

Program pro monitoring tlaku lisu

Extruder Pressure Monitoring Software

Bc. Tomáš Müller, DiS.



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2011/2012

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš MÜLLER, DiS.**
Osobní číslo: **A10295**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Program pro monitoring tlaku lisu**

Zásady pro vypracování:

1. Prostudujte rozhraní DRAK4 pro získávání provozních údajů z listu.
2. Implementujte modul pro čtení z A/D převodníku připojeného prostřednictvím sériového rozhraní.
3. Implementujte program pro zpracování údajů z 1 až 4 kanálů A/D převodníku. Program bude umožňovat současné zobrazení poslední naměřené hodnoty, odpovídající napětí na příslušném kanálu, dále vykreslení tlaku do grafu v daném intervalu. Program si bude uchovávat určitou dočasnou historii dat a umožní analýzu průběhu křivky pro automatické hlášení v případě neodpovídajícího výstupu.
4. Program bude navržen jako modulární, tzn. v případě nutnosti komunikace s jiným zařízením je možné nahradit modul pro komunikaci přes DRAK4 jiným komunikačním modulem.
5. Jednotlivé moduly programu budou umožňovat nastavení různých vlastností daného modulu převodníku.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. PECINOVSKÝ, Rudolf. GRADA. Myslíme objektově v jazyku Java: kompletní ucebnice pro začátečníky. 2., aktualiz. a rozš. vyd. Praha: Grada, 2009, 570 s. ISBN 978-802-4726-533
2. HEROUT, Pavel. Java bohatství knihoven. Vyd. 1. České Budejovice: KOOP, 2003, 242 s. ISBN 80-723-2209-5
3. ZAKHOUR, Sharon. Java 6: výukový kurz. Vyd. 1. Brno: Computer Press, 2007, 534 s. ISBN 978-802-5115-756
4. SPELL, Brett. Java. Programujeme profesionálně. 1. vyd. Praha: Computer Press, 2002, 1022 s. ISBN 80-722-6667-5
5. MELOUN, Milan a Jiří MILITKÝ. Statistická analýza experimentálních dat. Vyd. 2. uprav. rozš. Praha: ACADEMIA, 2004, 953 s. ISBN 80-200-1254-0
6. RXTX website. Dostupné z: <http://users.frii.com/jarvi/rxtx/>
7. JFreeChart website. Dostupné z: <http://www.jfree.org/jfreechart/>
8. Apache Derby website. Dostupné z: <http://db.apache.org/derby/>

Vedoucí diplomové práce:

Ing. Tomáš Dulík

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

24. února 2012

Termín odevzdání diplomové práce:

21. května 2012

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Hlavním cílem této diplomové práce bylo vytvoření softwaru pro monitoring tlaku lisu pro společnost Hella Autotechnik s.r.o. V teoretické části práce jsou uvedeny základní souvislosti mezi měřeným tlakem a jakostí výrobku. Dále je blíže specifikován A/D převodník s komunikačním protokolem DRAK 4, pomocí kterého jsou získávány provozní informace. Na závěr je probrán teoretický podklad korelační analýzy doplněný o reálné výsledky použití. Druhá polovina práce se zabývá popisem struktury vyvíjené aplikace, kde v první části jsou uvedeny zásady pro práci s „AppFrameworkem“.

Klíčová slova: tlak, DRAK4, COM, koeficient korelace, modul, křivka, kanál.

ABSTRACT

The main aim of this thesis was to develop extruder pressure monitoring software for the company Hella Autotechnik s.r.o. The theoretical part presents the basic relation solve between the measured pressure and the quality of the product. It also describes the A/D converter and the communication protocol DRAK 4, which is used for getting operational parameters of the extruder. Finally, it presents a theoretical basis of correlation analysis, which is completed by real results. The second part of the thesis deals with the description of the structure of the developed applications, where the first part presents principles for working with "AppFramework".

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Tomáši Dulíkovi za podnětné připomínky a rady při sestavování tohoto textu.

Dále bych rád poděkoval Ing. Lubomíru Benešovi za ochotu, trpělivost a spolupráci při tvorbě této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci, nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně, nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 EXISTUJÍCÍ ŘEŠENÍ	11
1.1 DOSAVADNÍ SOFTWARE.....	11
1.2 SOFTWARE IMPLEMENTOVANÝ V SAMOTNÉM VSTŘIKOVACÍM ZAŘÍZENÍ	11
2 PROCES VÝROBY	13
2.1 POPIS PROCESU VÝVOJE SVĚTLOMETU	13
2.2 KRAUSSMAFFEI 300.....	13
2.2.1 Vstřík hmoty.....	14
2.2.2 Vstříkovací tlaky	15
3 LABORATORNÍ A/D PŘEVODNÍK DRAK4.....	17
3.1 POPIS.....	17
3.2 BLOKOVÉ SCHÉMA A ČINNOST	17
3.3 KOMUNIKAČNÍ PROTOKOL DRAK 4.....	18
4 KORELAČNÍ ANALÝZA.....	20
4.1 STŘEDNÍ HODNOTA.....	20
4.2 ROZPTYL	20
4.3 SMĚRODATNÁ ODCHYLKA	21
4.4 KOVARIANCE	21
4.5 KOEFICIENT KORELACE	22
4.6 UKÁZKY POUŽITÍ NA REÁLNĚ NAMĚŘENÝCH DATECH	23
5 POUŽITÉ KNIHOVNY.....	24
5.1 KNIHOVNA RXTX.....	24
5.2 KNIHOVNA JFREECHART	25
5.2.1 Základní třídy knihovny	25
5.3 KNIHOVNA APACHE DERBY	27
5.3.1 Embedded.....	27
5.3.2 Server	27
5.3.3 Komunikace	27
5.3.4 Datové typy	28
5.3.5 Instalace.....	28
II PRAKTICKÁ ČÁST	29
6 APP FRAMEWORK	30
6.1 INSTALACE APPFRAMEWORKU	30
6.2 LOGOVÁNÍ V APPFRAMEWORKEM	31
6.3 TVORBA MENU NABÍDEK	32
6.4 PRAVIDLA TVORBY MODULŮ	33
6.5 PREFERENCES PANEL.....	33
6.5.1 Tvorba panelu.....	34
6.5.2 Pravidla při návrhu GUI.....	34

6.6	OBEČNÁ PRAVIDLA	35
7	NÁVRH DATABÁZOVÉ STRUKTURY	36
8	MODUL BUCHERSOURCEMODULE.JAVA	37
8.1	TŘÍDA BUCHERDEVICE.JAVA	37
8.2	BUCHERSOURCEPREFERENCES.JAVA	38
8.2.1	Návrh GUI.....	38
8.3	READINGFROMSERIALTHREAD.JAVA.....	39
8.3.1	Dostupné COM porty	39
8.3.2	Připojení ke COM portu	39
9	MODUL PRESSUREMONITORMODULE.JAVA	40
9.1	PRÁCE S DATABÁZÍ	41
9.1.1	Připojení k databázi	42
9.1.2	Struktura SQL dotazů.....	42
9.2	PRESSUREMONITORPANEL.JAVA	43
9.2.1	Návrh GUI.....	44
9.2.2	Tvorba grafu	44
9.2.3	Algoritmus pro vykreslení.....	45
9.2.4	Pravidlo vyhodnocení korelační analýzy	46
9.3	PRESSUREMONITORPREFERENCES.JAVA	47
9.3.1	Omezení hodnot	47
9.4	CORRELATIONANALYSIS.JAVA.....	48
9.5	HISTORYMONITORPANEL.JAVA	48
9.5.1	Návrh GUI.....	49
9.5.2	Tvorba tabulky pro výpis	49
9.5.3	Filtr pro vyhledávání	51
9.5.4	Zobrazení křivky	51
9.5.5	Aktualizace výchozích dat pro korelační analýzu.....	52
10	ZÁVĚREČNÉ FÁZE PROJEKTU	53
10.1	TVORBA SPUSTITELNÉHO (EXE) SOUBORU POMOCÍ LAUNCH4J	53
10.2	TESTOVÁNÍ APLIKACE	54
10.3	OVLÁDÁNÍ APLIKACE	55
	ZÁVĚR	57
	ZÁVĚR V ANGLIČTINĚ.....	58
	SEZNAM POUŽITÉ LITERATURY.....	59
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	60
	SEZNAM OBRÁZKŮ	61
	SEZNAM TABULEK.....	62

ÚVOD

Náplní této práce je tvorba softwaru pro monitoring tlaku lisu ve firmě Hella Autotechnik s.r.o. V současné době firma již vlastní starší softwarový produkt, který slouží pro monitoring tlaku lisu. Tento software se však stává zastaralým a pod novějšími verzemi operačních systémů nefunkčním. Také pro koncové uživatele je software navržen neefektivně, protože nezobrazuje přehledně informace, které koncové uživatele nejvíce zajímají. Jedná se o informaci o napětí na jednotlivých kanálech. Tyto informace lze získat, až v různých podnabídkách. Program také neumožňuje žádnou historii dat či analýzu křivek. Firma má také snahu o jednotný vzhled a správu používaných aplikací, které jsou tvořeny na základě jejich vlastního „AppFrameworku“. Proto se firma rozhodla nahradit tuto aplikaci vlastním softwarem.

Po vytvoření analýzy s koncovými uživateli a IT pracovníky vznikl seznam požadavků na nově vyvíjenou aplikaci. Aplikace byla tedy vyvíjena a upravována přesně podle těchto požadavků společnosti. Mezi tyto hlavní požadavky patřilo prostudovat a implementovat komunikační protokol DRAK4, určený pro získávání provozních informací ze vstřikovacích strojů Krauss Maffei 300. Program musí mít dále možnost současného zpracování údajů z 1 až 4 kanálů. Nejčastěji je však v reálném provozu použito dvou současně zpracovávaných kanálů. Program je navržen jako modulární, to znamená, že v případě nutnosti komunikace přes jiný protokol než DRAK4, dojde k nahrazení pouze tohoto modulu. Tato situace může nastat v případě zakoupení jiných A/D převodníků, nebo nahrazení starších převodníků za novější, které používají jiný komunikační protokol. Program si také uchovává v lokální databázi určitou dočasnou historii dat. Historii uložených průběhů lze jednoduše prohlížet. Jednotlivé zaznamenané průběhy křivek jsou podrobeny korelační analýze, která pomocí standardizovaného korelačního koeficientu určitým způsobem ohodnocuje kvalitu výstupních křivek. Výsledky ohodnocení shody se vzorovou křivkou jsou automaticky zobrazovány.

Aplikace je psaná v programovacím jazyce Java, která díky své přenositelnosti umožňuje pracovat na různých platformách. Pro běh aplikace stačí mít nainstalovaný Java Runtime Enviroment.

I. TEORETICKÁ ČÁST

1 EXISTUJÍCÍ ŘEŠENÍ

1.1 Dosavadní software

Jak bylo krátce zmíněno v úvodu této práce, firma již vlastní a používala software, který měl plnit obdobnou funkci jako tato vyvíjená aplikace. Tento už poměrně starý software se však stal pro dnešní operační systémy zastaralým a hlavně není navržen přesně podle aktuálních potřeb a požadavků firmy. Navíc obsluha této starší verze je pro uživatele poměrně komplikovaná, protože aplikace obsahuje spousty podnabídek. Například, pokud uživatel potřeboval najít pro něho důležité informace, musel se složitě proklikat do patřičné podnabídky.

Dosavadní aplikace nebyla navržena firemním IT oddělením, proto se správa a případné opravy chyb stávají neefektivní. Z tohoto důvodu je mnohem efektivnější a finančně výhodnější vytvořit nový vlastní produkt, který lze pak podle měnících se potřeb, poměrně snadno udržovat aktuálním. Oproti tomuto staršímu softwaru nabízí naše aplikace několik výhod, mezi které patří například archivace naměřených průběhů po dobu jednoho až sedmi dnů. Tuto historii si lze jednoduchým způsobem následně prohlížet. Dalším doplněním je snaha vyhodnocovat kvalitu jednotlivých průběhů pomocí korelační analýzy.

Další obrovskou nevýhodou stávající aplikace je nevyhnutelné budoucí nahrazení starších A/D převodníků firmy Papouch, které nebudou již používat komunikační protokol DRAK4. S touto skutečností již nová aplikace počítá a je navržena tak, že v případě potřeby se nahradí pouze modul zajišťující komunikaci přes tento protokol novým, ale nemusí se psát celá aplikace znovu.

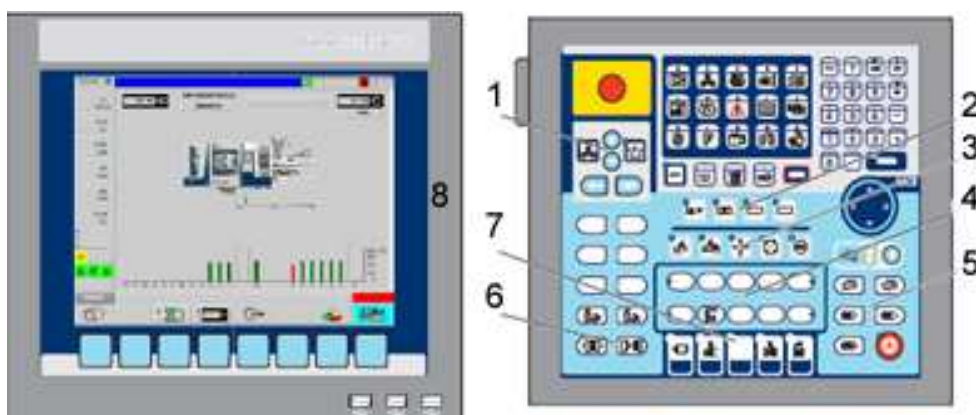
1.2 Software implementovaný v samotném vstřikovacím zařízení

Jednou z dalších možností kontroly vstřikování a vyhodnocení tlaku při samotném vstřikování nabízí do jisté míry také samotný vstřikovací stroj. Všechny novější verze těchto strojů, které ve firmě postupně nahrazují starší vstřikovací stroje, jsou doplněny o ovládací a indikační panel.

Tento panel je primárně určen pro nastavení vlastního vstřikovacího stroje, aby vytvářel výrobky s požadovanou jakostí. Zde se také nachází popis, jak by měl vypadat výsledný průběh tlaku, který kontroluje naše vyvíjená aplikace. Tento implementovaný software je ale opatřen i vlastními automatickými alarmy, které se zpustí při kritické události, která je specifikovaná pomocí nastavení hodnot odpovídajících parametrů. Může se tak hlídat

rychlost najíždění, teploty a tlaku ve vstřikovacím agregátu či samotné formě. Takže v případě velké kolize by tento software chybu také detekoval a upozornil na ni.

Neobsahuje však grafickou podobu právě probíhajícího měření a další provozní informace, které by byly jednoduše přístupné. Navíc do sekce nastavení ovládacího a indikačního panelu by měl vstupovat pouze vyškolený personál, protože neuvážená či nechtěná změna některých parametrů se může projevit na výsledné kvalitě produktů ve formě zmetků, které by pro firmu znamenaly značnou finanční ztrátu.



Obrázek 1: Ovládací a indikační panel (1)

Legenda:

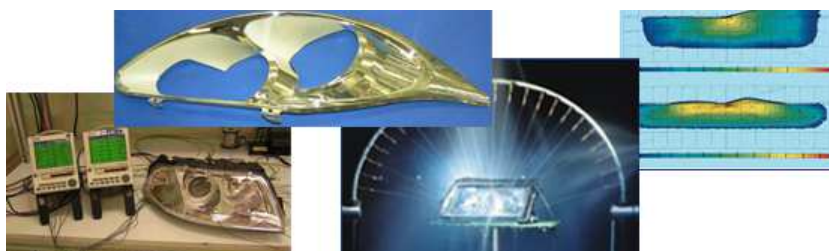
- 1) Modul tahače jader/modul upínacího zařízení
- 2) Modul druhu provozu
- 3) Robotový modul (PZ)
- 4) Modul zvláštních funkcí
- 5) Modul vstřikovacího agregátu
- 6) Modul uzavírací jednotky
- 7) Modul funkcí stroje
- 8) Obrazovka ovládacího a indikačního panelu

2 PROCES VÝROBY

2.1 Popis procesu vývoje světlometu

Protože naše aplikace je určitou malou částí, která se nachází v životním cyklu výroby světlometu, následuje krátký průlet procesem výroby.

V prvním fázi se podle požadavku zákazníka tvoří design produktu. Tento design je navrhován pomocí softwarových prostředků jako je CAD či CATIA V5. Takto navržený design je třeba doplnit o návrhy optiky a připravit podklady pro certifikaci výrobku. Následně se vytváří první počítačové simulace pro testování produktu. Jsou vytvořeny první prototypy a vzorky, které se podrobují různým testům, mezi které patří například vibrační a mechanické testování. Dále je třeba provést zkoušky těsnosti, koroze, funkce po stránce elektronické a kontroly světelných charakteristik. Po úspěšných testech a vytvoření finální podoby produktu se provede návrh výrobních montážních linek. Pro lepší představu náročnosti tohoto procesu je fakt, že doba před samotným návrhem výrobních linek je zhruba dva až tři roky. (2)



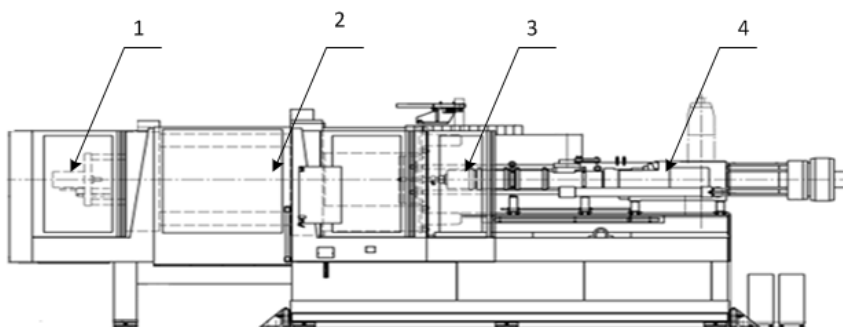
Obrázek 2: Výrobek v různých fázích vývoje (2)

Naše aplikace se v procesu výroby světlometu nachází ve fázi sériové výroby neobrobených pouzder, na které se pak následně napařují a nanáší další nezbytné vrstvy. Tento proces je realizován pomocí vstřikovacích strojů převážně značky Krauss Maffei.

2.2 Krauss Maffei 300

Vstřikovací stroj Krauss Maffei 300 je možno používat výlučně pro vstřikování artiklů z plastické hmoty. Jako vstřikovací materiál slouží granulované termoplasty a termosety, které lze použít v sestavě s přídatnými látkami v udaných mezích. Tento stroj je plánován pouze pro použití v průmyslové oblasti. Stroj se proto nesmí používat v obytné oblasti, v obchodních a živnostenských oblastech, jakož i v nejmenších provozech.

Stroj se používá pro výrobu dílů z různých plastických hmot v nejrozmanitější barvě, tvaru a velikosti. Firma pomocí Krauss Maffei 300 provádí výrobu světelných pouzder, různých tvarů a typů v závislosti na použité formě. Přitom se při zpracování termoplastů přivede výchozí materiál ve stroji zahřátím do plastického stavu a pod tlakem se vstříkuje do uzavřené formy a po ochlazení se vylisek z této formy vyjme. V dřívějších letech vyjmutí vylisku bylo prováděno zaměstnanci. Zde bylo velké nebezpečí popálení z důvodu vysoké teploty vylisku při vyjímání. V dnešní době na většině strojů tuto práci nahradila robotická ruka.



Obrázek 3: Nákres vstřikovacího stroje Krauss Maffei 300 (1)

Legenda:

- 1) Vyhazovač, který se stará o otevírání a zavírání formy
- 2) Forma schovaná za uzavíratelnými dveřmi, které se automaticky otevírají na čas určený k vyjmutí vylisku. Během vstřikování musí být tyto dveře zavřené.
- 3) Obslužný panel pro ovládání a nastavení parametrů stroje.
- 4) Vstřikovací agregát zajišťující vstřikování hmoty do formy.

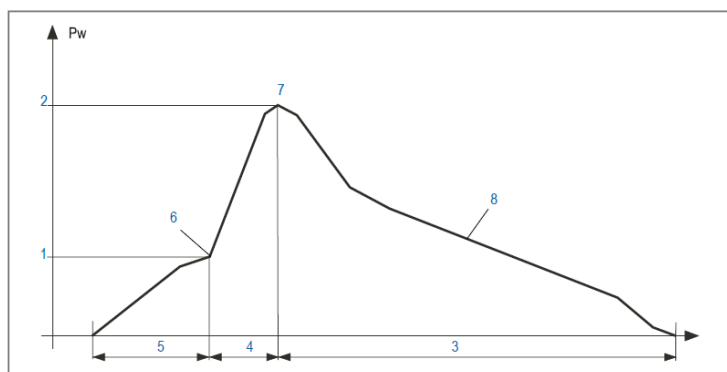
2.2.1 Vstřík hmoty

Do vstřikovacího agregátu plnicím otvorem nateče materiál, ten je dále přemísťován pomocí otáčejícího se šneku uvnitř agregátu směrem do formy. Vlastní stěny válce jsou horké a během otáčení vzniká další přídavné teplo potřebné pro vstřík hmoty. Tvářecí hmota, která se přepravuje do předsíně šneku, posunuje šnek s přibývajícím náplní vstupního prostoru šneku stále více dozadu, až je dosažena nastavená dráha plastifikace. Na plastifikovaná tvářecí hmota pak může být šnekem, který se již neotáčí a působí už pouze jako píst, vstřikována do dutiny formy nástroje. Závěra zpětného proudění na konci šneku přitom brání zpětnému proudění taveniny do závitů šneku. Zatímco se tvářecí hmota

v nástroji ochlazuje, může se začít s plastifikací tvářecí hmoty pro další vstřík. Během samotného vstříku rozlišujeme více vstřikovacích tlaků.

2.2.2 Vstřikovací tlaky

Mezi průběhem tlaku v nástroji v našem případě formě a jakostí tvarovaných součástí je úzká souvislost. Na obrázku „Vliv průběhu tlaku na jakost tváření“ bylo přiřazeno několik jakostních znaků oblastem tlakové křivky, které tlakovou křivku hlavně ovlivňují. Pomocí tohoto obrázku lze lépe pochopit tvar, souvislosti a význam naměřených křivek pomocí této aplikace. V následujícím obrázku je vidět popis vlivů jednotlivých částí grafu na jakost výrobku.



Obrázek 4: Vliv průběhu tlaku na jakost (1)

Legenda:

- 1) Bod plnění.
- 2) Tlak max.
- 3) Doba dotlaku.
- 4) Doba komprese.
- 5) Doba vstřikování.
- 6) Povaha povrchové vrstvy.
- 7) Zformování.
- 8) Zhutnění podél dráhy tečení.

Nyní budou podrobněji popsány některé základní fáze vstřikovacích tlaků, mezi které patří fáze plnění a dotlaku.

Fáze plnění

Fáze plnění neboli vstřikování je ve zvláštní míře zodpovědná za povrchovou vrstvu a mechanické a tepelné namáhání taveniny stříhem. Při vstřikování posunuje šnek, který působí jako válec, plastickou hmotu pod tlakem a při stanovené rychlosti vstřiku do uzavřeného nástroje. V závislosti na materiálu, nebo geometrii vstřikovaného dílu může být ovlivněna rychlost najíždění šneku dopředu a omezení tlaku jednotlivými nastavitelnými hodnotami. Přičemž existuje až 16 hodnot tlaku a rychlostí, které jsou spouštěny během zdvihu vstřikování pomocí poloh dráhy. Je nutné si dávat pozor na nastavení těchto hodnot, aby se nestala situace, kdy je nastavena hodnota větší než je stanovena maximální hodnota plastifikace. Po ukončení této fáze se automaticky přepne na fázi dotlaku.

Fáze dotlaku

Samotný začátek dotlaku vychází z předchozí fáze, která již stanovila rozdělení teploty ve hmotě pro start dotlaku. Toto rozdělení teploty ovlivňuje veškerý další průběh tlaku a všechny jakostní znaky, které jsou s tím spojené. Dotlak způsobuje zhutnění taveniny a výrazně poznamenává hlavně takové vlastnosti jakosti, které jsou závislé na:

- Hustotě
- Měrném objemu, jako např. hmotnost, vtaženiny, smrštění, prodlení.

Šnek zde působí omezeně hodnotami tlaku a doby na hmotu vstříknutou do nástroje. Nejdříve se vyrovná objemové smrštění výlisku pomocí chlazení. Následně se dotlak musí nastavit pouze tak vysoko, aby to stačilo k zamezení vzniku míst se staženinami a vtaženinami. Podle zkušeností je u částečně krystalických plastických hmot nutný vyšší dotlak než u amorfních hmot. (1)

Jako zajímavost bych ještě zmínil, že v procesu vyhodnocení tlaku naší aplikací jsou zobrazovány výsledné hodnoty tlaku v nástroji. Tyto hodnoty jsou získány pomocí speciálního snímače tlaku, který je zabudován v tomto nástroji.

3 LABORATORNÍ A/D PŘEVODNÍK DRAK4

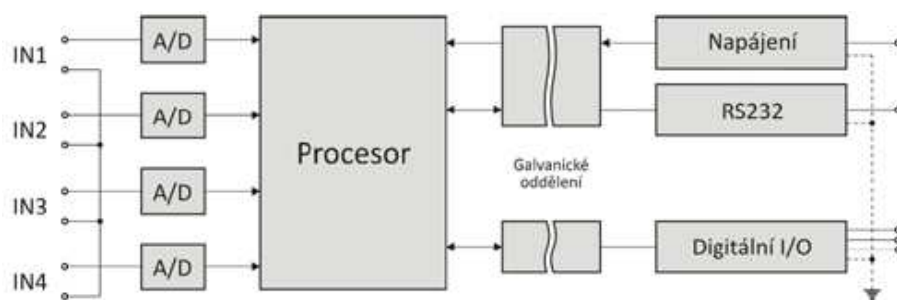
3.1 Popis

Jedná se o analogově digitální převodník komunikující přes sériový port RS232, známý jako COM port. Sériová linka RS232 je jedním z nejstarších počítačových rozhraní, které se však v průmyslu ještě hojně používají. Nevýhodou RS232 je krátká maximální délka vodičů určená dle normy pouze na 15 metrů. Mezi nevýhody také patří nízká odolnost proti rušení a nebezpečí vzniku zemních smyček, kdy při propojení dvou zařízení napájených z různých potenciálů může dojít k poškození, nebo zničení zařízení. Z tohoto důvodu se pro ochranu proti zemním smyčkám využívá galvanické oddělení RS232. (3)

Měřicí přístroj Drak 4 je určen pro měření napětí až čtyř signálů 0 až 10 V s možností přepínání zesílení. Četnost měření je 50x za sekundu. Měření provádí 16-ti bitový A/D převodník, který není multiplexován, ale je samostatný pro každý signál. Přístroj umožňuje také čtení stavu dvou digitálních vstupů a ovládání jednoho digitálního výstupu. (4)

3.2 Blokové schéma a činnost

Měřicí část je kompletně galvanicky oddělena od měřicí linky a napájení. Měřicí přístroj Drak 4 používá pro měření čtyři samostatné sigma-delta A/D převodníky. Po zapnutí napájecího napětí provede procesor inicializaci A/D převodníků a nastaví parametry přístroje podle údajů v paměti. Pak začne cyklicky měřit hodnoty napětí na analogových vstupech a výsledky ukládá do své paměti. Měřicí vstupy jsou unipolární.



Obrázek 5: Blokové schéma DRAK 4

Pro připojení vstupů IN1 – IN4 jsou určeny konektory typu BNC. Na plášť konektoru je připojena zem signálu a na středu konektoru samostatný měřený signál. Firma využívá zpravidla pouze 2 současně připojené vstupy.

Převodník obsahuje dva základní režimy pro měření a to jednorázové měření a kontinuální měření. Jednorázové měření funguje tak, že při požadavku odešle na výstup

jednu hodnotu z každého vstupu, zatím co kontinuální režim zasílá periodicky v zadaném intervalu hodnoty všech vstupů. V aplikaci je využito periodické volání jednokrokového měření. Důvodem pro nevyužití kontinuálního měření je problém s detekcí ukončení tohoto režimu. V předchozí aplikaci, kterou firma doposud využívala, se vyskytovaly problémy s ukončovacím signálem tohoto režimu. Při nesprávném ukončení kontinuálního měření totiž zařízení DRAK 4 nereaguje na žádné řídicí instrukce, kromě instrukce na ukončení. Zde právě nastal problém při nestandardním odpojení aplikace od zařízení, kdy zařízení stále pracovalo v režimu kontinuálního měření a přestalo reagovat. (4)

3.3 Komunikační protokol DRAK 4

Zařízení DRAK 4 firmy Papouch komunikuje přes protokol DRAK 4. Jedná se o starší typ komunikačního protokolu, který výrobce v novějších zařízeních doporučuje nahrazovat protokolem Spinel. Před samotnou komunikací je třeba vhodně nastavit základní komunikační parametry jako:

- Komunikační linku. V našem případě COM port, na kterém je připojeno zařízení.
- Komunikační rychlost z rozsahu od 1,2 kBd do 115,2 kBd, kde výchozí hodnota je 9,6 kBd.
- Počet datových bitů na 8.
- Paritu žádnou.
- Počet stop bitů na hodnotu 1.

Instrukce pro komunikaci v tomto protokolu mají délku vždy 3 byty. Jedinou výjimkou je instrukce pro ukončení kontinuálního měření, která má délku 1 byte. Doba odezvy je 20ms, pokud není uvedeno jinak. Instrukce jsou ASCII, někdy s binární hodnotou. Tento způsob byl zvolen jako kompromis mezi přehledností a rychlostí. V případě zadání neexistující, nepovolené instrukce, nebo jejího chybného formátu DRAK vrátí ERR [CR]. Jak již bylo jednou zmíněno, je důležité pamatovat na to, že v režimu kontinuálního měření jsou všechny instrukce kromě ukončovací ignorovány. V následující tabulce je uvedena malá část důležitých a zajímavých instrukcí, které dle mého názoru stojí za pozornost. (5)

Instrukce	Odezva	Význam
M [mezera] [mezera]	M [H1][L1] . [H2][L2]. [H3][L3] . [H4][L4] .[D1][D2][CR]	Jednorázové měření. ➤ HnLn - binární byte pro všechny 4 kanály oddělené tečkami. Hodnota se počítá 256 x Hn+ Ln. ➤ D1 D2 jsou ASCII byte s hodnotou 0, nebo 1 digitálních výstupů.
MC [T] T – rychlost vzorkování x 20ms	M [H1][L1] . [H2][L2]. [H3][L3] . [H4][L4] . [D1][D2] [CR]	Kontinuální měření. ➤ Odezva 1. vzorku je 200ms, a pak v zadaném intervalu. ➤ T je binární byte 1-255. HnLn – binární byte pro všechny 4 kanály oddělené tečkami. Hodnota se počítá 256 x Hn + Ln. ➤ D1 D2 jsou ASCII byte s hodnotou 0, nebo 1 digitálních výstupů.
X	X [CR]	Ukončení kontinuálního měření. ➤ Jediná instrukce o délce 1 byte. Pokud není poslána v režimu kontinuálního měření, hlásí chybu (ERR).
TTT	T [A] [B] [N] [V] [CR]	Testovací instrukce. Odezva jsou 4 byte : ➤ A – adresa DRAKA (doporučuji znak A-Z) ➤ B – používaná komunikační rychlost ➤ 1 – 9600Bd, 2 – 19200Bd, 3 - 38400Bd ➤ N – počet kanálů ➤ V – verze software DRAKA ➤ Tuto instrukci se doporučuje použít na začátku, aby se odstranily případné chybné byte při zapojování apod. PC posílá jednotlivá, „T“ tak dlouho, až dostane odpověď.
N [A] A – adresa DRAKA	ON [A] [CR]	Zapnutí a vypnutí komunikace ➤ Instrukce pro možnou adresaci. Adresa A je porovnána s nastavenou v DRAKOVÍ. V případě shody je odeslána odezva a DRAK komunikuje. Liší-li se adresy, DRAK neodpoví, na žádné instrukce nereaguje a čeká na instrukci ON s platnou adresou. Jako adresu se doporučuje použít znaky A až Z, a např. mezeru jako instrukci OFF. Odezva 200ms.

Tabulka 1: Instrukce protokolu DRAK 4 (5)

Kromě uživatelských instrukcí, mezi které patří výše uvedené příklady, existují také servisní a firemní instrukce. Servisní instrukce slouží pro nastavení parametrů komunikace a adresy přístroje. Do firemních instrukcí řadíme zadávání kalibračních konstant. Před každou servisní či firemní instrukcí je třeba uvést instrukci SRV.

4 KORELAČNÍ ANALÝZA

Vyhodnocení podobnosti tvaru signálu jsme se v této aplikaci pokusili realizovat pomocí korelační analýzy, která umožňuje provádět analýzu průběhu křivek. Pro charakterizaci chování vektorů používá tato analýza středních hodnot a rozptylů. K vyjádření míry intenzity vztahu mezi složkami pak slouží druhý smíšený centrální moment nazývaný kovariance. Tuto kovarianci je výhodnější standardizovat, kdy se standardizovaná kovariance označuje jako korelační koeficient. (6)

4.1 Střední hodnota

Střední hodnota se nejčastěji značí $E(X)$. Jedná se o charakteristiku, která vystihuje nejpravděpodobnější polohu hodnot obsažených ve statistickém souboru. Z tohoto důvodu se střední hodnota stává komplexní informací, která umožňuje nahradit všechny prvky jedním údajem. Střední hodnotu můžeme vyjádřit následujícím vztahem:

$$E(X) = \frac{1}{N} * \sum_{i=1}^N X_i * p_{Xi} \quad (1)$$

kde výsledná hodnota je určena jako součet všech hodnot ze statistického souboru dělená jejich počtem. Každá hodnota je ovlivněna také její pravděpodobností výskytu. V našem případě je však u všech hodnot pravděpodobnost stejná, z tohoto důvodu lze jako střední hodnotu použít aritmetický průměr definovaný jako:

$$E(X) = \frac{1}{N} * \sum_{i=1}^N X_i \quad (2)$$

4.2 Rozptyl

Rozptyl statistického souboru patří mezi jeho nejvýznamnější charakteristiky. Zkoumá variabilitu všech hodnot obsažených v daném statistickém souboru. Tato variabilita je vztažena k aritmetickému průměru. Rozptyl je definován jako střední hodnota kvadrátů odchylek od střední hodnoty. Pro diskrétní náhodnou veličinu jej můžeme definovat vztahem:

$$\sigma^2 = \sum_{i=1}^N [x_i - E(X)]^2 * p_i = \sum_{i=1}^N x_i^2 * p_i - [E(X)]^2 \quad (3)$$

kde x_i jsou hodnoty, kterých může náhodná veličina X nabývat s pravděpodobnostmi p_i a $E(X)$ je střední hodnota veličiny X . V našem řešeném případě je pravděpodobnost všech diskrétních hodnot stejná, proto se předchozí vztah zjednoduší na:

$$\sigma^2 = \frac{1}{N} * \sum_{i=1}^N [x_i - E(X)]^2 \quad (4)$$

4.3 Směrodatná odchylka

Směrodatná odchylka označuje, jak se hodnoty ve statistickém souboru průměrně odchyľují od svého aritmetického průměru. Spolu s hodnotou aritmetického průměru do jisté míry udává nejpravděpodobnější místo výskytu jednotlivých prvků souboru. Je-li tato hodnota malá, jsou si prvky souboru většinou navzájem podobné, a naopak velká směrodatná odchylka signalizuje velké vzájemné odlišnosti. Vypočítat lze ze vztahu: (7)

$$\sigma = \sqrt{\frac{1}{N} * \sum_{i=1}^N [x_i - E(X)]^2} = \sqrt{\sigma^2} \quad (5)$$

4.4 Kovariance

Kovariance udává těsnost vazby mezi dvěma náhodnými veličinami. Náhodná veličina X je nezávislá na náhodné veličině Y v případě, že zákon rozdělení proměnné X není závislý na hodnotě, kterou nabyła proměnná Y . Vztah kovariance $C(X,Y)$ je definován jako:

$$C(X,Y) = E\{[X - E(X)] * [Y - E(Y)]\} \quad (6)$$

Samotný odhad vzájemné kovariance, pak lze určit ze vztahu:

$$\hat{C}_{xy} = \frac{1}{N} * \sum_{k=1}^N (x(k) - E(X)) * (y(k) - E(Y)) \quad (7)$$

Mezi základní vlastnosti kovariance můžeme zařadit:

- Kovariance může být kladná, záporná, nebo nulová.
- Kovariance je symetrickou funkcí svých argumentů.
- V absolutní hodnotě je shora ohraničená součinem $\sigma_x * \sigma_y$. (6)

4.5 Koeficient korelace

Vyjadřuje stupeň těsnosti lineární závislosti mezi dvěma veličinami. Definiční obor leží v rozmezí od -1 do +1. Jestliže je koeficient korelace větší než 0, jde o pozitivně korelované veličiny. V opačném případě se jedná o negativně korelované veličiny. Pokud je koeficient roven extrémní hodnotě +1, můžeme konstatovat, že mezi proměnnými je přímá lineární závislost. Hodnota extrému v -1, pak značí nepřímou lineární závislost. Čím více se tedy přibližuje hodnota koeficientu korelace jedničce, tím vzniká silnější závislost mezi dvěma veličinami. Závislost veličiny se sebou samotnou musí být tedy rovna 1. (6)

Je zde ale důležité zdůraznit i problémy s vyhodnocením závislosti touto metodou. Jednou z možností špatného vyhodnocení je výskyt nějaké třetí skryté proměnné. Dalším důvodem špatného vyhodnocení může být výpočet koeficientu s příliš malého souboru dat, kdy nám může vycházet velký stupeň závislosti, ale zkreslený z důvodu menšího počtu zkoumaných hodnot. V našem případě se nejčastěji určuje závislost ze souboru dat o velikosti zhruba 140 hodnot.

Vlastní hodnota koeficientu korelace je určena pomocí vztahu:

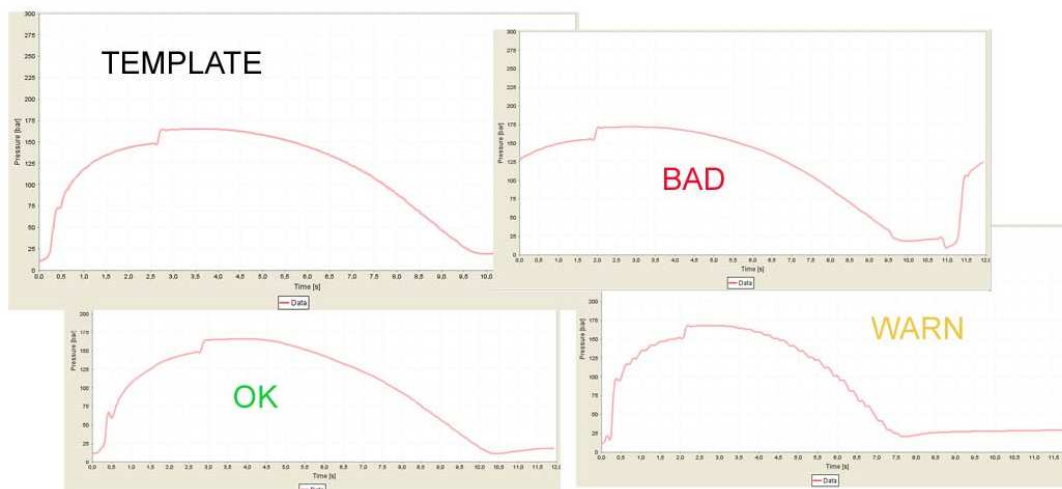
$$\bar{R} = \frac{\tilde{C}(U, Y)}{\tilde{\sigma}_U * \tilde{\sigma}_Y} \quad (8)$$

kde v čitateli je hodnota vzájemné kovariance mezi veličinami U a Y podělená součinem směrodatných odchylek U a Y. Výsledek získané těsnosti vazby můžeme obecně ohodnotit takto:

$\bar{R} < 0,3$	Těsnost nízká.
$0,3 \leq \bar{R} < 0,5$	Těsnost mírná.
$0,5 \leq \bar{R} < 0,7$	Těsnost význačná.
$0,7 \leq \bar{R} < 0,9$	Těsnost velká.
$0,9 \leq \bar{R}$	Těsnost velmi vysoká.

Tabulka 2: Obecné ohodnocení koeficientu korelace

4.6 Ukázky použití na reálně naměřených datech



Obrázek 6: Ukázky vyhodnocení korelační analýzy

V předchozím příkladu jsou zobrazeny výsledky vyhodnocení shody křivek pomocí korelačního koeficientu na námi řešený problém. Jako vzorová šablona je použita křivka „TEMPLATE“, kterou bylo potřeba nejprve zaznamenat a uložit do databáze a následně ji nastavit jako vzorovou šablonu. Limity nastavené pro vyhodnocení stavů „WARN“ a „BAD“ jsou 0,95 a 0,75. Tyto hodnoty lze v případě potřeby přenastavit. Je nutné brát na zřetel, že pro vyhodnocení je důležité nejen vhodně zvolit vzorovou šablonu, ale také hodnoty těchto parametrů.

5 POUŽITÉ KNIHOVNY

5.1 Knihovna RXTX

RXTX je Java knihovna implementovaná přes Java Native Interface, poskytující sériovou a paralelní komunikaci pro Java Development Toolkit. Knihovna je založena na specifikaci Sun's Java Communications API. Rozšiřuje tedy vlastnosti balíku gnu.io. Knihovna je navržena jako multiplatformní a nabízí podporu operačních systémů jako Solaris, Linux, Mac a samozřejmě Windows. (8) Java tedy může s využitím této knihovny přistupovat k jádru operačního systému.

Vlastní implementace je následující. Nejprve je třeba umístit patřičné knihovny do systémové cesty, nebo adresáře. U jednotlivých operačních systémů se musí použít odpovídající knihovny. Umístění a názvy knihoven verze RXTX 2.1-7 je vidět v následující tabulce.

OS	knihovna	cesta
Windows	rxtxSerial.dll	\jre\bin
MAC	librxtxSerial.jnilib	/Library/Java/Extensions
Linux	librxtxSerial.so	/jre/lib/[machine type]
Solaris	librxtxSerial.so	/jre/lib/[machine type]

Tabulka 3: Umístění knihoven v závislosti na OS

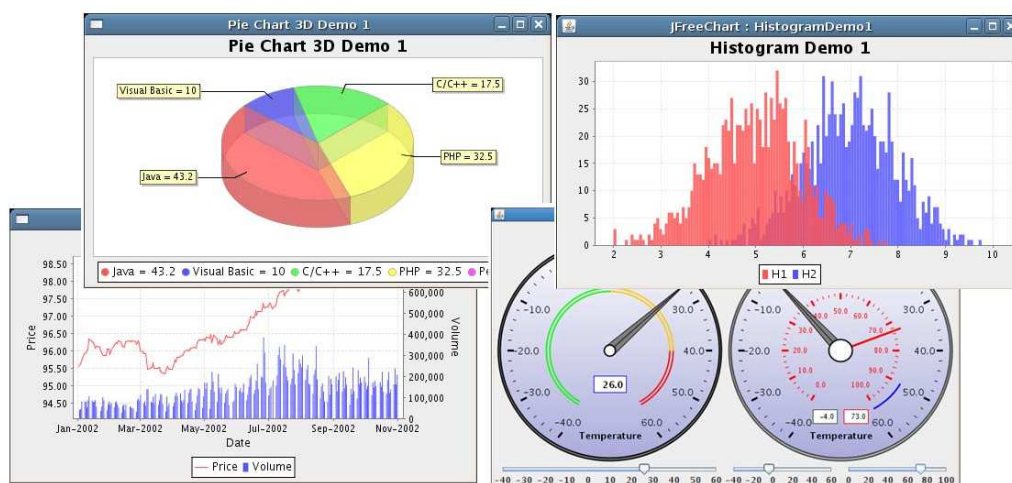
Při správném umístění knihovny načte kompilátor Javy tuto knihovnu automaticky. Pro samotnou práci pak slouží společný balíček pro všechny operační systémy pojmenovaný RXTXcomm.jar. Doporučuje se tento balíček umístit přímo do projektu využívajícího tuto knihovnu.

Knihovna je šířena pod licencí GNU LGPL. Jedná se o licenci svobodného softwaru, která se primárně používá pro softwarové knihovny. V programech lze pak tyto knihovny libovolně použít a následně šířit pod libovolnými licenčními podmínkami.

Mezi výhody tohoto řešení lze zařadit určité licenční podmínky a poměrně dobrou podporu v podobě ukázkových řešení použití této knihovny. Mezi nevýhody patří sice dostupná API dokumentace, ale s velmi skromným popisem používaných metod. Jako jednu z dalších nevýhod lze považovat i nutnost zkopírování nativních knihoven.

5.2 Knihovna JFreeChart

Jedná se o třídu Java knihoven, které usnadňují vývojářům zobrazit grafy na profesionální úrovni. Mezi základní vlastnosti patří dobře zdokumentované API se spoustou ukázkových příkladů implementace. Velkou výhodou je široká podpora tvorby všech možných druhů grafů, jako jsou koláčové, sloupcové či spojnicové grafy. Tato aplikace využívá spojnicového grafu. Umožňuje export do různých typů formátů, mezi které patří například PNG, JPEG, vektorové grafické formáty včetně PDF, EPS a SVG. Knihovny jsou distribuované pod podmínkami licence GNU Lesser General Public Licence neboli LGPL. Bližší popis těchto licencí byl uveden v kapitole věnované knihovně RXTX. Pro práci s těmito knihovnami je vyžadován JDK 1.3.1 a vyšší.



Obrázek 7: Ukázka různých typů grafů. (9)

5.2.1 Základní třídy knihovny

Pro přiblížení práce s těmito knihovnami jsou popsány klíčové třídy, které jsou využívány pro práci s grafy. Popis jednotlivých tříd vychází z API dokumentace dostupné na oficiálním webu tohoto projektu. (10)

JFreeChart

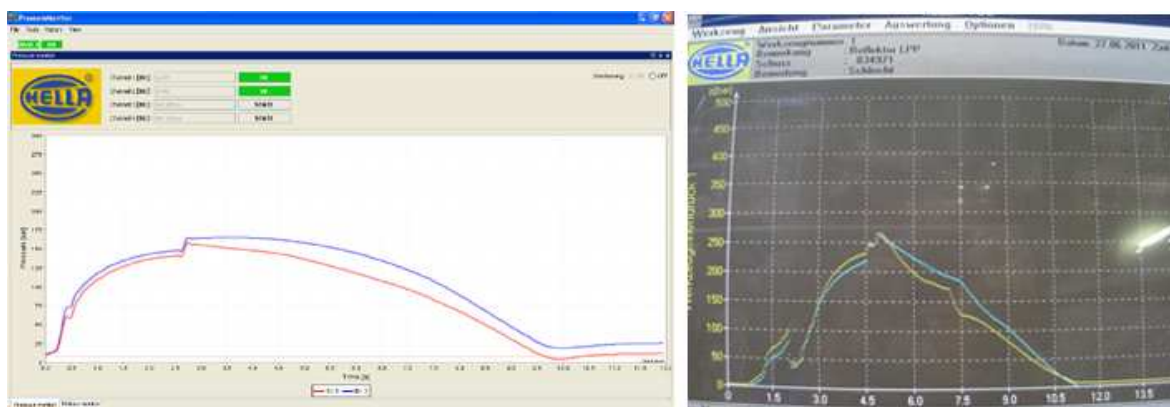
Tato třída má na starosti veškeré vykreslení grafu a koordinuje několik klíčových objektů, které se starají o možnost vykreslení grafu do Java 2D zařízení jako Graphics2D, nebo do ChartPanelu, o kterém bude řeč později. Mezi tyto klíčové objekty můžeme zařadit Title, Plot a Dataset. Title zpravidla velmi často zahrnuje legendu grafu. Plot má na starosti například rozsah jednotlivých os a stará se o samotné vykreslení Datasetu. Dataset obsahuje data, která mají být zobrazena.

ChartPanel

Jedná se o Swing GUI komponentu odvozenou z `javax.swing.JPanel`, která slouží pro zobrazení JFreeChart objektů. Tento graf se zaregistruje spolu s grafem k přijetí oznámení v případě jakékoli změny kterékoli složky grafu. Tím se zajistí automatické překreslení vždy, když dojde ke změně některé ze složek grafu. Další výhodou použití ChartPanelu je možnost práce s tímto panelem za běhu aplikace pomocí rozšířené menu nabídky vyvolané pravým tlačítkem myši. Máme možnost tak tisknout aktuálně zobrazený graf, nebo uložit ho jako obrázek ve formátu PNG. Dále lze provádět přiblížení či oddálení zobrazované plochy, nebo změnit nastavení pro jednotlivé osy.

XYSplineRenderer

Veškeré druhy rendererů, které v aplikacích můžeme používat, pochází z hlavní třídy `org.jfree.chart.renderer.AbstractRenderer`. Renderery zajišťující vlastní proložení datových bodů „splinou“. V této aplikaci je použit `XYSplineRenderer`, který využívá k proložení bodů cubicsplines. Díky tomu je zajištěno hladkého průběhu zobrazených dat. Z tohoto důvodu je výsledný graf do jisté míry zkreslený, jestliže nastanou příliš rozdílné skokové změny, které následují bezprostředně za sebou. Tuto skutečnost můžeme vzhledem k charakteru křivek zobrazovaných v reálném prostředí zanedbat. Navíc samotná aplikace umožňuje možnost volby vypnutí prokládání. Předchozí používaná verze softwaru pro zobrazení tuto možnost neumožňovala a výsledný graf byl složen pouze z naměřených bodů. Porovnání výsledků je znázorněno na následujícím obrázku, kde v pravé polovině je vidět vykreslení křivky stávajícím softwarem a v levé části zobrazení výsledku pomocí této aplikace se zapnutým prokládáním.



Obrázek 8: Porovnání výsledků měření mezi dvěma aplikacemi

5.3 Knihovna Apache Derby

Derby původně distribuovaná pod společností IBM, nyní pod distribucí Sun Java DB je open source relační databázový systém, který je založen na technologii Java a SQL. Derby je napsána a implementována kompletně v programovacím jazyce Java. Zajišťuje integritu dat a poskytuje sofistikovanou podporu transakcí. Databázový formát používaný Derby databází je přenositelný a platformě nezávislý. Díky tomu můžeme přenášet data ze stroje na stroj, aniž bychom museli jakkoli modifikovat tyto data. Pro použití této databáze je třeba mít nainstalovaný Java Development Kit verze 1.4 a vyšší.

U Apache Derby můžeme rozlišit dvě základní možnosti nasazení, mezi které patří jednoduchá embedded DB a složitější Network server DB.

5.3.1 Embedded

Jedná se lokální databázi, která poskytuje Java aplikacím single-user připojení. Derby tedy běží na stejném Java Virtual Machine jako samotná aplikace. Pro koncového uživatele je téměř neviditelná, protože se spouští a zastavuje samotnou aplikací a nevyžaduje žádnou administraci. Pro naše účely uchování dat je tento režim velice výhodný, protože odpadá starost s administrací databázového serveru. Další výhodou je snadná implementace, kdy nám stačí stáhnout jeden *.jar soubor o velikosti zhruba 2.5MB.

5.3.2 Server

Tento typ prostřednictvím sítě umožňuje multi-user přístup k databázi. Derby běží na JavaVirtual Machine hostitelského serveru. Aplikace se tedy připojují z různých JVM. V této aplikaci není síťový mód využit. (11)

5.3.3 Komunikace

Obecně pro vykonání jakéhokoli dotazu se musí nejprve navázat komunikace, vytvořit dotaz pomocí statement, vykonat samotný dotaz a zpracovat výsledek pomocí java.sql.ResultSet. Rozlišujeme dva základní druhy SQL příkazů. Pro jednoduché dotazy bez parametrů je použit java.sql.Statement, zatímco pro předkompilované SQL dotazy, které mohou obsahovat vstupní parametry, se používá java.sql.PreparedStatement.

5.3.4 Datové typy

Následující seznam uvádí nejběžněji používané datové typy. Jako doplňující informace je uvedena velikost datového typu v bytech.

Celočíselné datové typy	SMALLINT (2), INTEGER (4), BIGINT (8)
Číselné typy s plovoucí desetinnou čárkou	REAL (4), DOUBLE (8)
Znakové datové typy	CHAR (1), VARCHAR (4)
Časové datové typy	TIMESTAMP, TIME, DATE

Tabulka 4: Datové typy Apache Derby

5.3.5 Instalace

Chcete-li nainstalovat a používat Derby je třeba si stáhnout patřičnou distribuci. Přičemž existuje hned z několika distribucí.

- Bin distribuce. Obsahuje skripty, demonstrativní programy, dokumentaci a optimalizované Jar soubory stejné jak v následující Lib distribuci.
- Lib distribuce. Jedná se o stabilní optimalizovaný malý Jar soubor. V této aplikaci je využito právě této distribuce.
- Lib-debug distribuce. Jedná se o rozšířenou Lib edici pro lepší ladění a reportování problémů.
- Src distribuce. Skládá se ze zdrojových souborů, které se používají k vytvoření všech předchozích distribucí Bin, Lib a Lib-debug..

Každou z těchto distribucí lze dále rozdělit podle data vydání stabilní verze. V naší aplikaci je použita nejnovější aktuálně dostupná distribuce z roku 2011.

II. PRAKTICKÁ ČÁST

6 APPFRAMEWORK

Cílem této práce je navržení modulů, jednak pro komunikaci se sériovým rozhraním pomocí komunikačního protokolu DRAK 4, tak zobrazení a zpracování naměřených dat získaných z tohoto sériového rozhraní. Tyto moduly ke svému běhu vyžadují instalovaný firemní framework, který si firma navrhla jako podporu pro vývoj svých aplikací. Celá aplikace je psána pomocí jazyka Java.

Důvody pro navrhování aplikací na základě firemního „AppFrameworku“, jenž si firma postupem času vyrobila a nadále se na jeho vývoji podílí, je hlavně jednotný vzhled a správa aplikací. Další výhodou je využití komponent obsahujících oproti „Swing“ komponentám další atributy pro efektivnější použití a ušetření práce při návrhu. Zjednodušení lze vidět například při návrhu „Preferences panelů“, které jsou popsány v kapitole věnující se jejich návrhu.

Jednou z dalších velkých výhod je rozšiřování tohoto „AppFrameworku“ o nově napsané moduly. Tyto moduly lze pak využívat v nově vznikajících aplikacích a umožnit tak plně teamový vývoj softwaru.

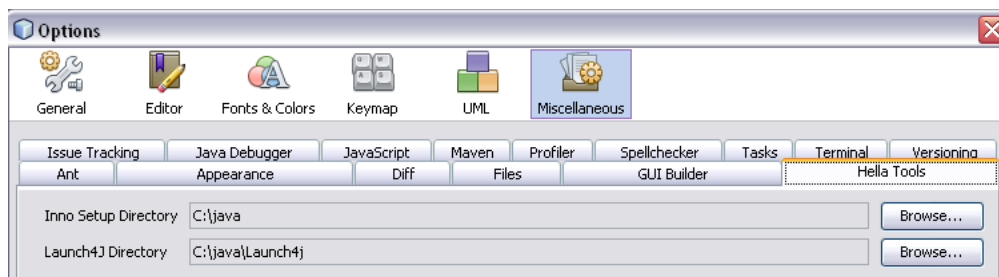
6.1 Instalace APPFRAMEWORKU

Aby bylo možné tedy vytvořit projekt v rámci této organizace bylo potřeba nainstalovat a nastavit potřebné vývojové prostředí. Prvním krokem je zkopírování potřebných dat. Jako výchozí cesta pro data se doporučuje použít directory C:/java. Jedná se o datové balíčky Launch4j a Projects. Datový balíček Projects zahrnuje projekty AppFrameworkTools a AppFramework.

Jako vývojové prostředí je voleno Netbeans 7.0.1, které je zdarma ke stažení na stránkách <http://www.netbeans.org/>. Jedná se o vývojové prostředí pro Windows, Mac, Linux a Solaris. Umožňuje rychle vytvářet webové, mobilní či desktopové aplikace na platformě Java.

Jako další krok se v Netbeans přidá nový existující projekt „AppFrameworkTools“, jenž je podle předchozího kroku zkopírován do direktory C:/java. Po úspěšném přidání projektu je provedena instalace. Instalace se spustí kliknutím pravým tlačítkem myši na tomto projektu a v zobrazené nabídce se vybere položka „Install/Reload in Development IDE“.

V podnabídce „Tools“ → „Options“ → „Miscellaneous“ přibyla položka „Hella Tools“. Zde se nastaví cesty pro položky „Inno SetupDirectory“ a „Launch4JDirectory“ dle následujícího obrázku. Cestu položky „Inno SetupDirectory“ lze volit libovolně.



Obrázek 9: Dialogové okno pro nastavení parametrů v podnabídce Hella Tools

V dialogu určenému pro tvorbu nových projektů byla přidána podkategorie „Hella“, pomocí které lze vytvořit následující typy projektů:

- iTac 5.0 Application
- iTac 6.0 Application
- Simple Hella Application

Navrhované moduly jsou vytvořeny na základě projektu „Simple Hella Application“. Posledním krokem před samotným vývojem aplikace je nutno nastavit v nově vytvořeném projektu všechny cesty ke knihovnám, které obsahuje projekt „AppFramework“.

6.2 Logování v APPFRAMEWORKEM

Mezi základní vlastnosti patří jednotné základní ovládání a podobný design aplikace. Základní kostra aplikace umožňuje zobrazení zpráv událostí, kde ke každé této události lze zobrazit její detailní popis. Do tohoto logovacího panelu lze přesměrovat jak standardní výstupy například `System.out.println("Message")`, tak chybové výstupy, mezi které patří například `System.err.println("ErrorMessage")`. Tyto standardní výstupy by se neměly v aplikaci používat. Pro zasílání zpráv do tohoto okna realizujeme prostřednictvím instance odvozené z `org.apache.log4j.Logger`. Instance se pak vytvoří následovně.

```
1 Logger logger = Logger.getLogger(this.getClass());
```

Mezi výhody aplikací navržených na základě tohoto „AppFrameworku“ je také možnost archivace těchto logovacích informací pro případnou pozdější potřebu. Kde mezi základní nastavitelné parametry logování patří název logovacího souboru, maximální velikost logovacího souboru, maximální počet těchto souborů či formát uložení logu. Tato

podnabídka pro nastavení parametrů je vyvolána v nabídce „Tools“ → „Možnosti“ → záložka Aplikace.

6.3 Tvorba menu nabídek

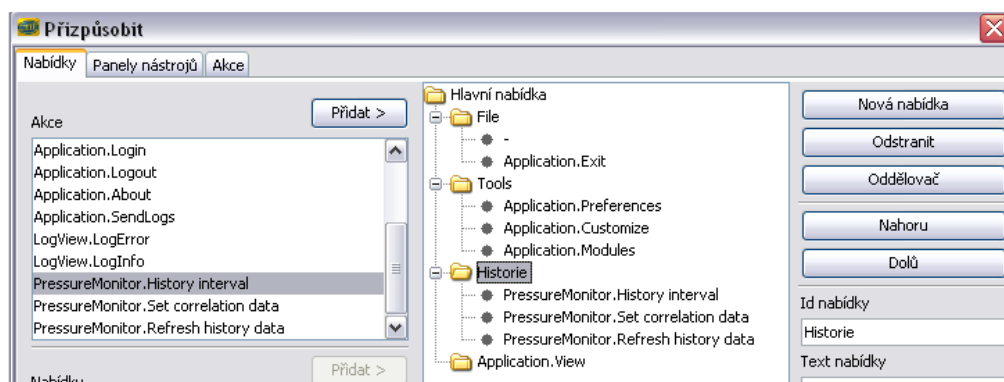
Vlastní tvorba menu nabídek není vytvářena standardním způsobem, na jaký je většina uživatelů zvyklá. Jednotlivé menu položky jsou tvořeny pomocí akcí. Tyto akce se následně zpřístupňují v příslušných nabídkách. Nabídky se definují v modulech, které logicky souvisí s danou nabídkou. O vlastní tvorbě modulů bude řeč později. Výhodou využití těchto akcí je možnost nastavení bezpečnostních oprávnění. Ty zaručí, že koncoví uživatelé budou moci zasahovat pouze do těch částí programu, které nezbytně ke své práci potřebují. Vlastní akce lze vytvořit následujícím způsobem.

```
1 ApplicationActionnameAction = newApplicationAction() {  
2 {  
3 setName("label");  
4 }  
5 @Override  
6 public voidactionPerformed(ActionEventae) {  
7 }  
8 };
```

V předchozím kódu se v první části nastavil popis akce příslušné akce. V druhé části na řádku 5-7 je využito přepsání metody `actionPerformed(ActionEventae)`, která je volaná při vyvolání akce a provede příkazy obsažené v těle této zprávy. Zatím není tato akce přidružena žádnému modulu. To provádí inicializační metoda `initModule()` daného modulu pomocí metody `addAction(nameAction, "Label")` obsahující dva povinné parametry název akce a její ID.

```
1 @Override  
2 public voidinitModule() {  
3 addAction(nameAction, "Label");  
4 }
```

Z takto nachystaných akcí lze ve vlastní aplikaci vytvořit menu nabídky. Sekce „Tools“ → „Vlastní“ obsahuje veškerou práci s nabídkami. Zde se nachází několik záložek. Záložka „Nabídky“ slouží pro vytvoření stromové struktury menu nabídek a přiřazení akcí nabídkám. Za zmínku stojí také záložka „Akce“, kde lze jednotlivým akcím nastavit label či přístupové klávesové zkratky. Následující obrázek zobrazuje strukturu menu nabídky této aplikace.



Obrázek 10: Dialogové okno pro správu menu nabídek

6.4 Pravidla tvorby modulů

Modul je základní stavební prvek, ze kterého vznikla tato aplikace. Modul je odvozen od třídy `com.hella.appfw.app.AbstractModule`. Při vytváření této třídy reprezentující modul je potřeba implementovat abstraktní metodu `initModule()`. Tato metoda se provede vždy při inicializaci daného modulu. Ke každému vytvořenému modulu je zvykem vytvořit panel odvozený od `com.hella.appfw.preferences.PreferencesPanel`. Tento panel slouží pro nastavení všech proměnných, které jsou s modulem svázané. Jednou z velmi vhodných vlastností modulu je automatické ukládání hodnot všech proměnných definovaných pro modul do konfiguračního. Tento soubor je uložen v kořenovém adresáři projektu `/config/name.properties`. To zaručuje nastavení hodnot těchto proměnných z konfiguračního souboru při každém restartu, nebo opětovném spuštění aplikace.

Aby po spuštění aplikace byl samotný modul přístupný, je třeba vytvořený modul vybrat. To se provede v dialogovém okně nacházejícím se v „Tools“ → „Moduly“. Tento dialog obsahuje seznam dostupných modulů a seznam pro vybrané moduly. Do seznamu pro vybrané moduly umístíme ty, které aplikace používá.

Pro tuto aplikaci jsou navrženy dva moduly. První modul řeší získání provozních informací ze sériového rozhraní za použití komunikačního protokolu DRAC 4. Druhý modul řeší veškeré zpracování a vizualizaci těchto naměřených hodnot. Tyto moduly jsou navrženy tak, že v případě potřeby lze modul pro získání provozních informací nahradit modulem jiným.

6.5 Preferences panel

Jak již bylo uvedeno, každý modul by měl obsahovat preferences panel určený pro nastavení hodnot důležitých proměnných.

6.5.1 Tvorba panelu

Preferences panel je odvozený od `com.hella.appfw.preferences.PreferencesPanel`. Každý z těchto panelů po zobrazení v aplikaci má přístupná tlačítka „OK“, „Použít“, „Storno“. Pro odchycení události vyvolané kliknutím na tlačítko „OK“, nebo „Použít“ je třeba implementovat abstraktní metodu `applyChanges()`. Dále aplikace využívá abstraktní metodu `resetValues()` vyvolanou při zobrazení dialogu se všemi preferences panely.

6.5.2 Pravidla při návrhu GUI

Při návrhu GUI tohoto panelu musí programátor dodržet několik základních pravidel, které byly navrženy pro zefektivnění a usnadnění práce. Jedno z pravidel je povinnost použití firemních komponent pro návrh GUI. Většinu těchto komponent nalezneme v package `com.hella.appfw.preferences`. Pro snazší použití je vhodné si tyto komponenty přidat do nové palety v Netbeans.

Dále je třeba dodržet správné pojmenování proměnných, aby došlo k vzájemnému provázání s daným prvkem. Proměnná v modulu musí začínat vždy malým písmenem. K této proměnné je třeba vygenerovat tzv. Getter/Setter. Jedná se o metody určené pro nastavení a získání hodnoty příslušné proměnné. Umožňují také určité zapouzdření proměnné a ochranu její informace, která je přístupná pouze prostřednictvím těchto metod. Následně je nutné provést provázání této proměnné s danou komponentou. Daná komponenta obsahuje atribut „`propertyName`“. Do tohoto atributu se nastaví stejný název proměnné jaký je uveden v modulu, s tím rozdílem, že název musí začínat vždy velkým písmenem. Za zmínku stojí uvést ještě jeden atribut zvaný „`runtimeUpdate`“. Ten dává informaci o tom, jestli se změna hodnoty projeví okamžitě při běhu aplikace, nebo až po jejím restartu. Pro lepší pochopení je uveden příklad provázání proměnné sloužící pro uchování názvu sériového portu.



Obrázek 11: Ukázka implementace provázání proměnné s danou komponentou

V levé části je zobrazena komponenta „ComboBoxPropertyEditor“ rozšířena o ukázkou nastavení výše zmiňovaných atributů. Pravá část představuje definici proměnné, která je provázána s komponentou.

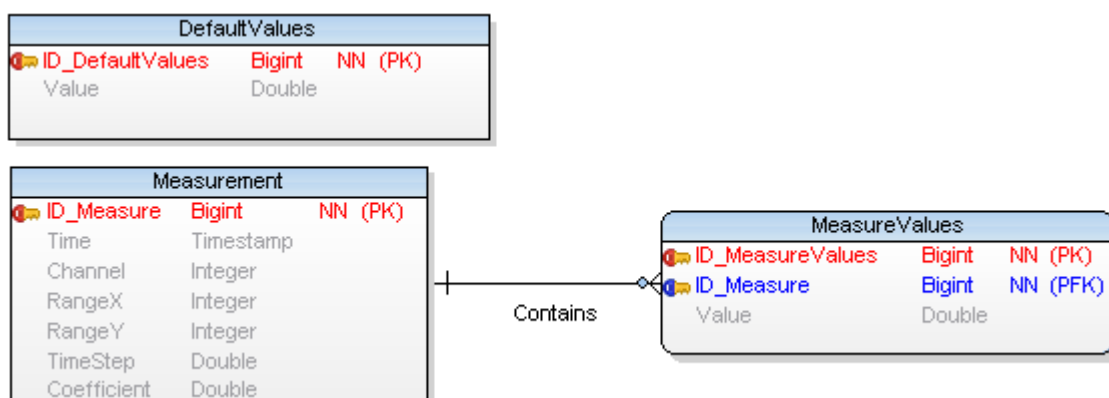
6.6 Obecná pravidla

Názvy metod, proměnných, komponent a tříd se volí tak, aby tento název co možná nejvíce vystihoval charakter a účel použití. Názvy proměnných, metod a komponent by měly začínat vždy malým písmenem, zatímco názvy tříd se uvádí písmenem velkým.

7 NÁVRH DATABÁZOVÉ STRUKTURY

Jelikož si aplikace potřebuje dočasně uchovávat naměřené hodnoty, bylo třeba zvolit způsob uchování naměřených dat. K tomuto účelu aplikace využívá databázi. Zvolena byla databáze implementovaná v čisté Javě. Jedná se o databázi Apache Derby běžící v režimu embeddable.

Databázová struktura je tvořena třemi tabulkami. Struktura jednotlivých tabulek s vzájemnými relacemi je znázorněna na univerzálním entitně relačním diagramu.



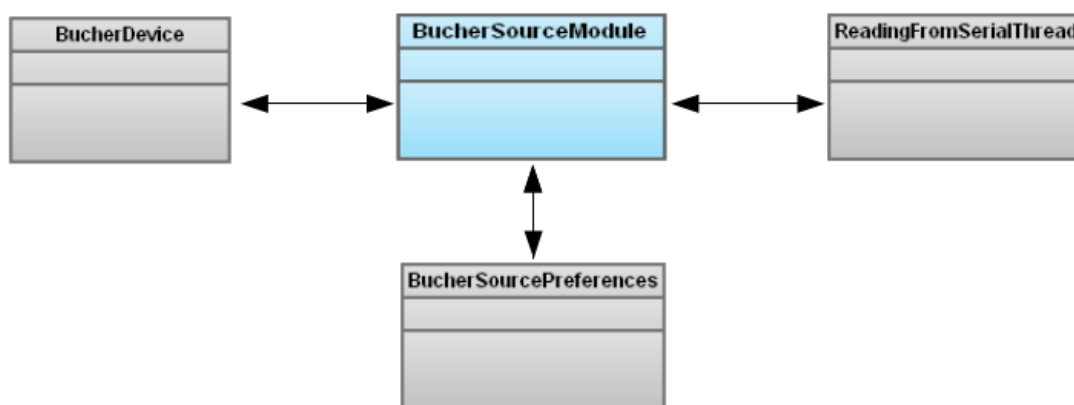
Obrázek 12: Univerzální ERD databázové struktury

Každá tabulka je identifikovaná pomocí primárního klíče (PK), který je v diagramu označen červeným klíčem. Symbol modrého klíče pak slouží pro definici cizího klíče (PFK). Z důvodu velkého množství ukládaných informací se jednotlivým klíčům volil datový typ „Bigint“, který poskytuje 8 bytovou velikost pro celočíselné hodnoty.

Tabulka „DefaultValues“ slouží pro uchování výchozího souboru dat použitého při vzájemné korelační analýze ve fázi stanovení koeficientu korelace. Tabulka „Measurement“ je určena pro uchování informací o jednotlivých měřeních, které bylo potřeba zaznamenávat. Každé měření je popsáno několika naměřenými hodnotami. Jejich počet je závislý na zvolené vzorkovací periodě a délce jednoho měření. Tyto jednotlivé zaznamenané hodnoty jsou uloženy v tabulce „MeasureValues“, která je propojena relací 1:N s tabulkou Measurement, kde jedno měření obsahuje několik hodnot.

8 MODUL BUCHERSOURCEMODULE.JAVA

Tento modul je odvozený od třídy `com.hella.appfw.app.AbstractModule` a implementuje komunikaci přes sériovou linku COM pomocí komunikačního protokolu DRAK 4. Modul však neumožňuje žádné uchování informací, které získal z připojeného COM portu. K uložení provozních informací slouží modul pro zobrazení a zpracování naměřených údajů. Proto pokud by se komunikační protokol DRAK 4 stal zastaralým, nebo by se firma rozhodla používat jiné A/D převodníky komunikující přes jiný protokol, dojde k nahrazení komunikačního modulu „BucherSourceModule.java“ jiným. Veškerá práce s naměřenými daty však zůstane nezměněná.



Obrázek 13: Diagram logického členění tříd modulu „BucerSourceModule.java“

Předchozí speciální typ diagramu se snaží jednoduše zobrazit návrh logického členění tříd. Z diagramu je dále patrné, jak probíhá komunikace mezi jednotlivými třídami a navrhovaným modulem. Z diagramu lze tedy konstatovat, že třída „BucherSourceModule.java“ může zaslat zprávu všem zobrazeným třídám v diagramu.

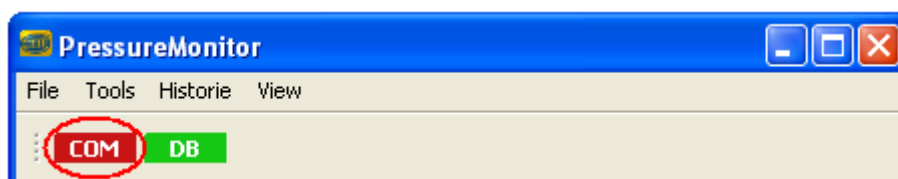
V inicializaci modulu dojde k vytvoření zařízení pro připojení k sériovému portu a k vytvoření vlákna odposlouchávající příchozí byty na připojeném portu. Pokud není dostupné žádné aktivní připojení, vlákno čeká a v pravidelných intervalech kontroluje, jestli nebylo spojení navázáno. Jestliže je spojení aktivní provádí se bytové čtení se zasíláním provozních informací modulu „PressureMonitorModule.java“.

8.1 Třída BucherDevice.java

Je odvozena z třídy `com.hella.appfw.device.AbstractDevice`. Tato třída nám reprezentuje grafickou komponentu určenou pro připojení k COM portu. V třídě je nutné implementovat abstraktní metody „`initDevice()`“ a „`closeDevice()`“. Metoda „`initDevice()`“ je volána při pokusu o realizaci připojení. Proto je v těle této metody prováděn pokus

o připojení. Oproti tomu metoda „closeDevice()“ se provede v případě zrušení aktivního spojení. V těle této metody se uzavírá otevřený COM port. Pro rychlou orientaci o stavu připojení komponenta mění barvu podle stavu, ve kterém se nachází. V případě aktivního spojení je zbarvena zelenou barvou a pro neaktivní spojení je přidělena barva červená.

Aby takto navržené zařízení bylo přístupné v modulu a viditelné v aplikaci, je třeba ho přidat do modulu pomocí metody „addDevice(IDevicedevice, String id)“. Jestliže se zařízení úspěšně přidalo do modulu, aplikace jej zobrazí.



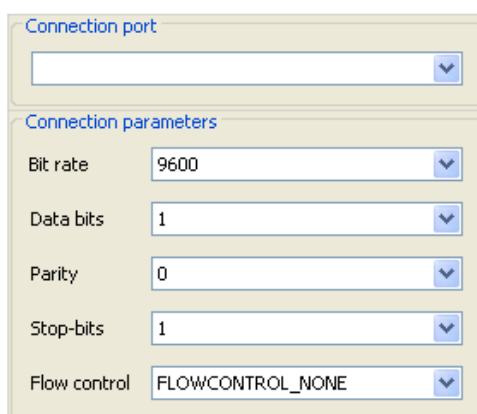
Obrázek 14: Ukázka zařízení pro připojení k COM portu a databázi

8.2 BucherSourcePreferences.java

Třída odvozená z `com.hella.appfw.preferences.PreferencesPanel`. Tento panel umožní nastavovat parametry připojení k sériovému portu. Mezi tyto parametry patří seznam dostupných COM portů, bit rate, počet datových bitů, parita, počet stop bitů a flowcontrol, který slouží pro řízení toku dat mezi dvěma propojenými zařízeními.

8.2.1 Návrh GUI

Při návrhu grafického uživatelského rozhraní se jednotlivé komponenty umístily na mřížku typu „GridBagLayout“. Výsledný návrh je vidět na následném obrázku.



Obrázek 15: GUI BucherSourcePreferences.java

8.3 ReadingFromSerialThread.java

Je odvozena od třídy Thread. V této třídě je třeba implementovat metodu run(). Ta se provede vždy po spuštění Thredu. Vlastní Thread se spouští pomocí metody start(). ReadingFromSerialThread obsahuje veškeré metody spojené s navázáním komunikace, zasíláním řídicích příkazů, zjištění všech dostupných COM portů, bytové čtení z připojeného portu či nastavení parametrů připojení. Nyní budou uvedeny zjednodušené ukázky klíčových programových kódů. Záměrně nejsou uvedeny plné výpisy daných metod.

8.3.1 Dostupné COM porty

```
1 import java.util.ArrayList;
2 import java.util.Enumeration;
3 import gnu.io.CommPortIdentifier;
4 ArrayList<String>listPorts = newArrayList<String>();
5 Enumerationenumeration = CommPortIdentifier.getPortIdentifiers();
6 CommPortIdentifier cpi;
7 while (enumeration.hasMoreElements()) {
8     cpi = (CommPortIdentifier) enumeration.nextElement();
9     listPorts.add(cpi.getName());
10 }
```

Řádek 5 získá objekty odpovídající dostupným portům. Následně cyklus while projde všechny jednotlivé objety a získá z nich jméno portu. Seznam všech dostupných COM portů je pak uveden v kontejneru „listPorts“.

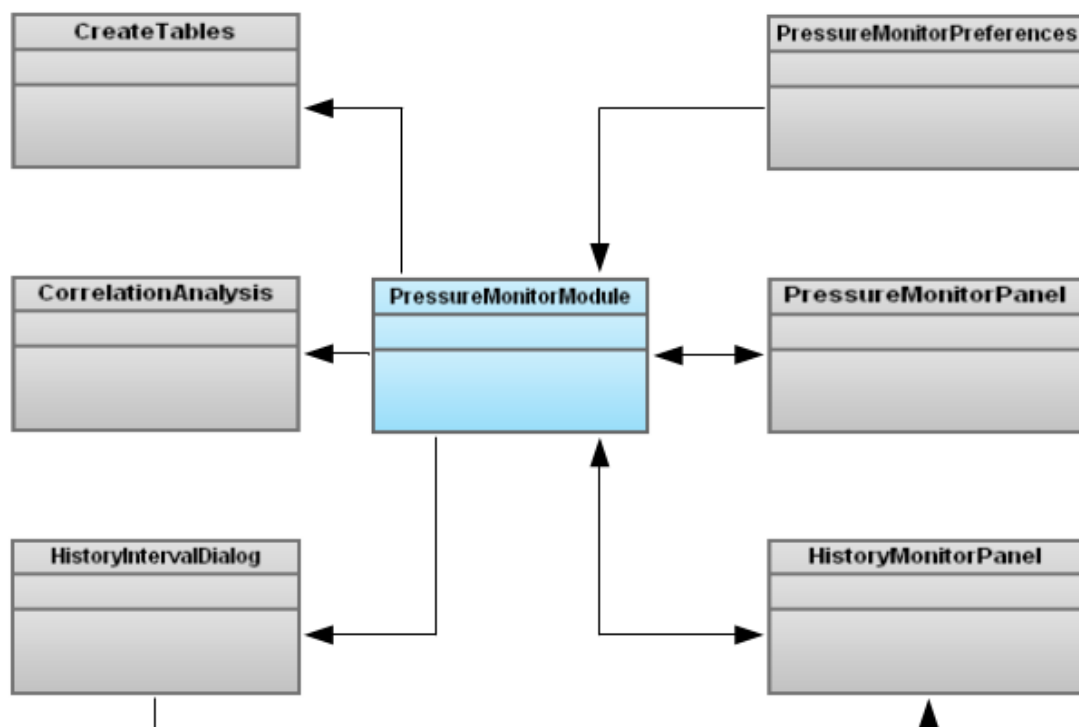
8.3.2 Připojení ke COM portu

```
1 import gnu.io.CommPortIdentifier;
2 import gnu.io.SerialPort;
3 CommPortIdentifierportId =
    CommPortIdentifier.getPortIdentifier("portName");
4 SerialPortserialPort = (SerialPort) portId.open("Demo application",
    5000);
```

Řádek 3 získá identifikátor portu, kterému odpovídá název uvedený v parametru „portName“. Následující řádek 4 otevře zvolený port s timeaoutem 5000ms. Samotná aplikace při navazování spojení dále ověřuje připojované zařízení pomocí testovací instrukce komunikačního protokolu DRAK 4. Pokud není detekováno správné zařízení, port se automaticky uzavře.

9 MODUL PRESSUREMONITORMODULE.JAVA

Modul odvozený z třídy `com.hella.appfw.app.AbstractModule` slouží pro zobrazení a zpracování provozních informací. Modul je navržen tak, aby mohl přijímat a zpracovávat současně 4 kanály. Každý z těchto kanálů obsahuje veřejnou přístupovou metodu, která umožní vložení provozních hodnot k vlastnímu zpracování. Díky takto navrženému modulu lze celý modul použít pro zobrazení a zpracování široké škály průběhů. Aktuálně je ve firmě tento modul využit pro zpracování provozních informací ze zařízení pro tvorbu světelných pouzder, ale pokud by bylo třeba zobrazovat průběhy odlišného charakteru, mělo by stačit v aplikaci vhodně přenastavit parametry modulu.



Obrázek 16: Diagram log. členění tříd modulu „PressureMonitorModule.java“

Stejně jako u předchozího modulu je zobrazen speciální typ diagramu, který se snaží jednoduše zobrazit návrh logického členění tříd a ukázat probíhající komunikaci mezi jednotlivými třídami a navrhovaným modulem.

Při inicializaci modulu se vytvoří příslušné akce, z kterých se vytvoří v aplikaci menu položky. V inicializační části je též vytvořen objekt, který je instancí modulu pro připojení k databázi. Podle požadavků firmy byl využit databázový modul implementovaný v „AppFrameworku“. Pro nastavení parametrů připojení k databázi slouží opět

preferencespanel. Pokus o připojení či odpojení k databázi je opět realizovaný pomocí zařízení, pracující na stejném principu jako zařízení pro připojení COM portu.

Pro odchycení událostí při připojování a odpojování databáze je využit DeviceListener odvozený z com.hella.appfw.device.DeviceListener. Ten implementuje abstraktní metody rozlišující následující stavy:

- Připojování: void initializing(DeviceEventvt)
- Připojeno: void initialized(DeviceEventvt)
- Odpojování: void closing(DeviceEventvt)
- Odpojeno: void closed(DeviceEventvt)

Metoda definující stav připojeno provádí kontrolu existence potřebných tabulek. Existenci tabulek zkontroluje objekt obsahující metadata databáze, ke které je aplikace připojena. Tyto metadata zahrnují informace o databázových tabulkách. Pokud některá z tabulek neexistuje je vytvořena dle navržené databázové struktury definované v ERD. Skripty pro vytvoření celé struktury tabulek implementuje třída „CreateTables“. Pro efektivní a bezpečnou práci s daty je implementováno omezení mezi tabulkami „Measurement“ a „MeasureValues“.

```
1 public final static StringalterTable =  
2     "ALTER TABLE APP.MEASUREVALUES ADD CONSTRAINT "  
3     + " MEASUREVALUES_MEASUREMENT_FK FOREIGN KEY (ID_MEASURE) "  
4     + "REFERENCES APP.MEASUREMENT (ID_MEASURE) "  
5     + "ON DELETE CASCADE ON UPDATE RESTRICT";
```

Provedení předchozího skriptu zaručuje, že při odstranění konkrétního měření z tabulky MEASUREMENT identifikované pomocí ID_MEASURE, se automaticky odstraní všechny jeho hodnoty uložené v tabulce MEASUREVALUES. Skript také zvyšuje bezpečnost pomocí omezení, které znemožňuje přidat hodnotu do tabulky MEASUREVALUES, která by byla identifikována neexistujícím měřením, neboli neplatným ID_MEASURE.

9.1 Práce s databází

V prvotní fázi byla do projektu zahrnuta třída umožňující práci s databází. Po požadavku firmy a následné společné spolupráci byl navrhnout univerzální modul pro práci s Derby databází. Tento modul se však umístil mimo tuto aplikaci a implementoval

se do samotného „AppFrameworku“. Modul byl rozšířen o preferences panel a zařízení, přes které probíhá připojování k databázi. Následně budou uvedeny klíčové metody a ukázky kódů, které lze využít při práci s databází.

9.1.1 Připojení k databázi

Před samotným připojením k databázi je třeba získat požadovaný driver. Java SQL Framework umožňuje přistupovat na více databázových driverů. Tato aplikace používá driver „org.apache.derby.jdbc.EmbeddedDriver“. Pro načtení ovladače slouží DriverManager, který zaregistruje instanci driver class. To znamená, že uživatel může načíst a zaregistrovat driver pomocí příkazu uvedeného na řádku 2. Řádek 3 nám pak zajistí připojení k požadované databázi.

```
1 java.sql.Connection connection;  
2 Class.forName(driver).newInstance();  
3 connection = DriverManager.getConnection( URL, userName, password);
```

Pro připojení jsou definované 3 parametry. První parametr URL je tvořen z protokolu (jdbc:derby:) a jména databáze (historyDB;create=true). Výsledné URL je pro tuto aplikaci složeno z následujícího řetězce „jdbc:derby:historyDB;create=true“. Poslední dva parametry userName a password jsou nepovinné a využívají se pro logovací údaje. Pro účely této aplikace nebylo třeba zabezpečit přístup pomocí loginu.

9.1.2 Struktura SQL dotazů

Existuje několik možností jak vytvářet samotné SQL dotazy. Tato aplikace se snažila co nejvíce využít takovou strukturu dotazů, která je vhodná a efektivní při častějším provádění stejného dotazu s rozdílnými parametry. Následuje zjednodušená demonstrace použití SQL dotazu.

```
1 java.sql.PreparedStatement insertMeasurementStatement;  
2 insertMeasurementStatement =  
  dbConnectionModule.getConnection().prepareStatement("INSERT INTO  
  MEASUREMENT (CHANNEL, COEFFICIENT) VALUES (?,?)",  
  PreparedStatement.RETURN_GENERATED_KEYS);  
3 insertMeasurementStatement.setInt(1, valueChannel);  
4 insertMeasurementStatement.setDouble(2, valueCoefficient);  
5 insertMeasurementStatement.executeUpdate();  
6 java.sql.ResultSet rs =  
  insertMeasurementStatement.getGeneratedKeys();
```

Na řádku 1-2 je vytvořen objekt, který představuje předkompilovaný SQL dotaz. Takto přichystané dotazy lze pak poměrně jednoduše používat. V prvním kroku je třeba definovat

hodnoty jednotlivých atributů dotazu, které se dosadí za jednotlivé otazníky v předkompilované předloze. Dosazení hodnot představují řádky 3-4, které používají metodu set podle datového typu dosazované hodnoty. Jako první parametr se volí pozice atributu a druhý parametr dosadí hodnotu zvolenému atributu. Pro provedení celého dotazu slouží metoda na řádku 5 „executeUpdate()“.

V této ukázce je záměrně předveden také způsob získání automaticky generovaných klíčů. Řádek č. 6 pomocí metody getGeneratedKeys() získá seznam všech generovaných klíčů. V aplikaci je těchto údajů využito pro získání posledního generovaného klíče, který je na poslední pozici v seznamu. Tato informace je použita pro hodnotu cizího klíče v jiné tabulce „MEASUREVALUES“.

9.2 PressureMonitorPanel.java

Tento panel slouží pro samotné zobrazení naměřených dat. Data jsou interpretovaná ve dvou různých podobách. První reprezentace naměřených dat představuje aktuální hodnotu napětí neboli poslední naměřenou hodnotu. Druhý způsob prezentace naměřených dat je v podobě grafu, zobrazující průběh tlaku vypočteného z napětí. Důvodem rozdílného zobrazení je požadavek seřizovacích pracovníků, pro které je aplikace převážně určena. Při diagnostice, výpadku či tvorbě zmetku pracovníka zajímá aktuální hodnota napětí vyjádřena v číselné podobě. Naopak grafické zobrazení může dát informaci o tom v jaké fázi tvorby výlisku se projevuje chyba. Vliv tvaru křivky na jakost byla již zmíněna v teoretické části práce.

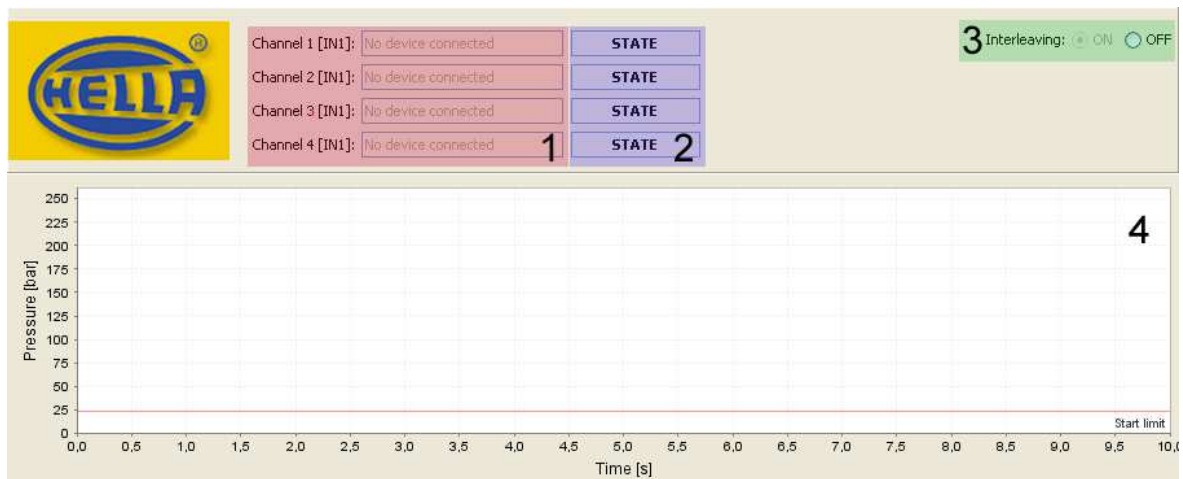
Každý kanál v panelu je rozšířen o status box, jenž pomocí různých stavů jednoduše graficky vyhodnocuje kvalitu každého měření. Pro ještě přehlednější zobrazení je každému stavu přiřazena barva. Následující tabulka vyjadřuje stavy s popiskem a barvou. Volba jednotlivých barev je odvozena ze zavedených zvyklostí a pravidel v této firmě.

Status	Popisek	Barva
0	STATE	STATE
1	OK	OK
2	WARN	WARN
3	BAD	BAD

Tabulka 5: Parametry stavů komponenty „StatusBox“

9.2.1 Návrh GUI

Návrhu grafického uživatelského rozhraní vychází převážně z požadavků koncových uživatelů. Výsledný návrh je vidět na následném obrázku.



Obrázek 17: GUI PressureMonitorPanel.java

Z předchozího návrhu vidíme rozložení jednotlivých částí panelu. Barevný blok č. 1 je určen pro hodnoty napětí. V bloku č. 2 jsou umístěny jednotlivé „statusBoxy“. Blok č. 3 umožňuje nastavit prokládání. Graf z bloku č. 4 tedy může zobrazovat průběh pomocí bodů, nebo jednotlivými body proložit křivku.

9.2.2 Tvorba grafu

Graf byl vytvořen pomocí knihovny jFreeChart. Následně je uveden příklad vytvoření a zobrazení grafu do `avax.swing.JPanelu`. Tuto základní strukturu v aplikaci dále rozšiřuje o specifické nastavení některých parametrů převážně pro renderer či definici bodů v případě vypnutí prokládání.

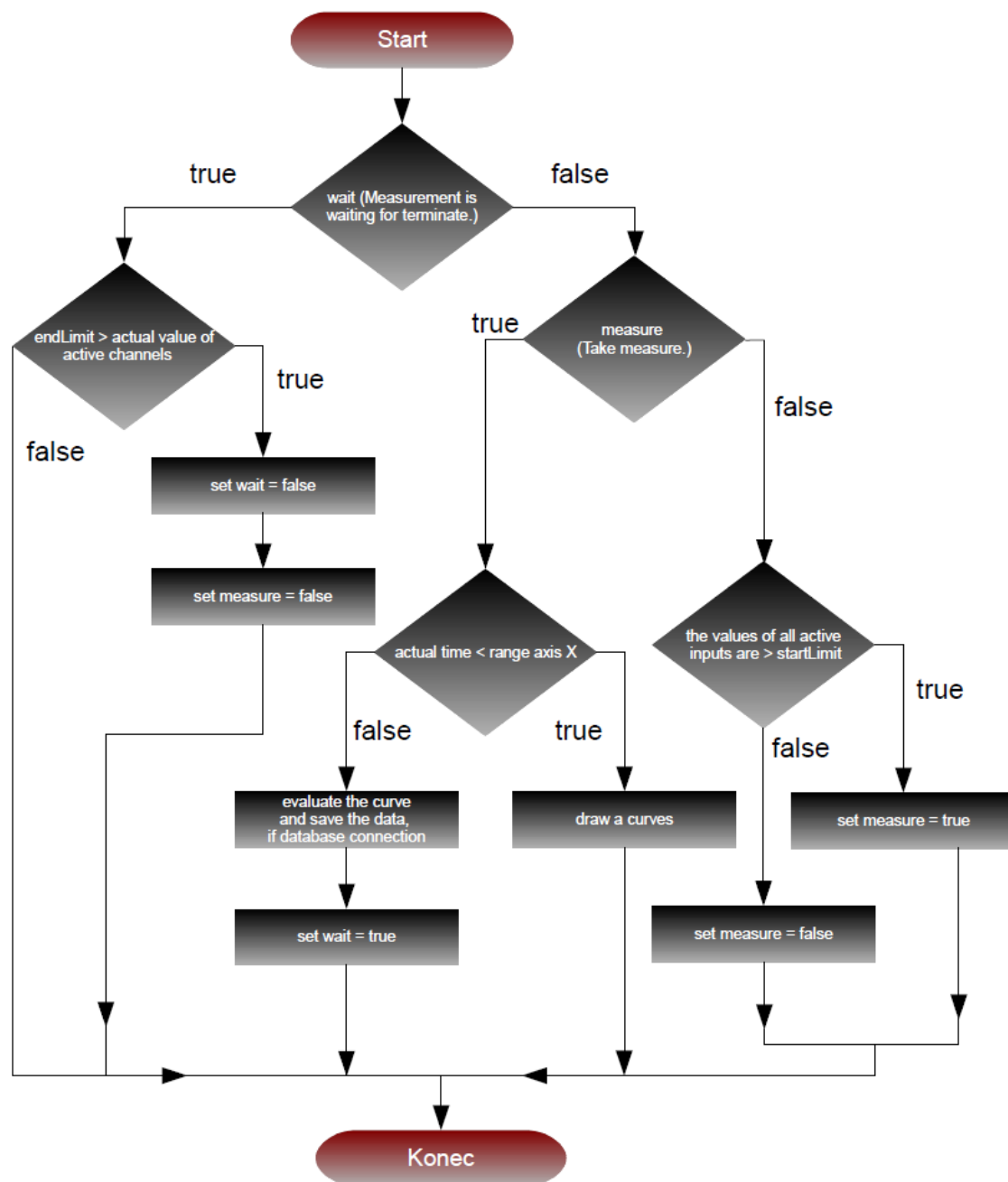
```
1 import org.jfree.chart.renderer.xy.XYSplineRenderer;
2 import org.jfree.chart.axis.NumberAxis;
3 import org.jfree.data.xy.DefaultXYDataset;
4 import org.jfree.chart.plot.XYPlot;
5 import org.jfree.chart.JFreeChart;
6 import org.jfree.chart.ChartPanel;
7
8 XYSplineRenderer renderer = new XYSplineRenderer();
9 NumberAxis axisTime = new NumberAxis("Time [s]");
10 NumberAxis axisPressure = new NumberAxis("Pressure [bar]");
11 DefaultXYDataset dataset = new DefaultXYDataset();
12 XYPlot plot = new XYPlot(dataset, axisTime, axisPressure, renderer);
13 JFreeChart chart = new JFreeChart(plot);
```

```
13 ChartPanel panel = newChartPanel(chart);
14 axisTime.setRange(double lower, double upper);
15 axisPressure.setRange(double lower, double upper);
16 panel.setVisible(true);
17 avax.swing.JPaneluchartPanel.add( panel);
```

Nejprve řádky 1-6 importují použité třídy. Řádek 7 vytvoří renderer pro proložení bodů křivkou. Řádky 8-9 vytvoří číselné osy, kterým je přidělen rozsah na řádcích 14-15. Dataset vytvořený na řádku 10 se používá pro přidání průběhu, který chceme zobrazit. Zobrazený průběh musí být popsán pomocí dvourozměrného pole typu double. Pole musí mít definovanou strukturu v podobě matice o dvou řádcích a počtem sloupců shodný s počtem zobrazených bodů. První řádek pak představuje souřadnice osy X a druhý řádek souřadnice osy Y. Výsledný bod vzniká jako průsečík těchto souřadnic v prostoru. Řádek 13 vytvoří samotný graf, který na řádku 17 zobrazíme do panelu na požadovaném místě v naší aplikaci.

9.2.3 Algoritmus pro vykreslení

Algoritmus popisující Flow chart diagram je generován v pravidelných intervalech odpovídající nastavenému time stepu. O realizaci této časové úlohy se stará časovač odvozený z třídy javax.swing.Timer. Vzhledem k charakteru použití aplikace bylo třeba z důvodu synchronizace více křivek a doby trvání samotného měření rozlišovat startovací a ukončovací limity. Při samotném startu vykreslování se čeká na společný začátek, dokud všechny aktivní křivky nepřekročí stanovený limit pro start. Po startu se aktivní průběhy vykreslují do té doby, než překročí rozsah časové osy X. Následně se čeká na dokončení samotného měření do doby, než měřené hodnoty dosáhnou hodnoty menší než je ukončovací limit. Během čekání na ukončení se provede vyhodnocení korelační analýzy a v případě aktivního spojení s databází uložení dat.



Obrázek 18: Flow diagram algoritmu vykreslování

9.2.4 Pravidlo vyhodnocení korelační analýzy

Pokud je koeficient korelace větší, nebo roven hodnotě korelačního limitu „OK“ je křivka generovaná stavem „OK“. Hodnoty v rozmezích „OK“ a „WARN“ jsou symbolizované jako „WARN“. Stav „BAD“ nastane v případě koeficientu menším, nebo rovno hodnotě „WARN“. Nastavení limitů pro vyhodnocení se nachází v panelu připadající tomuto modulu.

9.3 PressureMonitorPreferences.java

Panel je rozdělen do tří základních bloků. První část zvaná „Activechannel“ umožňuje nastavit ty kanály, které mají být měřeny. Možnost volby je od žádného do čtyř kanálů. Druhá část „Display chart parameters“ poskytuje možnost nastavit vykreslování křivek dle potřeby. Nastavit lze parametry jako rozsah časové osy X, rozsah tlakové osy Y, hodnotu konstanty určenou k výpočtu tlaku z naměřeného napětí. Při nastavení na hodnotu jedna se vykreslují přímo měřené hodnoty. Jednotlivým křivkám lze definovat tloušťku čáry. Tato vlastnost je využita z důvodu dálkové vizuální kontroly, která je v reálném provozu často potřeba. Změnit lze také time step. U pokusu změny tohoto parametru je generováno varovné okno umožňující nastavit výchozí hodnotu (80ms), nebo potvrdit uživatelské nastavení. Změna jiné než výchozí hodnoty time stepu vede k nesprávné práci statistických analýz. Pro univerzální využití korelační analýzy lze upřesnit hodnoty, z kterých se provádí vyhodnocení jako stav „OK“, „WARN“ a „BAD“. Poslední nastavitelná položka definuje délku uložení dat v databázi. Tato kontrola se provede vždy při uložení nové křivky do databáze. V první fázi se kontrola prováděla při připojení k databázi. Aplikace však může běžet bez restartu i několik týdnů a skript pro update starých záznamů se nevykonával. Databáze pak obsahovala obrovské množství nepotřebných dat, která zabírala zbytečný prostor na disku. Poslední blok nazvaný „Channelnames“ dovoluje pojmenovat popisky grafu pro jednotlivé kanály.

9.3.1 Omezení hodnot

Pro zvýšení bezpečnosti byly některým parametrům omezeny možnosti nastavení. Toto omezení do určité míry omezuje oblast použití modulu, ale v případě potřeby lze jednoduše tyto omezení upravit ve zdrojovém kódu. Podle aktuálních požadavků a využití aplikace ve firmě lze toto omezení považovat za přínos.

Protože omezující podmínky se týkají GUI prvku `JSpinner`, vlastní omezení je realizováno pomocí `SpinnerModelu`. Výsledný seznam omezení hodnot u jednotlivých parametrů je znázorněn v následující tabulce.

Parametr	Dolní mez	Horní mez	Výchozí hodnota	Krok
Rozsah osy X	5.0	60.0	12	1.0
Rozsah osy Y	150.0	1000.0	350	1.0
Startovací hodnota	8	200	58	1.0
Konstanta pro výpočet tlaku	8	100	58	1.0
Time step	0.08	1	0.08 (sec)	0.01
Tloušťka křivky	2.0	9.0	3.0	1.0
Korelační limit stavu OK	0.05	0.99	0.95	0.01
Korelační limit stavu WARN	0.05	0.99	0.55	0.01
Počet dnů archivace	1.0	7.0	2	1.0

Tabulka 6: Omezení hodnot pro modul PressureMonitorModule.java

9.4 CorrelationAnalysis.java

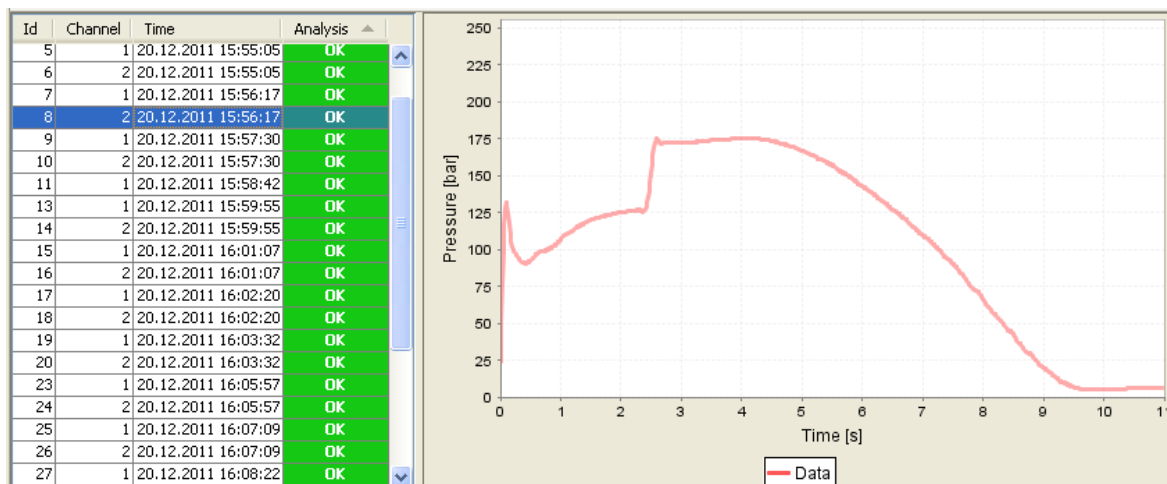
Třída zahrnuje veškeré funkce spojené s výpočtem korelačního koeficientu, vyjadřující míru shody při vzájemné korelační analýze. Mezi privátní zapouzdřené funkce patří výpočet střední hodnoty, rozptylu a směrodatné odchylky vyžadující jeden parametr definovaný jako jednorozměrné pole typu double. Funkce korelačního koeficientu je veřejná a samotný výpočet porovnává dva stejně velké soubory dat. Z tohoto důvodu je do vlastní funkce implementována standardizace velikosti jednotlivých souborů dat. Například při porovnání 15 sekundového průběhu s 12 sekundovým se provede vyhodnocení pouze úseku do 12 sekund, protože další data pro srovnání nejsou k dispozici. Stejně tak to platí v opačném případě, kdybychom chtěli porovnávat kratší průběh s delším. Volba délky výchozího průběhu je tedy také důležitá. Aplikace však obsahuje možnost volby výchozího průběhu za běhu aplikace. Tato možnost umožní vhodně nastavit korelační analýzu přímo na konkrétní problém. Průběhy a doby měření se totiž na jednotlivých strojích mohou lišit.

9.5 HistoryMonitorPanel.java

Jelikož do databáze jsou ukládána jednotlivá zaznamenaná data pro případné pozdější potřeby, bylo vhodné umožnit zobrazení těchto dat uložených v databázi. Možnost prohlížení dat je realizována právě pomocí přídatného panelu historyMonitorPanel.java. Panel je dostupný vždy, když je načtený modul pressureMonitorModule.java. Nezbytnou podmínkou pro práci s tímto panelem je aktivní spojení s databází. S ovládáním historie je spojeno několik menu položek s dialogem pro filtrované vyhledávání.

9.5.1 Návrh GUI

Panel je rozdělen na dvě základní části, kde levá polovina představuje oblast pro seznam vyfiltrovaných záznamů a pravá polovina vykresluje aktivní vybraný záznam v podobě grafu. Tento panel zobrazuje níže uvedený obrázek.



Obrázek 19: GUI historyMonitorPanel.java

9.5.2 Tvorba tabulky pro výpis

Vlastní tabulka je implementovaná pomocí ClassTableModelu odvozeného z com.hella.appfw.table.ClassTableModel. Pro získání přehledu o práci s ClassTableModelem je zde uveden příklad možné implementace. Pro jednoduchost a názornost ukázky nebude uvedena tvorba všech položek obsažená v této aplikaci. Prvním krokem je vytvořena pomocná třída pro uložení samotných dat.

```

1 class ResultInfo {
2     public Datetime;
3     public ResultInfo(Datetime) {
4         this.time = time;
5     }
6 }

```

Ve vlastní aplikaci každému sloupci tabulky odpovídá jeden atribut z této třídy. Protože vlastních dat je zpravidla zobrazeno větší množství, jsou jednotlivé záznamy ukládány do seznamu java.util.List. Následující příklad vytváří tento seznam.

```

1 List<ResultInfo>results = new ArrayList<ResultInfo>();

```

Nyní následuje vlastní definice ClassTableModelu prezentující datovou strukturu vytvořené pomocné třídy:

```

1 ClassTableModel<ResultInfo>resultsModel =
  newClassTableModel<ResultInfo>() {
2 @Override
3   public voiddefineColumns() {
4   defineColumn(newDateColumn<ResultInfo>(this, "Time") {
5   {
6   this.setEditable(false);
7   this.setRendererFormat(DateFormat.getDateTimeInstance(DateFormat.MED
      IUM, DateFormat.MEDIUM));
8       }
9       @Override
10      public DategetValue(ResultInfoobject) {
11          return object.time;
12      }
13      @Override
14      public voidsetValue(ResultInfoobject, Datevalue){
15      }
16  });
17  }
18  };

```

ClassTableModel pro definici jednotlivých sloupců tabulky, vyžaduje implementaci abstraktní metody defineColumns() viz. řádek č. 3. Tělo této metody volá metodu se stejným názvem, která vytvoří samotný sloupec s předem definovaným datovým typem. Tento krok je znázorněn na řádce č. 4, kde je vytvořen sloupec jménem „Time“ s datovým typem „Date“. Řádky č. 5-7 představují konstruktor, v kterém se zakazuje možnost editace buněk tohoto sloupce a nastaví se formát času. Řádky č. 9-15 implementují povinné abstraktní metody pro nastavení a získání hodnot z buněk. Za pozornost stojí všimnout si provázání s třídou ResultInfo.

Takto předem vytvořená struktura tabulky se jednoduše zobrazí ve vytvořeném GUI panelu příkazem:

```
1 panelForTable.setClassModel(resultModel);
```

Pro prezentaci vlastních dat nejprve řádek č. 1 naplní seznam hodnotami a následně řádek č. 2 data zobrazí do tabulky.

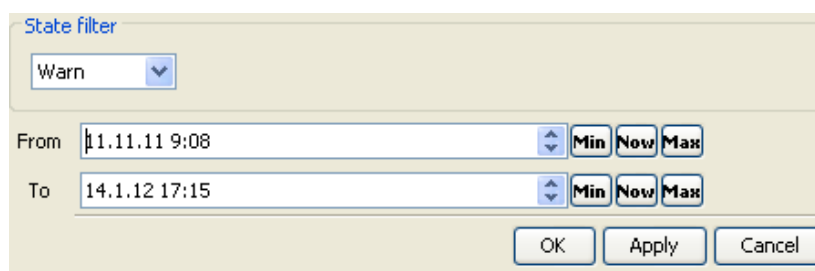
```

1 results.add(newResultInfo( newDate(„Yourtime“)));
2 resultsModel.setDataList(results);

```

9.5.3 Filtr pro vyhledávání

Při požadavku na vyhledání starých záznamů slouží `javax.swing.JDialog`. Tento dialog je vyvolán z menu nabídky „History“ → „History interval“. Vyhledávat lze podle několika kritérií. První kritérium umožní třídit záznamy dle zaznamenaného času v intervalu „od“, „do“. Při vyvolání dialogu se automaticky do příslušných komponent pro nastavení času načte do položky „od“ čas o dvě hodiny menší, než je čas aktuální. Horní interval „do“ obsahuje aktuální čas. Druhé kritérium rozlišuje záznamy v závislosti na vyhodnocení korelačního koeficientu. Rozlišují se stavy „All“, „WARN“ a „BAD“. Přičemž logika je následující. Při zvoleném filtru „ALL“ projdou všechny záznamy. Při filtru „WARN“ se zobrazí záznamy „WARN“ a všechny horší, tedy i záznamy „BAD“. Při nastaveném filtru „BAD“ budou vybrány záznamy pouze se stavem „BAD“, protože horší záznamy už nejsou. Rozdělení do jednotlivých kategorií se vyhodnotí podle aktuálně nastavených mezí pro intervaly korelační analýzy a záznamy odpovídající podmínkám nastaveného filtru se načtou do tabulky v `historyMonitorPanel.java`. GUI dialogu je vidět na následujícím obrázku.



Obrázek 20: GUI `historyIntervalDialog.java`

K zefektivnění práce s historií je navržena akce pro refresh, které je přiřazena přístupová klávesa F5. Při vyvolání této akce se provede aktualizace seznamu vybraných záznamů při zachování předem nastavených podmínek filtrace, kde se přenastaví pouze parametr pro časový interval „do“ na aktuální čas. Této akce je značně využíváno při prohlížení historie za současného záznamu nových dat, kdy uživatel nemusí neustále nastavovat filtr přes dialog a pouze klikne na klávesu F5.

9.5.4 Zobrazení křivky

Křivka se vykreslí do grafu právě tehdy, když vybereme odpovídající záznam z tabulky. Pro detekci výběru je využito odchycení události vyvolané při změně výběru hodnoty v tabulce. V následující ukázce zdrojového kódu vidíme odchycení této události.

```
1 tablePanel.getTable().getSelectionModel().addListSelectionListener(  
    newListSelectionListener() {  
2     @Override  
3     public void valueChanged(ListSelectionEvent e) {  
4         showDataToChart();  
5     }  
6 });
```

Při změně výběru je vždy vykonán obsah těla abstraktní metody „valueChanged“, kde se zavolá metoda pro vykreslení křivky do grafu. Aktuálně zvolený záznam je pak identifikován pomocí ID vybraného řádku. Toto ID zajistí metoda obsahující následující příkaz:

```
1 tablePanel.getTable().getSelectedRow()
```

Samotná data jsou získávána z databáze, proto při každém požadavku na změnu vykreslované křivky je potřeba aktivní spojení s databází. Stejně tak při aktualizaci filtru pro výběr záznamů je databázové spojení nezbytné.

9.5.5 Aktualizace výchozích dat pro korelační analýzu

Pro univerzálnější využití korelační analýzy jde aktualizovat a tím přenastavit výchozí data pro výpočet korelační analýzy. Jako nový vzorový soubor lze použít kteroukoli křivku zaznamenanou touto aplikací. Postup pro aktualizaci je pak následující. Při nasazení aplikace na jiný stroj si uživatel nechá zaznamenat a uložit průběh do databáze. Tento průběh si zobrazí v panelu pro historii. Ze zaznamenaných průběhů zvolí takový průběh, který bude uživatel považovat za vzorový. Pomocí menu nabídky pro změnu dat pro korelační analýzu, pak program přepíše data v tabulce DEFAULTVALUES.

10 ZÁVĚREČNÉ FÁZE PROJEKTU

10.1 Tvorba spustitelného (exe) souboru pomocí Launch4j

Pro vytvoření spustitelného exe souboru z projektu této aplikace je použit projekt Launch4j. Tento projekt lze zdarma stáhnout ze stránek <http://launch4j.sourceforge.net/>. Launch4j obsahuje vlastní GUI, z kterého lze nastavit potřebné informace a následně vytvořit spustitelný exe soubor. Nyní bude následovat vzorová ukázka vytvoření exe souboru s minimálními požadovanými informacemi nutnými k vygenerování tohoto souboru.

V prvním kroku je třeba nechat provést nad daným projektem „Build“, který je dostupný v nabídce zobrazené po kliknutí pravým tlačítkem myši nad daným projektem. Při této akci se vytvoří v aktuálním adresáři složka `../dist`, do které se vygeneruje spustitelný (Executeable) Jar soubor našeho projektu a dále se automaticky zkopírují všechny potřebné knihovny do podsložky `../dist/lib`.

Nyní jsou vytvořeny veškeré potřebné podklady pro Launch4j. Po spuštění GUI prostředí je vidět několik záložek, pod kterými lze nastavit poměrně velké množství informací specifikující exe soubor. Mezi povinné údaje patří v záložce „Basic“ položky „Jar“, „Output file“ a „Java download URL“. Položka „Jar“ definuje cestu k vygenerovanému Executeable Jar souboru projektu. „Output file“ určuje cestu cílového exe souboru a poslední položka pro download je automaticky před vyplněná na <http://java.com/download>. Dále je nutné v záložce JRE vyplnit minimální verzi JRE potřebnou pro spuštění aplikace. Nyní stačí spustit samotný proces generování pomocí tlačítka „Buildwrapper“, který do zvolené cesty vytvoří spustitelný exe soubor. Při tomto základním nastavení je nutné ručně zkopírovat potřebné knihovny do adresáře, v kterém se nachází vygenerovaný exe soubor.

Druhou možností je opět využít výhody „AppFrameworku“ a provést samotné vygenerování přímo z prostředí Netbeans. Tento způsob oproti předchozímu řešení automaticky přilinkuje potřebné knihovny do cílového adresáře, vytvoří grafickou ikonu exe souboru a zobrazí Hella splash soubor po spuštění aplikace. Veškerá data jsou vygenerována do složky `../dist-app` v adresáři projektu. Tento proces vyžaduje nastavenou cestu k Launch4j v „Tools“ → „Options“ → „Miscellaneous“ → „Hella Tools“. Vlastní generování se spustí po kliknutí na položku „Run Target“ → „make-dist“, přístupnou v nabídce vyvolané při stisku pravého tlačítka myši na soubor `appfwtools.xml`.

10.2 Testování aplikace

První verze aplikace byly odladěny v simulovaném prostředí, kdy firma umožnila zapůjčení A/D převodníku firmy Papouch. K tomuto převodníku pomocí transformátoru a potenciometru připojeného na vstup převodníku se simulovaly příchozí data a komunikace přes protokol DRAK 4. Před nasazením aplikace do provozu ji bylo třeba řádně odladit i v reálném prostředí na skutečných datech. V první fázi testování se kontrolovaly konektivity databáze a připojení přes COM port.

Velká pozornost se při testování věnovala korelační analýze, kdy nebylo předem jisté, zda bude tato analýza fungovat na vyhodnocení reálných dat. Z tohoto důvodu byla aplikace navržena co nejvíce univerzálně a umožňuje přes příslušnou nabídku „Preferences“ nastavovat většinu důležitých hodnot. Po vhodném nastavení těchto hodnot korelační analýza dosahovala velmi uspokojivých výsledků, kdy vyhodnocovala shodnost křivek přes 96%, tedy koeficient korelace byl roven hodnotám lepším než 0,96. Stejně tak při úmyslném záznamu neodpovídající křivky, byla korelační analýza schopna tuto křivku detekovat a vyhodnotit jako špatnou. Doporučené hodnoty jsou uvedeny v sloupci výchozí hodnoty tabulky „Omezení hodnot pro modul PressureMonitorModule“ uvedené v kapitole PressureMonitorPreferences.java.

Drobné změny dosáhly i na úpravu GUI, kdy při uživatelském testování některé důležité prvky nebyly dostatečně viditelné z větší dálky. Zvětšila se tak plocha pro graf, zpřístupnila možnost nastavit tloušťku křivek a barevně vyhodnotit výsledky korelační analýzy.

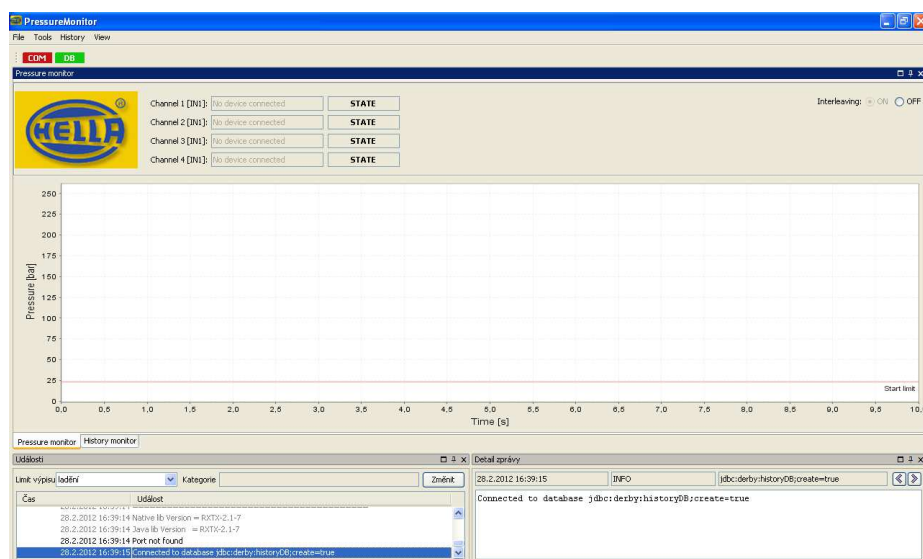
Dalším neméně důležitým úkolem bylo nechat aplikaci obsluhovat pracovníky a pozorovat tak případně nedostatky v ovládání. Zde bylo odhaleno několik problémů, které zbytečně znepříjemňovaly práci s aplikací. Z tohoto důvodu byla několikrát upravena vlastní logika ovládání za účelem co největší automatizace, aby se nestávaly situace, kdy se například nastavily parametry pro připojení COM portu či databáze a v případě restartu či výpadku bylo třeba tyto parametry znovu ručně nastavit. Také tlačítka pro vlastní připojení se nastavily do režimu automatického pokusu o připojení v případě restartu a zapnutí aplikace.

10.3 Ovládání aplikace

Snahou bylo navrhnout aplikaci tak, aby její ovládání bylo co nejvíce intuitivní. Předchozí kapitoly do jisté míry již nastínilly ovládání aplikace, ale i přes to následuje krátké ucelené shrnutí ovládání.

Při prvním spuštění je třeba definovat vybrané moduly, s kterými aplikace chce pracovat. V tomto případě se jedná o moduly BucherSourceModule, PressureMonitorModule a DBconnectionModule. Zde je následně nutný restart aplikace, pro načtení nově přidanych modulů. Nyní jsou již v horní části viditelné a přístupné zařízení pro připojení k databázi a sériovému COM portu. Vždy při restartu se automaticky provede pokus o připojení k oběma těmto zařízením. Nastavení všech parametrů, nejen pro připojení, nalezneme v dialogovém okně „Tools“ → „Možnosti“.

Jednotlivé dostupné panely, které obsahují vybrané moduly, zobrazuje menu položka „View“. Minimalizovat, maximalizovat a zavřít panel realizují klasické ikonky pro práci s okny. Díky dokování při stisknutí levého tlačítka myši nad vybraným panelem a následným tažením uživatel jednotlivý panel umístí do různých pozic obrazovky. Takto vytvořené uživatelské rozvržení lze rovněž uložit v sekci „View“ → „Rozvržení“. Zde také v případě potřeby lze načíst dostupná rozvržení. Následuje ukázka možného nastavení uživatelského rozvržení.



Obrázek 21: Příklad uživatelského rozvržení aplikace

Pokud je aktivní spojení s COM portem, jsou zpracovávány provozní informace v příslušném panelu. Volbu či změnu aktivních kanálů, které mají být měřeny, nastaví odpovídající záložka v dialogu „Tools“ → „Možnosti“. Na této záložce se nalézá také

nastavení všech potřebných parametrů pro přesné doladění vykreslovaných průběhů a korelační analýzy. Panel pro vykreslení ve svém pravém horním rohu zahrnuje možnost zvolit prokládané či bodové vykreslení křivek.

Při korelační analýze se jako výchozí použije předdefinovaný průběh. Z tohoto důvodu je dobré, aby nejprve aplikace zaznamenala správný průběh, uložila si ho do databáze a následně si ho uživatel načetl do okna pro historii a pomocí „History“ → „Set correlation data“ přenastavil vzorová data. Před tímto krokem je důležité pamatovat na vhodné nastavení potřebných parametrů.

Ovládání při práci s historií je opět velmi jednoduché. Pokud je zobrazen panel historie, nejprve uživatel nastaví zvolené podmínky vyhledání dat. Tento dialog pro vyhledávání se nachází v „History“ → „History interval“. Pokud výběru odpovídaly nějaké záznamy, načetly se do tabulky umístěné v levé polovině tohoto panelu. Po kliknutí na záznam se vykreslí jemu odpovídající průběh. Pro aktualizaci seznamu lze použít klávesu F5. Vlastní tabulku pro výpis seznamu záznamů lze pomocí nabídky zobrazené po kliknutí pravým tlačítkem myši blíže specifikovat. Lze tak například upřesnit, které sloupce chceme, aby byly viditelné. Tím se dá přesněji doladit grafická podoba výpisu záznamů podle potřeb uživatele.

ZÁVĚR

Během fáze vývoje a testování aplikace prodělávala nejrůznější úpravy. Finální verze splňuje požadavky jak kladené firmou, tak požadavky definované v zadání této práce. Mezi výhody vytvořeného řešení bych určitě zařadil grafické rozvržení, kdy uživatel přehledně vidí potřebné informace, které ho nejvíce zajímají. Další výhodou je navržení oddělených modulů pro případ potřeby nahrazení komunikačního protokolu DRAK4. Tento protokol se totiž podle mého průzkumu stává zastaralým a v budoucí době ho bude třeba nahradit. Novější převodníky již také automaticky obsahují možnost komunikace přes USB, protože novější počítačové sestavy zpravidla již COM port neobsahují. Firma zatím ale vlastní dostatečné množství těchto starších převodníků, tak nad jejich výměnou neuvažuje. Také většina pracovních stanic, na kterých tato aplikace běží, jsou velmi staré a neobsahují USB port. U novějších stanic, které by neobsahovaly COM port, lze řešit připojení přes USB pomocí převodník USB to RS232, který výrobce A/D převodníků nabízí. Avšak cena tohoto kabelu je vyšší než zakoupení externí karty s COM porty. Za výhodu této aplikace bych také uvedl poměrně úspěšné aplikování korelační analýzy, kdy nebylo předem jisté, zda tato metoda bude na řešený problém fungovat.

Mezi problémová místa v aplikaci mohu z největší pravděpodobností zařadit nutnost nahrazení během několika let modulu pro komunikaci. Citlivým místem je také nastavení parametrů korelační analýzy, kdy v případě změny „time stepu“ nebude korelační analýza pracovat správně.

ZÁVĚR V ANGLIČTINĚ

During the development phase and testing, applications went through different updates. The final version fulfills the requirements of the company and the requirements defined in the terms of this work. Among the benefits of the created solution, I would certainly include graphical layout, where the user clearly sees the necessary information, what is the most interesting for him. Another advantage is the design of separate modules for the case of necessity to replace the communication protocol DRAK 4. This protocol will be obsolete in the future according to my research and we will need to replace it. The latest A/D converters also automatically include the ability to communicate via USB because newer computers don't usually have a COM port. The company currently owns enough older A/D converters, and is not considering replacing these older converters, where is most of the workstations, where the application is running, are very old, they do not contain a USB port. In the newer stations which do not contain a COM port, we can resolve connection via USB by using converter from USB to RS232, which the manufacturer of A/D converters offers. But the price of this cable is higher than buying an external card with COM ports. As to the benefit of this application, I can also mention the relative successful application of correlation analysis, when it was not certain if this method could have solved this problem.

Among to the drawbacks with this application, I can conclude the most likely is the necessity of replacement of the communication module during a few years. Weak point is also setting of parameters of correlation analysis, when in case of change "Time Step", correlation analysis does not work right.

SEZNAM POUŽITÉ LITERATURY

1. KRAUSS MAFFEI. *Technická dokumentace vstřikovacího stroje Kraus Maffei 300*. 2008.
2. HABERLANDOVÁ, Jana. HELLA AUTOTECHNIK, s.r.o. *Prezentace Hella Group*. Mohelnice, 2011.
3. Papouch. PAPOUCH, s.r.o. *Papouch* [online]. [cit. 2012-04-07]. Dostupné z: <http://www.papouch.com/cz/website/mainmenu/products/prevodniky/rs232/>
4. PAPOUCH, s.r.o. *Kompletní dokumentace měřicího přístroje Drak 4*. 2010. Dostupné z: www.papouch.com/cz/shop/product/drak4-laboratorni-ad-prevodnik/drak4.pdf/_downloadFile.php
5. PAPOUCH, s.r.o. *Dokumentace protokolu Drak4*. 2010. Dostupné z: www.papouch.com/cz/shop/product/drak4-laboratorni-ad-prevodnik/protokol-drak4.pdf/_downloadFile.php
6. MELOUN, Milan a Jiří MILITKÝ. *Statistická analýza experimentálních dat*. Vyd. 2. uprav. rozš. Praha: ACADEMIA, 2004, 953 s. ISBN 80-200-1254-0.
7. ŠPALEK, Jiří. MASARYKOVA UNIVERZITA, Ekonomicko-správní fakulta. *Skripta Aplikovaná statistika I*. Brno, 2003.
8. *RXTX: The prescription for rtransmission* [online]. [cit. 2012-04-07]. Dostupné z: <http://users.frii.com/jarvi/rxtx/>
9. *JFreeChart: The prescription for transmission* [online]. [cit. 2012-04-07]. Dostupné z: <http://www.jfree.org/jfreechart/>
10. *JFreeChart: Overview*. [online]. [cit. 2012-04-07]. Dostupné z: <http://www.jfree.org/jfreechart/api/javadoc/index.html>
11. *Apache Derby* [online]. [cit. 2012-04-07]. Dostupné z: <http://db.apache.org/derby/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EDD	Entitně relační diagram
Timeout	Překročení časového limitu.
PK	Primary Key.
PFK	Primary Foreign Key.
GNU	GNU's Not Unix.
LGPL	Lesser General Public License.
E(X)	Střední hodnota.
σ^2	Rozptyl.
σ	Směrodatná odchylka
\hat{C}_{xy}	Vzájemná kovariance.
\bar{R}	Koeficient korelace.
ASCII	American Standard Code for Information Interchange.
BNC	Bayonet Nut Coupler.
PNG	Portable Network Graphics.
JPEG	Joint Photographic Expert Group.
PDF	Portable Document Format.
EPS	Encapsulated Post Script.
SVG	Scalable Vector Graphics.
JVM	Java Virtual Machine.
JDK	Java Development Kit.

SEZNAM OBRÁZKŮ

Obrázek 1: Ovládací a indikační panel (1)	12
Obrázek 2: Výrobek v různých fázích vývoje (2).....	13
Obrázek 3: Náčrtek vstřikovacího stroje Krauss Maffei 300 (1)	14
Obrázek 4: Vliv průběhu tlaku na jakost	15
Obrázek 5: Blokové schéma DRAK 4	17
Obrázek 6: Ukázky vyhodnocení korelační analýzy	23
Obrázek 7: Ukázka různých typů grafů. (9)	25
Obrázek 8: Porovnání výsledků měření mezi dvěma aplikacemi.....	26
Obrázek 9: Dialogové okno pro nastavení parametrů v podnabídce Hella Tools	31
Obrázek 10: Dialogové okno pro správu menu nabídek.....	33
Obrázek 11: Ukázka implementace provázání proměnné s danou komponentou	34
Obrázek 12: Univerzální ERD databázové struktury	36
Obrázek 13: Diagram logického členění tříd modulu „BucerSourceModule.java“	37
Obrázek 14: Ukázka zařízení pro připojení k COM portu a databázi	38
Obrázek 15: GUI BucherSourcePreferences.java.....	38
Obrázek 16: Diagram log. členění tříd modulu „PressureMonitorModule.java“	40
Obrázek 17: GUI PressureMonitorPanel.java	44
Obrázek 18: Flow diagram algoritmu vykreslování	46
Obrázek 19: GUI historyMonitorPanel.java	49
Obrázek 20: GUI historyIntervalDialog.java.....	51
Obrázek 21: Příklad uživatelského rozvržení aplikace.....	55

SEZNAM TABULEK

Tabulka 1: Instrukce protokolu DRAK 4 (5).....	19
Tabulka 2: Obecné ohodnocení koeficientu korelace	22
Tabulka 3: Umístění knihoven v závislosti na OS.....	24
Tabulka 4: Datové typy Apache Derby	28
Tabulka 5: Parametry stavů komponenty „StatusBox“	43
Tabulka 6: Omezení hodnot pro modul PressureMonitorModule.java	48