, 2009

[9] *Wikipedia - the free encyclopedia,* http://en.wikipedia.org/, 2009

[10] *Wikipedie – otevřená encyklopedie,* http://cs.wikipedia.org/, 2009

[11] *W3C Konsorcium: The Extensible Markup language,* http://www.w3.org/XML/, 2009

[12] *CodeProject: SnapFormExtender,* http://www.codeproject.com/KB/cs/eugsnapformextender.aspx*, 2008*

Rejstřík