VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VÝMĚNA DAT V AUTOMATIZOVANÉ VÝROBĚ

DATA TRANSFER IN AUTOMATION PRODUCTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAEL KONEČNÝ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. BRANISLAV LACKO, CSc.
SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Michael Konečný

který/která studuje v **bakalářském studijním programu**

obor: **Strojní inženýrství (2301R016)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Výměna dat v automatizované výrobě

v anglickém jazyce:

Data transfer in automation production

Stručná charakteristika problematiky úkolu:

Analyzovat problémy, které vznikají při spolupráci různých automatizačních prostředků a subsystémů v automatizované výrobě.

Cíle bakalářské práce:

1)Sestavit základní problémy výměny dat v automatizované výrobě

2)Popsat vybraný problémový případ a doporučit způsob řešení

Seznam odborné literatury:

Balátě, J.: Automatické řízení. BEN 2003 Praha

Lacko,B.-Beneš,P.-Maixner,L.-Šmejkal,L.: Automatizace a automatizační technika. Computer Press 2000 Praha

Švarc,I.: Automatizace a automatické řízení. CERM 2002 Brno

Vedoucí bakalářské práce: doc. Ing. Branislav Lacko, CSc.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2010/2011.

V Brně, dne 9.11.2010

L.S.

Ing. Jan Roupec, Ph.D. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc. Děkan fakulty

Abstrakt

Cílem této práce je stručně popsat základní problémy výměny dat v automatizované výrobě a poté popsat vybraný problémový případ a doporučit způsob řešení. Jako problémový případ byl vybrán požadavek firmy Indet Safety Systems, a. s. zřídit na jedné z jejích automatických linek zaznamenávání měřených dat a možnost jejich zpětné analýzy. Tento případ je podrobně popsán a je navrhnuto jeho kompletní řešení.

Klíčová slova

automatizace, přenos dat, apigraf, softwarové PLC, montážní linka

Abstract

The aim of this work is to briefly describe main problems of data transfer in automation production and then to descfibe a chosen problematic case and recommend its solution. As the problematic case was chosen the request of Indet Safety Systems, a. s. to come up with a solution that would allow storing and retro-analysis of data measured on produced parts by one of their automatic assembly lines. This case is thoroughly described and a complete solution is proposed.

Keywords

automation, data transfer, apigraf, software PLC, assembly line

KONEČNÝ, M. Výměna dat v automatizované výrobě. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. XY s. Vedoucí bakalářské práce doc. Ing. Branislav Lacko, CSc..

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma "Výměna dat v automatizované výrobě" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/200 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Branislavu Lackovi, CSc. za velmi promptní a praktické komentáře při konzultování práce, panu Ing. Štěpánovi Kopeckému za vytrvalou podporu při studiu linky a nekonečném testování nového software a panu Ing. Jiřímu Václavíkovi za zprostředkování možnosti řešení praktického problému.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Automatizací ve výrobě většinou rozumíme použití kontrolních systémů a informačních technologií za účelem snížení potřeby lidské práce při výrobě produktů [14]. Tím, že automatizace z velké části nahrazuje lidskou práci, umožňuje nám soustředit své síly na jiné potřebné úkoly, zvyšovat tak produktivitu a většinou i snižovat cenu výrobků. Kromě toho je třeba také říct, že k automatizaci vede snaha člověka osvobodit se nejen od činností fyzických, ale i od jednotvárných činností duševně unavujících [3]. Toto, a řada dalších, jsou důvody, proč je vhodné se automatizací zabývat a proč může být výhodné ji využít.

První nástup jejího většího rozvoje se datuje do doby industriální revoluce, kdy byl v roce 1801 patentován první automatizovaný tkalcovský stav používající děrné štítky. Další silnou vlnou rozvoje bylo období druhé poloviny 20. století, kdy se poprvé objevily počítače s dostatečným výkonem pro řízení opakovaných operací strojů [13, 3]. Dnes už vývoj hardwaru výpočetní techniky pokročil tak, že ve většině případů svými možnostmi překročila požadavky ve výrobní automatizaci a vývoj softwaru se soustředí více na jednoduchost použití programů, jejich komplexnost a na využívání nových možností jako je obrazová detekce apod. Předpokládá se, že kromě těchto faktorů umožní další vývoj mikroelektroniky pokračování zmenšování rozměrů a spotřeby elektrické energie, zvýšení spolehlivosti díky zabudování diagnostických funkcí, snižení ceny automatizačních prostředků a zkrácení doby návrhu a zavádění automatizace [2].

Vzhledem k tomu, že jsem měl osobně vždy blízko jak k mechanice - už jako malý jsem vždy všechno rozebíral, protože mě zajímalo, jak to funguje, tak k počítačům - můj otec má softwarovou firmu zabývající se vybavením pro praktické lékaře, je pro mě automatizace přirozeně blízkým oborem. Když se mi navíc naskytla možnost řešit praktickou část ve firmě, která má v našem kraji dobré jméno, neváhal jsem této příležitosti využít.

Kapitola 2

Cíle bakalářské práce

Cíli bakalářské práce bylo:

1. sestavit základní problémy výměny dat v automatizované výrobě
2. popsat vybraný problémový případ a doporučit způsob řešení.

Jako problémový případ byl řešen požadavek firmy Indet Safety Systems, a. s. (Vsetín), která na jedné ze svých automatických linek chtěla zřídit zaznamenávání dat měřených při výrobě pyrotechnických iniciátorů a vytvořit možnost jejich zpětné analýzy.

Kapitola 3

Základní problémy výměny dat v automatizované výrobě

Výměna dat v automatizované výrobě má stále větší a větší význam. Zatímco v dřívějších dobách byl přenos informací mezi různými stroji či subsystémy uskutečňován ručně, případně později pomocí přenosných výměnných médií, dnes je stále více využíváno automatického přenosu.

Jednotlivé subsystémy pak tedy tvoří *systém*, čímž tedy rozumíme soubor prvků, mezi nimiž existují vzájemné vztahy a jako celek má určité vztahy ke svému okolí [1].

Problematika komunikace v automatizovaných systémech se také velmi úzce pojí s pojmem *průmyslová sběrnice.*

Sběrnice je v počítačové architektuře subsystém, který přenáší data mezi komponenty počítače, nebo mezi jednotlivými počítačemi. První počítačové sběrnice byly doslova tvořeny paralelně zapojenými elektrickými vodiči, ale dnes se tento termín používá pro jakékoliv fyzické uspořádání, které zprostředkovává logicky stejnou funkcionalitu jako paralelní elektrická sběrnice. Moderní počítačové sběrnice mohou používat jak paralelní tak sériové zapojení, přičemž existuje více variant možných topologií [15].

Mezi nejznámější standardy počítačových sběrnic patří:

- ISA - starší typ pasivní sběrnice, šířka 8 nebo 16 bitů, přenosová rychlost < 8 MB/s
- PCI - novější typ „inteligentní“ sběrnice, šířka 32 nebo 64 bitů, burst režim, přenosová rychlost < 130 MB/s (260 MB/s)
- USB - sériová polyfunkční sběrnice, 2 diferenciální datové vodiče + 2 napájecí vodiče 5 V/500 mA, široké použití
  verze 1.1 - “full-speed” - přenosová rychlost 12 Mb/s (˜1,43 MB/s)
  verze 2.0 - “high-speed” - přenosová rychlost 480 Mb/s (˜57 MB/s)
  verze 3.0 - “SuperSpeed” - přenosová rychlost 4800 Mb/s (˜572 MB/s)

- FireWire - sériová polyfunkční sběrnice, široké použití, 50 MB/s
- RS-485 - sériová průmyslová sběrnice, (někdy jako proudová smyčka), do prostor s vysokým elektromagnetickým rušením [16]
- RS-232C - klasická sériová komunikace zejména pro automatické měřící přístroje

V automatizaci je však často potřeba zabezpečit odolnost sběrnic proti rušení a provedení kabeláže tak, aby odolávala vlivům průmyslového prostředí, jako může být prach, zvýšená teplota, chvění, agresivní plyny nebo kapalina, apod. Proto byla vyvinuta celá řada průmyslových sběrnic.

Jednou z nejznámějších takových sběrnic je fieldbus. Z výsledků nedávno ukončené studie spojené s výzkumem trhu průmyslových komunikačních systémů provedené organizací ARC Advisory Group vyplývá, že ve zpracovatelském průmyslu výrazně převládá právě použití průmyslové sběrnice fieldbus. Studie naznačuje, že prodané systémy vybavené touto sběrnicí se v současnosti podílejí na celkových tržbách v odvětví průmyslových komunikačních systémů až dvěma třetinami [12].

Kromě fieldbusu však byla vyvinuta i řada jiných průmyslových sběrnic. Patří mezi ně např. PROFINET, PROFIBUS, CAN-bus, průmyslový Ethernet a další.

Pokud automatizační prvky používají těchto mezinárodně standardizovaných sběrnic, je tím velmi usnadněno vzájemné propojení a snímání dat.

V řadě specifických případů je však potřeba řešit mnoho záležitostí individuálně. Jedná se většinou o řešení následujících problémů:

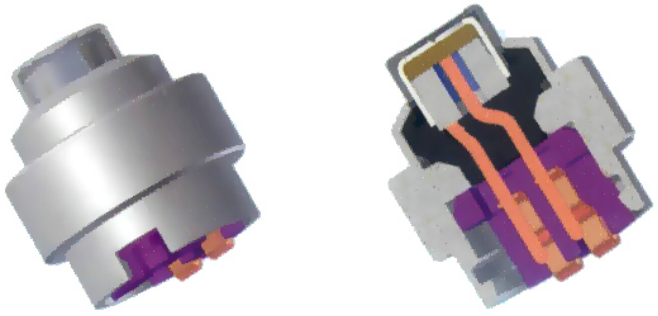
- propojení pomocí kabelů tak, aby na jednotlivých kontaktech byly odpovídající druhy signálů
- unifikace signálů z hlediska napětí, proudu, tvaru signálu, časové synchronizace a průběhu signálu
- způsob kódování údajů
- vzájemný protokol výměny dat

Kapitola 4

Popis zkoumaného problému

Firma Indet Safety Systems, a. s. vyvíjí a vyrábí pyrotechnické iniciační prvky pro použití v pasivních bezpečnostních systémech automobilů (airbagy, bezpečnostní pásy). Jedním z takových prvků je tzv. *load limiter*, který se kompletuje z jednotlivých částí právě na zkoumané lince.

Load limitery se používají v předpínačích bezpečnostních pásů (ty slouží k přitažení pásu o ca. 5 - 10 cm v případě nehody). Jsou iniciovány s malým časovým spožděním po předepnutí pásu a mají za úkol - přibližně v okamžiku, kdy tělo pasažéra vinou setrvačnosti působí na pás největší silou - pás mírně uvolnit a docílit tak delšího časového rozložení silového rázu na tělo, a tím zamezit např. zlomení žeber apod.



Obrázek 4.1: Load limiter

Jak již bylo řečeno, load limiter se kompletuje z několika částí. Hlavní částí je tzv. *squib*, což je uzavřený kalíšek naplněný pyrotechnickou směsí, ze kterého vedou dva elektrické kontakty. Při přivedení dostatečně velkého napětí na mezi kontakty se směs iniciuje a vybuchne. Tento squib je při kompletaci usazen do kovového držáku (*squib holder*), do kterého je následně zalisován ohnutím okraje squib holderu. Dále je do squib holderu vložen tzv. *shorting clip*, který k sobě vodivě spojí elektrické kontakty, a tím zamezí nechtěné iniciaci při manipulaci se součástkou, k jaké by mohlo dojít například i přeskočením "statické elektřiny".

Shorting clip je pochopitelně při montáži do auta odstraněn.

Po kompletaci load limiteru je nutné ověřit jeho správnou funkci. Proto se u každého vyrobeného kusu měří několik elektrických odporů - iniciační (bridge wire resistance), zkratovací (short-circuit resistance) a isolační (insulation resistance). Iniciační odpor musí být v pořádku, aby vždy došlo k zažehnutí směsi při přivedení iniciačního napětí, zkratovacím odporem se ověřuje, zda je správně vložen shorting clip a zda plní svou funkci a hodnota isolačního odporu značí, zda je v pořádku isolace mezi elektrickým kontaktem a vnějším kovovým držákem squibu (squib holderem). Naměřené hodnoty se musí vždy nacházet v předem vymezených mezích, pokud tomu tak není, kus je vyřazen.

Právě tyto tři měřené hodnoty jsou ty, které chtěla firma zaznamenávat.

4.1 Linka Mikron Flexcell Century 21 #078314

Celá součást load limiteru se skládá automaticky na lince Flexcell Century 21 švýcarské výroby firmy Mikron. Linka sestává z pracovní části, dopravního pásu, paletizéru, zásobníků dílů a řídící elektroniky.

Vlastní pracovní část je tvořena 21 stanicemi, na každé stanici se provádí určitá operace.

Skládané součásti jsou kompletovány a mezi stanicemi dopravovány ve speciálních paletkách, které jsou posouvány dopravním pásem. Ten z pracovní oblasti linky pokračuje dál do paletizéru, kde jsou součásti vyjímány a přeskládávány do polystyrenových palet určených pro přepravu. Prázdné paletky dále pokračují po pásu zpět na začátek linky.

Na stanicích, kde se k sestavě přidává další díl, je přísun dílů zajištěn vibračními zásobníky, z kterých jsou díly v řadě za sebou podávány na příslušné místo přes podavací lišty. Pokud se blíží prázdný stav zásobníku, je toto signalizováno na něm umístěným varovným světlem a obsluha díly doplní.

Vzadu za pracovní částí se nachází hlavní část řídící elektroniky, tvořená potřebnou kabeláží od čidel, průmyslovým PC, na kterém běží řídící software, porty pro případnou komunikaci s dalšími externími zařízeními, PC pro řízení laserového popisovače a modemem.

Připojení k řídícímu PC je možné lokálně pomocí ethernet portu přes crossover kabel nebo vzdáleně přes 56kbps modem (pro řešení jednoduchých úprav, případně řešení naléhavých situací). Pro výměnu dat lze také využít sériového portu (standard RS232).

4.2 Mechanické provedení linky

Hlavní mechanickou částí linky je jedna hřídel, umístěna za pracovní oblastí, podél pracovních stanic, která sahá přes celou délku pracovní oblasti. Na této hřídeli jsou umístěny vačky, jedna pro každou stanici, které ovládají mechanické prvky na jednotlivých stanicích. Hřídel se při

Obrázek 4.2: Linka Flexcell Century 21 #078314

automatickém provozu otáčí konstantní úhlovou rychlostí a úhel jejího natočení je snímán enkodérem a notně využíván řídícím programem. Ten z něj určuje např. pracovní fáze senzorů, zda v daném okamžiku najíždí daný nástroj, zda by měla být už určitá operace dokončena apod.

Toto uspořádání - mechanické propojení všech stanic jednou hřídeli - je velmi výhodné, protože je ním zabezpečena téměř dokonalá synchronizace stanic vůči sobě. Není ji tedy třeba potom řešit softwarově (či jinak elektronicky) a v softwaru stačí ohlídat, zda každá stanice během cyklu stihla dokončit požadovanou operaci.

4.2.1 Operace jednotlivých stanic

Na většině stanic je prováděna určitá operace. Některé stanice jsou, z důvodu celkového prostorového uspořádání částí linky, bez operace a paletka s výrobkem jimi jen projíždí. Zde jsou popsány operace jednotlivých stanic:

Stanice 1 ověř stav paletky a očisti ji

Prázdnost paletky je ověřena jednotkou s indukčním čidlem. Pokud je v paletce detekován cizí objekt, linka je zastavena a musí být uvedena zpět do provozu operátorem, který paletku zkontroluje.

Čištění paletky probíhá pomocí zařízení, které svou překryje celou plochu paletky a vysaje z ní případné nečistoty do sběrného vaku.

Stanice 2 bez operace

Stanice 3 podej, zajisti správnou orientaci a ulož do paletky squib holder

Squib holdery jsou podávány vibračním podavačem. Přenos součásti je zajištěn přenosovým nástrojem s pneumatickou uchopovací hlavou.

Součást je nejprve přenesena do "mezipozice", kde je pomocí motorku natočena do potřebné pozice. V dalším cyklu je druhou uchopovací hlavou přenesena do paletky.

Stanice 4 zkontroluj přítomnost squib holderu a jeho pozici

Stanice 5 umísti vnitřní O-kroužek

Součásti jsou podávány vibračním podavačem a přenášeny na výslednou pozici pomocí pneumaticky ovládané podavací hlavy.

Stanice 6 zkontroluj přítomnost a pozici O-kroužku

Přítomnost a pozice O-kroužku je kontrolována LVDT sondou Sony, která změří výšku, v které se nachází horní část O-kroužku. Pokud není změřený údaj v daných mezích, je součást označena jako vadná.

Stanice 7 umísti squib

Součásti jsou podávány vibračním podavačem. Nejprve jsou umístěny do mezipozice, kde jsou natočeny do potřebné pozice, a v dalším cyklu jsou přeneseny pneumatickou hlavou do squib holderu.

Stanice 8 ověř přítomnost a pozici squibu

Stanice 9 zacrimpuj squib do squib holderu

Squib je do squib holderu zalisován pomocí ohnutí části stěny squib holderu (*crimping*). Toto je provedeno pomocí hydro-pneumatického lisu, silou < 20 kN. Před lisováním je na nástroj nanesen lubrikant pomocí mlhového rozstřikovače.

Stanice 10 zkontroluj výšku zacrimpování

Výška zacrimpování je kontrolována pomocí standardní jednotky s LVDT sondou Sony. Pokud změřená výška není v mezích, je součást označena jako špatná.

Stanice 11 bez operace

Stanice 12 umísti vnější O-kroužek

Součásti jsou podávány vibračním podavačem. Jsou umístěny na svou pozici pomocí nástroje s pneumaticky ovládanou podavací hlavou.

Stanice 13 ověř přítomnost a pozici O-kroužku

Přítomnost je kontrolována pomocí standardní jednotky s LVDT sondou Sony. Pokud změřená vzdálenost není v mezích, je součást označena jako špatná.

Stanice 14 otoč celou součást o 180 a přenes ji do druhého hnízda paletky a ověř její přítomnost

Součást je otočena spodní částí nahoru, aby k ní byl umožněn přístup pro další operace. Přítomnost v druhém hnízdě je ověřena pomocí světelného čidla.

Stanice 15 zorientuj a umísti shorting clip pro výrobní variantu 2 a 3

Shorting clip je nejprve umístěn do mezipozice, kde je otočen do příslušné pozice, a v dalším cyklu je přenesen do squib holderu.

Stanice 16 zorientuj a umísti shorting clip pro výrobní variantu 1

Stanice 17 zkontroluj iniciační odpor

Konektor korespondující tvarem s shorting clipem je přiložen k shorting clipu a je provedeno měření odporu. K měření je použit multimetr Keithley.

Stanice 18 zkontroluj zkratovací a isolační odpor

Speciální hlava se 4 kontakty je přiložena ke kusu. Nejprve jsou k multimetru zapojeny 2 kontakty, přes které je změřen zkratovací odpor. Hodnota je uložena, přepínač přepojí k multimetru druhé 2 kontakty a je změřen isolační odpor. Je tak ušetřena jedna měřící jednotka, která je nahrazena funkčně jednodušším (a levnějším) přepínačem (switchem). K měření je použit multimetr Keithley.

Stanice 19 vyhoď špatné kusy do příslušných trubek

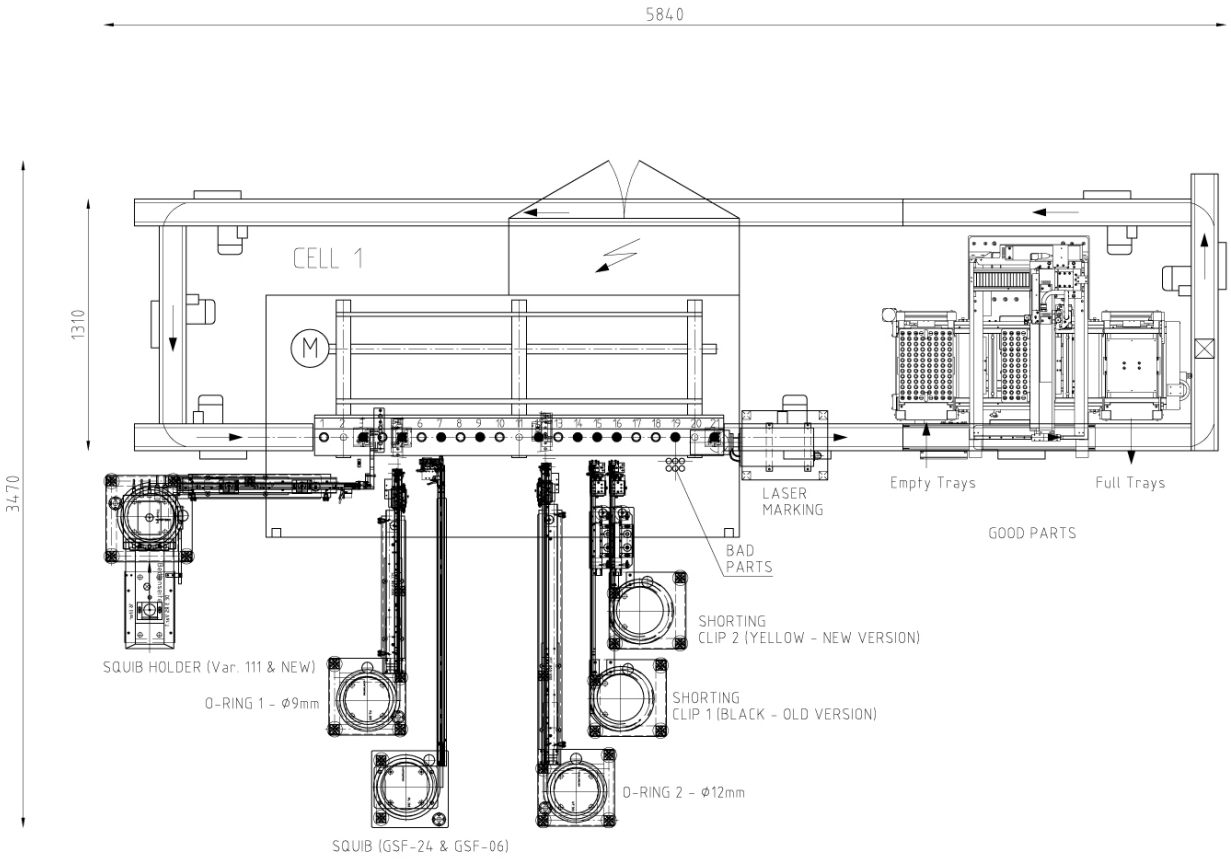
Kusy, které byly při skládání sestavy z nějakého důvodu označeny jako špatné, jsou zde uchopeny pneumatickou uchopovací hlavou a vyhozeny do podle typu vady do příslušné trubky, která vede do sbíracího boxu pro vadné kusy.

Stanice 20 bez operace

Stanice 21 popis laserem a komunikace s paletizérem

Na horní hranu squib holderu je laserem vypáleno číslo série. Laser je řízen vlastním PC, určeným pouze k tomuto účelu. Do tohoto PC je také před začátkem každé série zadáváno její čislo. Program linky dává pouze instrukce, kdy zahájit popis.

Dále je zde v programu uskutečněna komunikace mezi linkou a paletizérem, který je také řízen vlastním softwarem.



Obrázek 4.3: Linka Flexcell - pohled shora

4.3 Software linky

Linka je řízena vlastním softwarem vytvořeným ve firmě Mikron v programovacím jazyku Apigraf V9.0.

4.3.1 Apigraf

Apigraf je programovací jazyk určený k vytváření PLC programů, založený na Turbo Pascalu. Běží na operačním systému MS-DOS. Program v apigrafu je tvořen dvěma základními částmi [4]:

1. deklarace
2. sekvenciál

Deklarace

V časti deklarací je nastaveno vše, co je dále využíváno v samotném programu. Je zde nastaven titulek, jsou importovány moduly pomocí přikazu USES, definovány vstupy a výstupy a vnitřní proměnné.

Způsob importu modulů je identický s Turbo Pascalem. Používá se příkaz USES + název modulu a samotné moduly jsou psané přímo v jazyku Turbo Pascal, přičemž mají přístup k Apigrafovým proměnným. Soubor modulu se musí nacházet ve složce 'UNITS', která musí být ve stejné složce jako kompilovaný program. Kromě uživatelsky vytvořených modulů jsou v programu implicitně k dispozici některé standardní moduly jako System, Crt, Objects... Jejich kompletní seznam je možné dohledat v nápovědě Apigrafu na stránce Utilisation d'unités Pascal (Use of Pascal units).

Již více pro Apigraf specifickou částí jsou část I/O instrukcí. Apigraf obsahuje základní proměnné příslušící fyzickým logickým a analogovým vstupům a výstupům. Tyto jsou označeny I0, I1, I2, ... pokud jde o logické vstupy, O0, O1, O2, ... jde-li o logické výstupy, a ANI0, ANI1, ANI3, ... resp. ANO0, ANO1, ANO2, ... pokud jde o analogove vstupy, resp. výstupy. V části I/O instrukcí je možno pro každou z těchto základních proměnných definovat alias, ve francouzské resp. anglické dokumentaci označen jako *mnémonique/mnemonic*. To potom v programu umožňuje místo číslovaného označení vstupů (např. "I64") používat rozeznatelnější označení jako např. "Motor4On" (motor 4 je v provozu). Při změně zapojení vstupů je pak daleko jednodušší změnit čísla v deklaraci aliasů než procházet a upravovat celý program.

Další částí, velmi podobnou té předchozí, je definice aliasů pro vnitřní PLC proměnné. Apigraf disponuje několika typy vnitřních proměnných, všechny z nich jsou vždy označeny specifickým znakem, případně znaky. Proměnné booleovské jsou: logické značky L, logické přepínače SA a alarmy A a mezi numerické proměnné patří: integerové registry R, grafy nebo GRAFCET sekvence G, časovače DT a timestamp proměnné. Kromě těchto typů existují ještě další, jako řetězce znaků nebo různé předdefinované proměnné, ale o těch se zde není třeba zmiňovat. Co je ale důležité říct je, že ke každé z těchto interních proměnných lze definovat alias, který vždy začíná písmenem označujícím typ proměnné a následuje podtržítkem a požadovaným názvem. Místo "R34" pak můžeme mít např. "R_PartsProducedOK" (počet bezchybně vyrobených kusů).

Poslední možnou částí deklarací je *board instruction*, čili definice panelu manuálního ovládání. Zde lze definovat rozhraní, které se zobrazí na obrazovce při běhu programu a může být použito pro případné ruční ovládání linky nebo nastavení různých parametru. Tato část ale u složitějších programů není příliš využívaná a je spíše dávána přednost grafickému ovládacímu rozhraní, které lze relativně jednoduše vytvořit přímo v programovacím prostředí apigrafu a nabízí daleko větší možnosti.

Sekvenciál

Po provedení deklarací je třeba popsat, jak má program fungovat po tom, co bude spuštěn. V Apigrafu se tak učiní v části nazvané sekvenciál (*sequentiel/sequential*). Sekvenciál je část programu, která, když je program spuštěn, začne být opakována ve smyčce, která je ukon-

čena, až když program dostane impuls ke skončení. Dá se tedy říct, že sekvenciál běží stále nepřetržitě dokola, a zastaví se, až když se vypne celý program. Počet opakování za sekundu sice není zcela stálý, protože délka cyklu záleží na složitosti operací v něm prováděných a ty se mohou v každém cyklu lišit, ale i přesto závisí hlavně na hardwarové konfiguraci použitého počítače a i u dnes už starších PC (pro které byl Apigraf vytvořen) u středně složitých linek se pohybuje řádově ve stovkách za sekundu, což např. u linek firmy Mikron bohatě stačí pro zvládnutí jednoho cyklu sekvenciálu několikrát v průběhu otočení hřídele o 1°.

Syntaxe sekvenciálu je jednoduchá. Na začátku řádku je vždy zapsána podmínka a za ní (oddělen mezerou), případně na dalších řádcích (odsazen mezerami od začátku řádku je příkaz (resp. příkazy), který se má provést. Pokud je podmínka splněna, je příkaz proveden, pokud ne, je přeskočen a pokračuje se dalším kódem.

Jedinou vyjimkou z tohoto způsobu zápisu je příkaz INCLUDEAPIGRAF, který může být umístěn na začátku řádku. Tímto příkazem je možné spustit na daném místě sekvenciálu kód z jiného souboru.

Je možný i zápis komentářů - ty začínají vždy vykřičníkem a můžou být umístěny kdekoliv na řádku.

Na následující stránce je pro lepší představu uvedena ukázka části kódu sekvenciálu z linky Flexcell Century 21 #078314, část z příkazů pro stanici 16.

```
!ȚßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßÝ
!!−−−−−−−− Station 16 [Charge Shorting Clip Yellow Variant A]
!ȚÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÝ

UpJ[1601]     do Ana_Station  (16,1,L_ON16_1,L_OPT1_16,L_OPT2_16)

!ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄż
!ł Selection by Variant                                                                     ł
!ŔÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄŮ
UpJ[1601]
     do PLC_ReadOptStation(16,PLC_AskActVariante,L_ON16,L_TEMP_OPT_1,L_TEMP_OPT_2,R_DUMMY,R_DUMMY)

True
    L_ON16 = L_ON16 & PLC_ON(16)
    L_ON16_1 = L_ON16_1 & L_ON16

!      ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄż
!      łBowlfeeder Alimentation (K450)                                                       ł
!      ŔÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄŮ
True
  K450 = L_ON16 & O_BOWL_FEEDER_1

!      ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄż
!      łFlag use to check the state change empty −> full full −> empty                       ł
!      ŔÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄŮ

UpJ[1601]
    \L_16_LastPartToLoad
    \L_16_FirstPartToLoad

UpJ[1601] & \PLC_V1(16) & PLC_V1(15)
    L_16_LastPartToLoad

UpJ[1601] & PLC_V1(16) & \PLC_V1(15)
    L_16_FirstPartToLoad

((L_ON16 & \PLC_M1(16) & (PLC_ReadInfoPartInt(16,5) = 0) & \PLC_V1(16) & \L_16_LastPartToLoad) #
L_16_FirstPartToLoad)
    L_16_EnableLoadPart
else
    \L_16_EnableLoadPart

!      ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄż
!      łCheck presence of Shorting Clip 1 on selector (U106)                                 ł
!      ŔÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄŮ

INCLUDEAPIGRAF api\STRT_STP 16,L_16_EnableLoadPart,J[1607],U106,'U106'

!      ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄż
!      łOpen Stopper on Linear (Y230)                                                        ł
!      ŔÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄŮ

UpJ[1612] & L_16_EnableLoadPart
    Y230
DownJ[1612]
    \Y230
```

V kódu linek firmy Mikron je často využito proměnných J[XXYY], UpJ[XXYY] a DownJ[XXYY]. Tyto proměnné se váží k takzvaným *softwarovým vačkám* (software cams) linky.

Softwarové vačky jsou jednoduše virtuální vačky, které mají k sobě přiřazené booleanovské proměnné a které mají nadefinováno, pro jaké rozmezí úhlů pootočení hlavní hřídele linky mají mít hodnotu true (zapnuto), přičemž pro zbytek úhlů mají false (vypnuto). Tyto rozmezí úhlů je možné nastavit v uživatelském prostředí programu, přičemž jich přísluší vždy 20 k jedné stanici.

Proměnné příslušící k těmto vačkám jsou, jak již bylo řečeno, J[XXYY], UpJ[XXYY] a DownJ[XXYY]. První dvě číslice indexu (XX) vždy označují číslo příslušné stanice a druhé dvě číslice (YY) označují číslo vačky. Proměnná J odpovídá přímo stavu vačky - má hodnotu true, když je vačka sepnuta, a false, když není. Hodnoty UpJ a DownJ pak odpovídají stavům "vačka právě přešla do stavu zapnuto", resp. "vačka právě přešla do stavu vypnuto", a drží svou hodnotu pouze v průběhu jednoho cyklu sekvenciálu.

Ovládací rozhraní

Jak již bylo řečeno, k apigrafovému programu je také možno (a vhodno) vytvořit uživatelské ovládací rozhraní. To lze využít např. pro zobrazení důležitých aktuálních informací o lince, informací o senzorech, o průběhu měření, různých statistických informací či grafů, ale i pro interakci uživatele s programem, tzn. pro zadání výchozích dat, jako jsou různé limity pro měření, nastavení a vyběr varianty výrobku, nastavení požadovaného počtu vyrobených kusů, či pro umožnění manuálního provozu.

Ovládací rozhraní mít buď grafickou, nebo textovou podobu (klasické DOS rozlišení 80x25 znaků). Textové rozhraní je dnes, zdá se, spíše přežitek z dob, kdy grafika na počítačích nebyla ještě velmi rozšířená a k jeho použití v případě možnosti použít grafickou verzi většinou nejsou valné důvody. Vytvoření grafické verze je navíc relativně jednoduché, Apigraf dává k dispozici vlastní prostředí, v kterém je možno přímo na obrazovku pomocí myši rozmístit jednotlivé prvky jako tlačítka, texty a ilustrace, proměnná textová pole, progress-bary, prostředí pro grafy atd... Podobně jednoduché je i nastavení jejich parametrů, kdy se dvojitým klinutím na prvek vyvolá okno vlastností, kde je možné nastavit všechny potřebné parametry objektu, včetně maker, která se mají spustit při určité interakci uživatele s prvkem (stlačení/uvolnění tlačítka...). Makra pak mohou odkazovat na složitější funkce definované v samostatných souborech. Lze tak potom velmi efektivně zasahovat do běhu programu.

4.3.2 LANtastic Server

Pro komunikaci s jinými PC přes ethernet je v PC linky nainstalován LANtastic Server V8.00. Ten je spouštěn při startu systému přes soubor AUTOEXEC.BAT a běží stále na pozadí. Umožňuje tak přes síťový kabel a LANtastic Client na druhém počítači stálý přístup na oba

disky počítače linky s právy pro čtení i zápis, kdy na klientském počítači je možno disk PC linky připojit jako nový virtuální disk. Toto je výhodné pro úpravy PLC programu, kdy je možno připravit si zdrojové soubory na jiném počítači a poté je pouze na linku nahrát a program zkompilovat, případně je možné při odlaďování editovat přímo soubory na PC linky.

Díky tomu, že LANtastic běží na počítači pořád, lze jej využít i pro přenos dat mezi linkou a jiným PC při výrobě. Při tom je však třeba dávat pozor, protože LANtastic při přenosu dat spotřebovává poměrně velkou část systémových zdrojů. Při čtení většího množství dat se tak může stát, že nebude dostatek zdrojů pro dostatečně rychlý běh programu linky! Proto je třeba za provozu linky přenášet pouze malá množství dat a to, pokud možno, ne příliš často.

Kapitola 5

Návrh řešení

Jak již bylo napsáno výše, úkolem bylo zajistit zaznamenávání a zpracování měřených hodnot na složené součástce load limiteru. Záměrem bylo vše udělat tak, aby program řídící linky měřené hodnoty pouze jednoduše zaznamenával a všechny ostatní operace (trvalé uložení hodnot do databáze, jejich zobrazení, statistické výpočty atd.) byly prováděny zvlášť na druhém počítači, zejména kvůli co nejmenšímu přídavnému zatížení PC linky a kvůli co nejmenším zásahům do řídícího programu. Dále bylo také třeba oddělovat od sebe data z různých sérií.

5.1 Pascal unit

Po důkladném nastudování apigrafového kódu bylo zjištěno, že měřené hodnoty se v něm vyskytují ve vnitřních proměnných a jsou využívány pro vyřazení kusu v případě, že hodnota je mimo meze. Proto zbývalo jen vytvořit a zakomponovat do kódu vhodný pascalový modul, který bude obstarávat zápis dat.

Modul byl pojmenován STATISTI a v následujících částech textu je popsán.

5.1.1 Struktura zapisovaných dat

Všechna měřená data jsou zaznamená do složky /STATDATA, která se nachází v hlavní složce programu.

V této složce se nachází několik souborů:

- MAP.DAT - soubor o velikosti 0 b, vždy přítomen, slouží programu zpracovávájícím data k ověření, zda funguje síťové spojení (program soubor vidí ⇒ spojení OK, nevidí ⇒ NOK)
- 00000000.DAT, 00000001.DAT, 00000002.DAT, ... - datové soubory obsahující měřená data pro aktuální/poslední vyrobenou sérii. V prvním souboru (00000000.DAT) je první

řádek tvořen hlavičkou, obsahujícím informace o dané sérii, jako jsou limity pro měřené hodnoty, v budoucnu případně název série a pod. Všechny další řádky jsou již tvořeny naměřenými hodnotami oddělenými mezerami, ve formátu:

```
[ID kusu] [iniciační odpor] [zkratovací odpor] [isolační odpor] [
    rok] [měsíc] [den] [hodina] [minuta] [sekunda]
```

, kdy každý řádek odpovídá jednomu změřenému kusu.
V každém jednom souboru jsou hodnoty pro 10 změřených kusů. Program na zpracování dat si načítá data ze všech kromě posledního souboru (který ještě nemusí být dokončen), a tím je zajištěno, že při případném pádu systému (a obnově ze záložního obrazu disku) bude ztraceno maximálně 10 posledních hodnot.
Kromě toho jsou v prvním souboru (00000000.DAT) na prvním řádku vždy zapsány mezní hodnoty pro měření, ve formátu:

```
[iniciační odpor min] [iniciační odpor max] [zkratovací odpor min]
    [zkratovací odpor max] [isolační odpor min]
```

- LINEMPTY.DAT - pokud je výroba série dokončena, vytvoří se ve složce ještě tento soubor, který značí, že linka je už prázdná a do začátku další série už nebudou měřeny další hodnoty. Je to signál pro program zpracovávající data, že si může vzít data i z posledního souboru v číselné řadě.

Při začátku nové série se vždy všechny soubory, kromě MAP.DAT, vymažou. V sérii je běžně vyráběno ca. 15000 kusů, takže se nemůže stát, že by číslování "přeteklo".

Zmizení souboru LINEMPTY.DAT je i jednoznačný signál o začátku nové série pro zpracovávající program.

5.1.2 Časové rozvržení zápisu

Protože jsou odpory měřeny na různých stanicích (konkrétně na st. 17 a 18), tudíž nejsou všechny k dispozici v jednom okamžiku, ale zatímco je měřena první hodnota jednoho kusu, je současně doměřována poslední hodnota pro jiný, není možné jejich hodnoty psát do souboru bezprostředně po měření. Místo toho je třeba je ukládat do "mezipaměti" a až jsou pro daný kus změřeny všechny, teprve je zapsat.

Pro dočasné uložení hodnot bezprostředně po změření je použita funkce PLC_WriteInfoPartInt, která je součástí programu linky. Použití funkce je následující:

```
PLC_WriteInfoPartInt(číslo_stanice, index_hodnoty, hodnota)
```

pričemž číslo_stanice je číslo stanice, na které se součástka právě nachází a index hodnoty je index, rozlišující různé ukládáné hodnoty.

To je výhodné, protože program sám posouvá měřené hodnoty mezi jednotlivými stanicemi, takže pro vyvolání určité hodnoty pro kus, který se právě nachází na určité stanici pak stačí:

```
PLC_ReadInfoPartInt(číslo_stanice, index_hodnoty)
```

Kód pro uložení jedné právě změřené hodnoty odporu pak v programu vypadá následovně:

```
UpJ[1709] & L_ON17_1
    R_17_Measure_Keithley = 999999
    R_17_Measure_Keithley = Trunc(Kei2410cGetMeasurePAS(1,'3')
       *1000)
    do PLC_WriteInfoPartInt(17,2,R_17_Measure_Keithley)
```

tedy pokud otočení hlavní hřídele přešlo do pozice, kdy by mělo být doměřeno (UpJ[1709]), a stanice 17 má být aktivní (L_ON17_1), získej změřenou hodnotu a zapiš ji k datům dané součásti.

Všechny 3 hodnoty jsou pak posbírány do "bufferu hodnot" na stanici 20, pomocí funkce modulu Statisti Stat_CollectData:

```
UpJ[2002] & \PLC_V1(20)
  do Stat_CollectData(PLC_ReadInfoPartInt(20,2),
     PLC_ReadInfoPartInt(20,3),PLC_ReadInfoPartInt(20,4)
```

Omezením komplikujícím funkci modulu je fakt, že v Apigrafu nelze využívat funkce pro práci se soubory přímo v kódu sekvenciálu. Je to z důvodu časové náročnosti těchto procesů. Sekvenciál je třeba zopakovat mnohonásobně za sekundu, proto si nemůžeme dovolit jakákoliv zbytečná zdržení. Naštěstí disponuje Apigraf možností, jak se sekvenciálu vyhnout. Tou je speciální seznam funkcí ListeAttenteEvenement, jehož funkce jsou spouštěny vždy, když je dostatek volných systémových prostředků. Stačí tedy vytvořit funkci, která bude obstarávat zápis na disk a tu pak vložit do tohoto seznamu pomocí příkazu k tomu určenému:

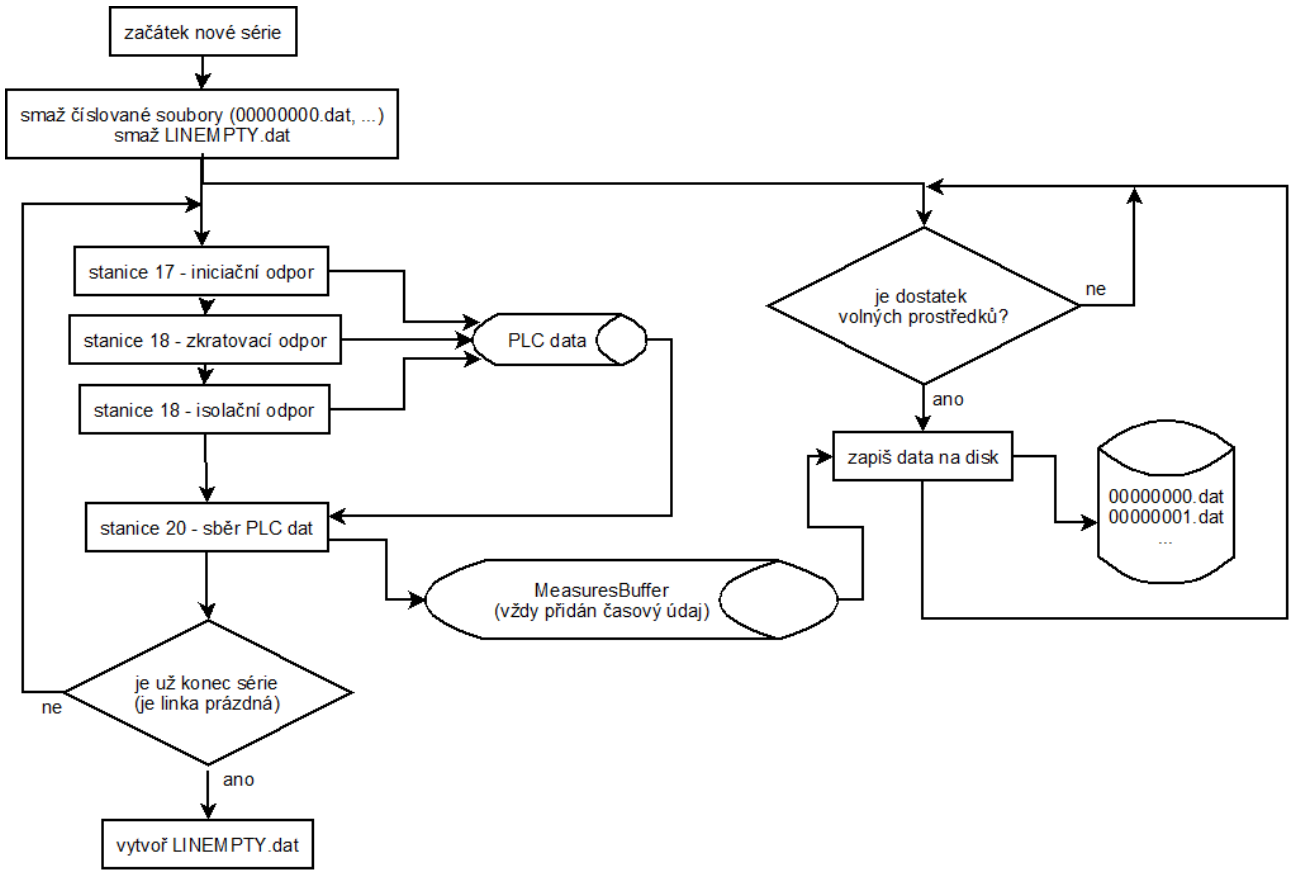
```
ListeAttenteEvenement.Insert(@Stat_WriteData)
```

Dále jsou v kódu určeny akce, které se mají vykonat při začátku a při konci série:

```
DFM(PLC_ReadFillingCycle)
  do Stat_NewBatch
  do Stat_SetLimits(R_17_VAL_MIN,R_17_VAL_MAX,R_18_SC_MIN,
                    R_18_SC_MAX, R_18_INSU_MIN)
UpJ[2102] & PLC_PCell^.DataMachine.WithEmptying &
(NbFixtEmptyFull-PLC_PCell^.DataMachine.CountEmptying-1=NbStationCell)
  do Stat_LineEmpty
```

, tedy pokud začne plnící cyklus zavolej funkci pro smazání číslovaných souborů a LINEMPTY.DAT a ulož limity měření pro danou sérii. Pokud běží vyprazdňovací cyklus a poslední paletka s kusem už opustila linku, zavolej funkci pro vytvoření souboru LINEMPTY.DAT.

Princip celého procesu sběru a zápisu dat je tedy mírně složitější, než by si člověk při prvním zamyšlení představil a je znázorněn diagramem na Obrázku 5.1.



Obrázek 5.1: Sběr měřených dat a jejich zápis na disk

5.1.3 Zdrojový kód modulu

Celý zdrojový kód modulu je dostupný v přílohách. Zde je popsán princip jeho funkce.

Modul používá unity DOS a FileUtils - pro práci se soubory, Objects - pro umožnění objektového programování a DecGloba a ProcGlob - pro možnost použití seznamu ListAttenteEvenement.

Je zde nadefinován vlastní datový typ (record) nazvaný PartMeasures. Ten obsahuje proměnné typu integer pro všechny informace, které je třeba o měřených hodnotách na součástce uchovat - iniciační, zkratovací a isolační odpor a čas uložení hodnot.

Dále jsou vytvořeny tyto proměnné:

- NoMeasInBuffer - počet měření uložených v MeasuresBuffer
- NoMeasWritten - počet měření zapsaných na disk od začátku série
- MeasuresBuffer - pole záznamů PartMeasures pro dočasné uložení hodnot před zápisem na disk
- Measure_17_Min, Measure_17_Max, ... - meze pro měřené hodnoty
- Flag_RunNewBatch, Flag_RunLineEmpty - návěstí pro spouštění procedur Stat_RunNewBatch a Stat_RunLineEmpty

Procedury a funkce modulu jsou popsány v dalším textu.

Procedura Stat_SetLimits

```
procedure Stat_SetLimits(meas_17_min,meas_17_max,meas_18_1_min,
                         meas_18_1_max,meas_18_2_min:longint);
```

Tato procedura nastaví do proměných Measure_17_Min, Measure_17_Max... příslušné hodnoty mezí pro měření pro aktuální sérii.

Procedura Stat_CollectData

```
procedure Stat_CollectData(meas_17,meas_18_1,meas_18_2:longint);
```

Tato procedura přidá do MeasuresBuffer zadané naměřené hodnoty a přidá k nim současný časový údaj.

Je volána pro každou paletku na stanici 20 a naměřené hodnoty jsou do ní načteny pomocí příkazu PLC_ReadInfoPartInt.

Procedura Stat_WriteData

Tato procedura zapíše všechny hodnoty z MeasuresBuffer do textového souboru. Pokud ještě nebyly zapsány žádné hodnoty, zapíše na začátek datového souboru hlavičku, obsahující meze měřených hodnot.

Při zápisu hodnot pro každý kus se zvýší hodnota NoMeasWritten o 1. Při každé desáté hodnotě se zvýší číslo souboru o 1, aby byly v každém souboru hodnoty pro 10 kusů.

Pro zápis do souboru jsou použity standardní pascalové funkce assign, append, reset, write, writeln a close [5].

Procedura je spouštěna vždy, když je dostatek volných systémových prostředků, což je zajištěno příkazem

```
ListeAttenteEvenement.Insert(@Stat_WriteData);
```

na konci modulu.

Procedura Stat_RunNewBatch

Tato procedura smaže všechny datové soubory (0000000.DAT, 00000001.DAT, ...), smaže soubor LINEMPTY.DAT a nastaví NoMeasWritten na 0.

Protože pracuje se soubory, nemůže být volána přímo ze sekvenciálu a musí být umístěna v seznamu ListeAttenteEvenement. Je tedy spouštěna vždy při dostatku systémových prostředků a na začátku jejího kódu je umístěna podmínka, která zajistí spuštění zbytku kódu jen při kladné hodnotě návěstí Flag_RunNewBatch.

Pokud tedy cheme aby tato procedura proběhla, zavoláme jinou proceduru, která nastaví návěstí.

Procedura Stat_NewBatch

Tato procedura nastaví návěstí pro spuštění procedury Stat_RunNewBatch.

Je volána na začátku sekvenciálu při zjištění začátku nové série.

Procedura Stat_RunLineEmpty

Tato procedura vytvoří soubor LINEMPTY.DAT.

Existence souboru LINEMPTY.DAT sděluje programu na vyhodnocení dat, že může vzít data i z posledního číslovaného souboru.

Její spouštění je zařízeno stejným způsobem jako u procedury Stat_RunNewBatch pomocí návěstí Flag_RunLineEmpty.

Procedura Stat_LineEmpty

Tato procedura nastaví Flag_RunLineEmpty na kladnou hodnotu.

Je volána po té, co je linka vyprázdněna.

Iniciace

Při použití modulu v programu jsou na začátku nastaveny hodnoty NoMeasInBuffer a NoMeasWritten na 0, Flag_RunNewBatch a Flag_RunLineEmpty na false a pomocí kódu

```
ListeAttenteEvenement.Insert(@Stat_WriteData);
ListeAttenteEvenement.Insert(@Stat_RunNewBatch);
ListeAttenteEvenement.Insert(@Stat_RunLineEmpty);
```

jsou přidány příslušné funkce do seznamu funkcí spouštěných při dostatku prostředků.

5.2 Aplikace pro zpracování naměřených hodnot

Požadavky na aplikaci na zpracování dat byly následující: program by měl automaticky číst, parsovat a ukládat data zapsaná linkou do jiného, trvalého úložiště. V programu by měly být pro každou sérii a každou měřenou veličinu zobrazeny grafy průběhu měřených hodnot a vypočteny tyto základní údaje: počet OK kusů, počet NOK kusů, podíl NOK kusů v sérii, průměrná hodnota z OK kusů, maximum z OK kusů, minumum z OK kusů. Dále by mělo být možné zobrazit zpětně libovolnou sérii a měly by jít zobrazit data pro jednotlivé kusy produktů.

Adepty na programovací jazyk pro vytvoření programu byli Python a Java. Byl zvolen Python pro svou jednoduchou syntaxi, dostatek využitelných modulů, pro danou aplikaci postačující rychlost a multiplatformnost.

5.2.1 Stručná charakteristika jazyka Python

Python je multiplatformní interpretovaný vysokoúrovňový jazyk. Jeho design klade důraz na jednoduchost a čitelnost kódu. Je trochu podobný Perlu, částečně i Javě, a přesto, že není zcela objektově orientovaný, má hlubokou podporu pro objektové programování. V základním balíku obsahuje širokou škálu rozšiřujících modulů, tyto jdou rozšířit ještě dalšími volně stažitelnými balíčky, případně je možné napojit i moduly napsané v C, C++, Javě nebo .NET.

Charakteristickou specialitou Pythonu je použití odsazení pro oddělení bloků kódu (místo např. složených závorek v C nebo php...), viz následující příklad:

```
if babka_jablka.count()==4:
        print 'Babka měla 4 jablka.'
        babka_jablka.dejDedkovi(1)
```

To doslova nutí k psaní kodu, který je jednotně formátovaný a tedy dobře čitelný. Velikost odsazení musí být samozřejmě v celém programu jednotná.

Kromě toho disponuje Python silnými nástroji pro hlášení chyb. Pokud při běhu programu nastane chyba, je vždy (pokud nejsou chyby ručně umlčeny) do terminálu vypsán tzv. traceback, v kterém je možno dohledat nejen řádek příkazu, při kterém chyba nastala, a typ chyby, ale i výpis všech objektů, přes které došlo k volání daného příkazu. To je velmi praktické, protože to výrazně zrychluje a zjednodušuje práci při odlaďování.

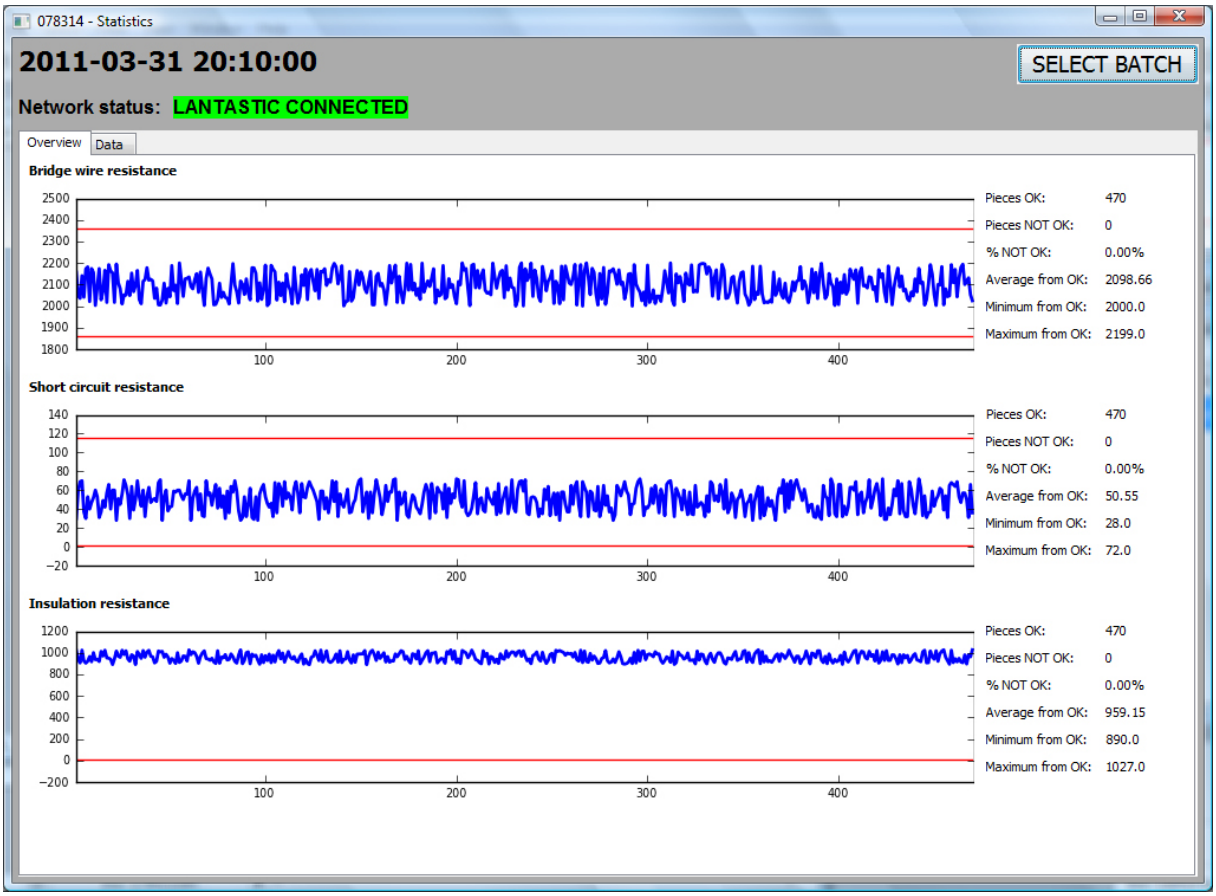
V současné době existují vedle sebe dvě stabilní verze Pythonu - 2.7.1 a 3.2. Mírně se liší syntaxí a svými schopnostmi. Vývoj verze 2 je již ukončen a nebude pokračovat a všechno současné vývojové úsilí je soustředěno na verzi 3. Přestože možnosti verze 3 jsou pravděpodobně o něco širší než verze 2 a Python 3 má před sebou perspektivní budoucnost, pro vytvoření aplikace v této BP byla zvolena verze 2.7.1 z důvodu maximální stability, rozsáhlejší

a dostupnější dokumentace a dostačujícího výkonu.

5.2.2 Grafické uživatelské rozhraní (GUI)

Návrh grafického rozhraní vycházel z velké části z požadavků firmy. Na hlavní obrazovce programu by tedy měly být pro všechny tři měřené hodnoty zobrazeny grafy s vyznačenými mezemi, vedle nichž by se měly nacházet vypočtené statistické hodnoty. Měla by zde být možnost zobrazit kompletní data pro všechny měřené kusy a také možnost změnit zobrazenou sérii. Také by mělo být vždy zobrazeno, zda je v pořádku síťové připojení.

Dle těchto požadavků bylo navrženo GUI, které je zobrazené na obrázku 5.2.

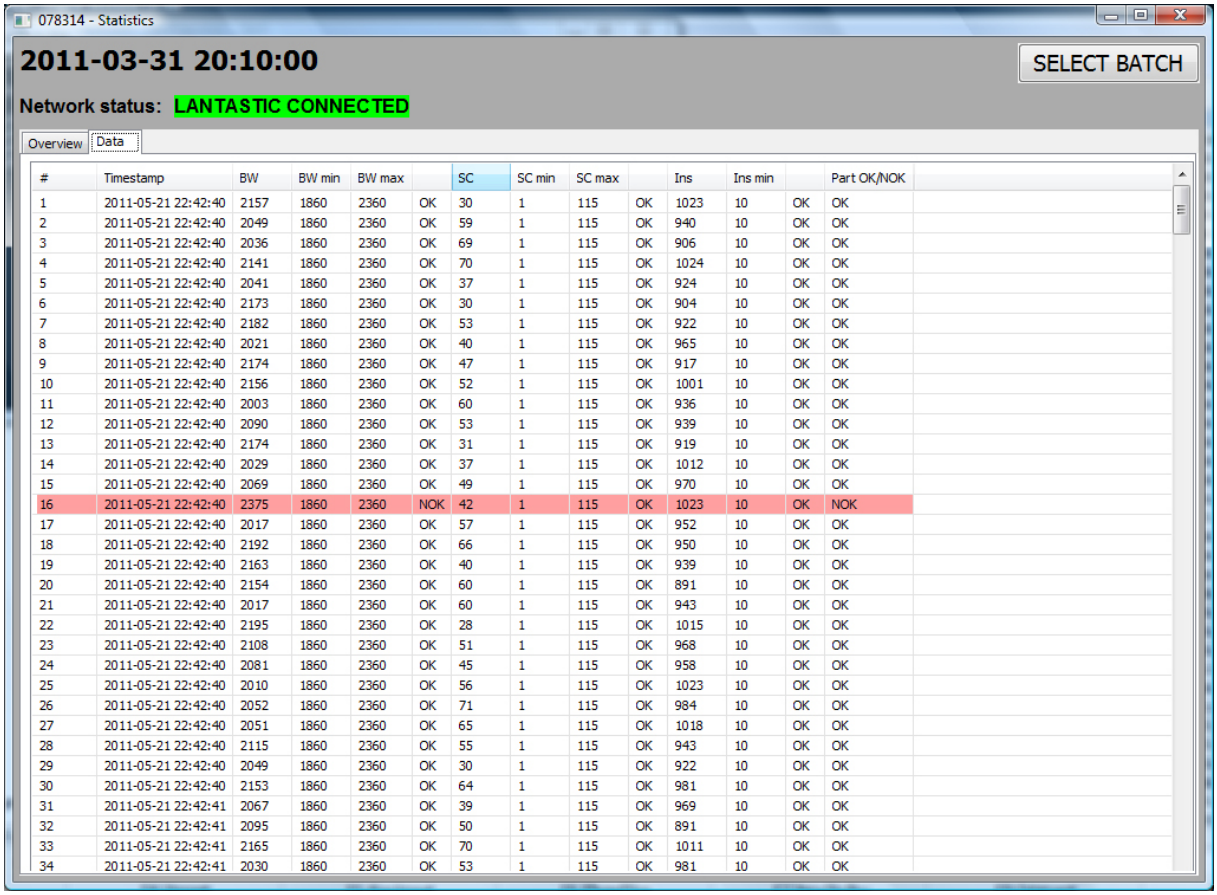


Obrázek 5.2: Hlavní obrazovka programu na vyhodnocování

Jak je vidět, základní obrazovka je velmi stručná a přehledná a při prvním pohledu dává dobrou představu o průběhu hodnot v dané sérii. Technik kvality tak může velmi rychle usoudit, zde je všechno v pořádku, nebo zda se například jeden z odporů pohybuje příliš blízko některé z mezí a zmetkovitost je vyvolána tím, že občas některé hodnoty přeskočí až kousek za mez. V takovém případě může pak učinit opatření, která budou mít za důsledek posunutí všech hodnot více ke středu dovoleného intervalu, čímž i zmenší zmetkovitost.

Pro případ potřeby dohledání hodnot konkrétních vyrobených kusů je zde karta “Data”, v které je zobrazen výpis všech naměřených hodnot v dané sérii. Zde je pro každý kus na řádku zobrazeno datum a čas záznamu měření, měřené hodnoty s příslušnými mezemi, zda je daná hodnota v mezích a - v posledním sloupci - zda jsou všechny hodnoty v mezích (kus je OK). Pokud kus není OK, je řádek zvýrazněn červeně pro jednodušší vyhledávání.

V současné době se vyrobené kusy laserově popisují pouze číslem série. Pokud bude v budoucnu zprovozněno popisovaní kusů i pořadovým číslem, přibude do tabulky hodnot ještě toto číslo. Každý vyrobený kus bude pak zpětně dohledatelný a bude možno určit, s jakými hodnotami odporů byl vyroben.

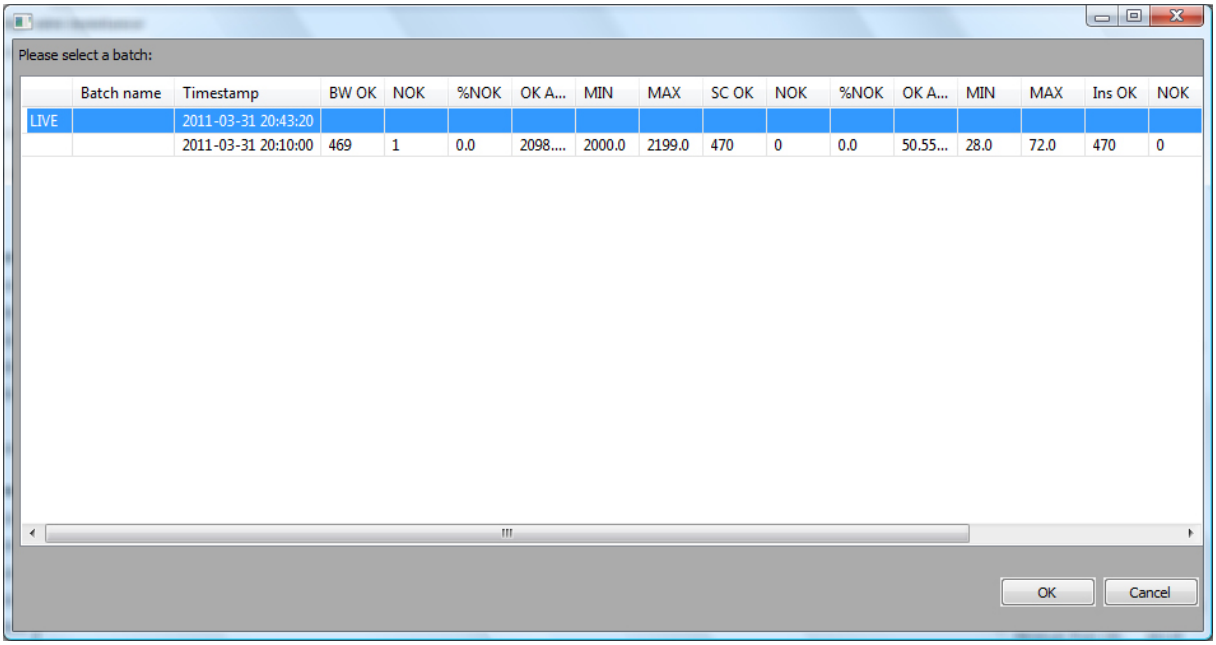


#	Timestamp	BW	BW min	BW max		SC	SC min	SC max		Ins	Ins min		Part OK/NOK
1	2011-05-21 22:42:40	2157	1860	2360	OK	30	1	115	OK	1023	10	OK	OK
2	2011-05-21 22:42:40	2049	1860	2360	OK	59	1	115	OK	940	10	OK	OK
3	2011-05-21 22:42:40	2036	1860	2360	OK	69	1	115	OK	906	10	OK	OK
4	2011-05-21 22:42:40	2141	1860	2360	OK	70	1	115	OK	1024	10	OK	OK
5	2011-05-21 22:42:40	2041	1860	2360	OK	37	1	115	OK	924	10	OK	OK
6	2011-05-21 22:42:40	2173	1860	2360	OK	30	1	115	OK	904	10	OK	OK
7	2011-05-21 22:42:40	2182	1860	2360	OK	53	1	115	OK	922	10	OK	OK
8	2011-05-21 22:42:40	2021	1860	2360	OK	40	1	115	OK	965	10	OK	OK
9	2011-05-21 22:42:40	2174	1860	2360	OK	47	1	115	OK	917	10	OK	OK
10	2011-05-21 22:42:40	2156	1860	2360	OK	52	1	115	OK	1001	10	OK	OK
11	2011-05-21 22:42:40	2003	1860	2360	OK	60	1	115	OK	936	10	OK	OK
12	2011-05-21 22:42:40	2090	1860	2360	OK	53	1	115	OK	939	10	OK	OK
13	2011-05-21 22:42:40	2174	1860	2360	OK	31	1	115	OK	919	10	OK	OK
14	2011-05-21 22:42:40	2029	1860	2360	OK	37	1	115	OK	1012	10	OK	OK
15	2011-05-21 22:42:40	2069	1860	2360	OK	49	1	115	OK	970	10	OK	OK
16	2011-05-21 22:42:40	2375	1860	2360	NOK	42	1	115	OK	1023	10	OK	NOK
17	2011-05-21 22:42:40	2017	1860	2360	OK	57	1	115	OK	952	10	OK	OK
18	2011-05-21 22:42:40	2192	1860	2360	OK	66	1	115	OK	950	10	OK	OK
19	2011-05-21 22:42:40	2163	1860	2360	OK	40	1	115	OK	939	10	OK	OK
20	2011-05-21 22:42:40	2154	1860	2360	OK	60	1	115	OK	891	10	OK	OK
21	2011-05-21 22:42:40	2017	1860	2360	OK	60	1	115	OK	943	10	OK	OK
22	2011-05-21 22:42:40	2195	1860	2360	OK	28	1	115	OK	1015	10	OK	OK
23	2011-05-21 22:42:40	2108	1860	2360	OK	51	1	115	OK	968	10	OK	OK
24	2011-05-21 22:42:40	2081	1860	2360	OK	45	1	115	OK	958	10	OK	OK
25	2011-05-21 22:42:40	2010	1860	2360	OK	56	1	115	OK	1023	10	OK	OK
26	2011-05-21 22:42:40	2052	1860	2360	OK	71	1	115	OK	984	10	OK	OK
27	2011-05-21 22:42:40	2051	1860	2360	OK	65	1	115	OK	1018	10	OK	OK
28	2011-05-21 22:42:40	2115	1860	2360	OK	55	1	115	OK	943	10	OK	OK
29	2011-05-21 22:42:40	2049	1860	2360	OK	30	1	115	OK	922	10	OK	OK
30	2011-05-21 22:42:40	2153	1860	2360	OK	64	1	115	OK	981	10	OK	OK
31	2011-05-21 22:42:41	2067	1860	2360	OK	39	1	115	OK	969	10	OK	OK
32	2011-05-21 22:42:41	2095	1860	2360	OK	50	1	115	OK	891	10	OK	OK
33	2011-05-21 22:42:41	2165	1860	2360	OK	70	1	115	OK	1011	10	OK	OK
34	2011-05-21 22:42:41	2030	1860	2360	OK	53	1	115	OK	981	10	OK	OK

Obrázek 5.3: Zobrazení podrobných dat

Tlačítko Select Batch slouží pro výběr jednotlivých sérií. Po kliknutí na něj se zobrazí okno s výpisem všech sérií uložených v databázi, včetně statistických hodnot vypočtených u už ukončených sérií. Série, která se právě vyrábí, je označena jako ’LIVE’ na začátku svého řádku. Po vybrání příslušné a potvrzení jsou z databáze načteny všechny měřené hodnoty, v případě nutnosti vypočteny statistiky, jsou vykresleny grafy a je vytvořen seznam hodnot v kartě Data.

Pro vytvoření GUI byl použit balíček *wxPython*, což je Pythonovská odnož C++ knihovny tříd *wxWidgets*, která umožňuje vytvářet grafická rozhraní aplikací pro většinu hlavních dnes

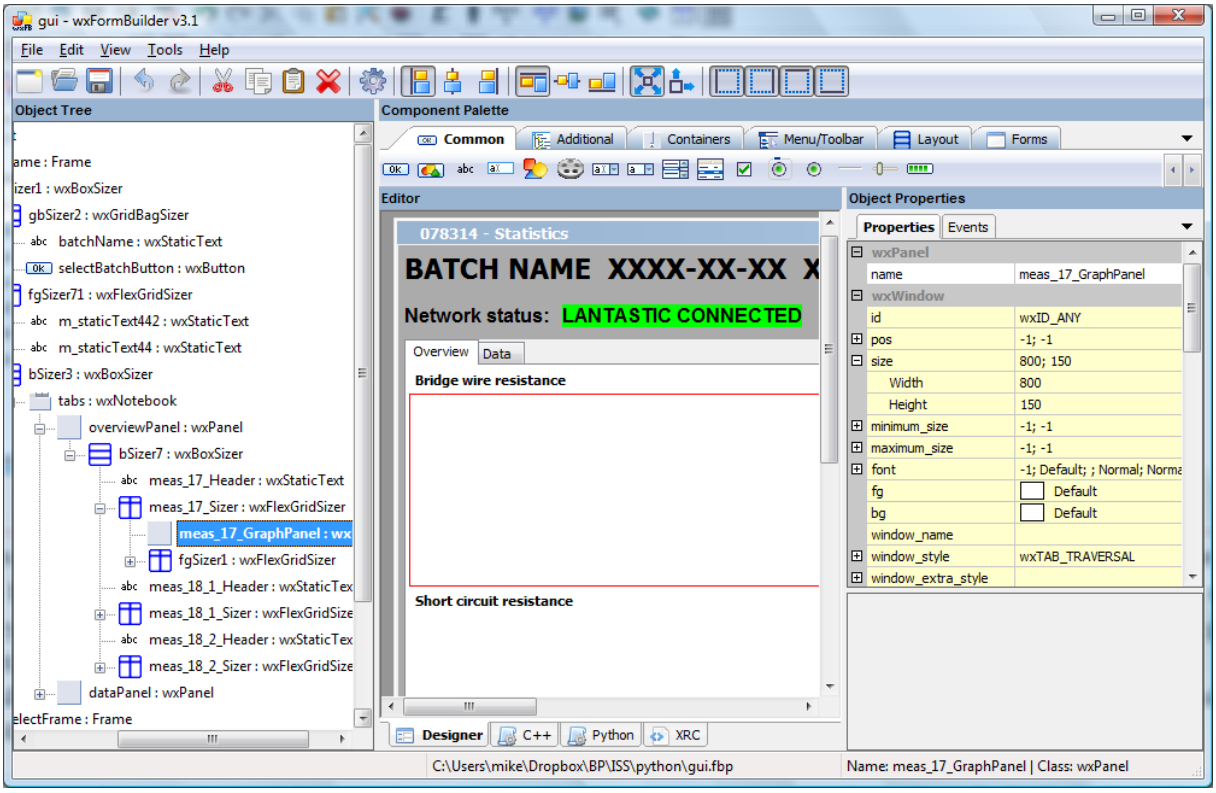


Obrázek 5.4: Okno výběru série

používaných operačních systémů (Windows, Mac OS X, Linux, OS/2 a další). V případě potřeby tedy bude možné po krátkém odladění vytvořenou aplikaci spustit i na PC s OS Linux (čímž se může ušetřit za licenci).

Pro wxPython existuje velice šikovný nástroj wxFormBuilder, což je aplikace dostupná pro Windows, Mac OS X a Linux, která umožňuje rychlé vytvoření GUI pomocí svého grafického rozhraní tak, že lze jednotlivé prvky vytvářeného okna přímo umístit do stromu objektů, přímo nastavit jejich parametry, a výslednou podobu je možné hned vidět. Z takto vytvořeného návrhu je pak možné přímo vygenerovat pythonový soubor obsahující příslušnou třídu s potřebnými objekty, kterou stačí importovat do programu a využít.

Tento nástroj byl při tvorbě aplikace využit pro vytvoření základního rozložení prvků v oknech. Z vygenerované třídy byla pak vytvořena třída odvozená, v které byly dodefinovány funkce pro jednotlivá tlačítka použitá při výběru série a dodefinovány spojení mezi událostmi těchto tlačítek a funkcemi.



Obrázek 5.5: prostředí wxFormBuilderu

5.2.3 Princip funkce programu

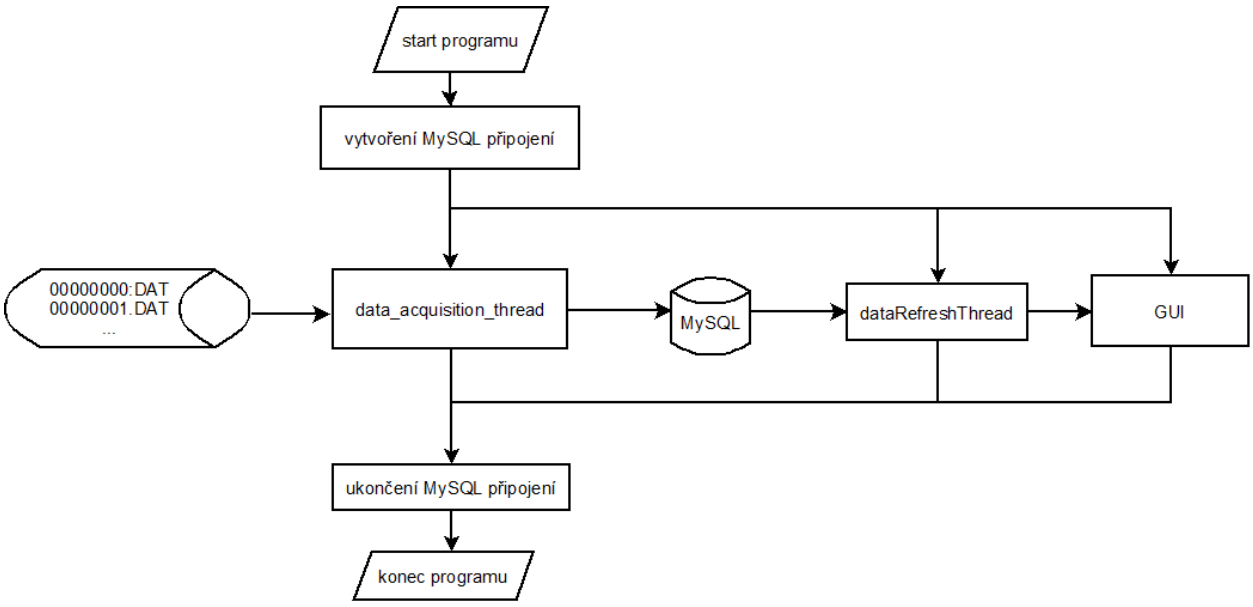
V programu běží po spuštění sumultánně několik procesů. Prvním jsou procesy v rámci data_acquisition_thread, které sledují soubory ve složce STATDATA na počítači linky a stará se o jejich ukládání do databáze. Dalším je dataRefreshThread, který je spuštěn v případě, že jsou v GUI zobrazeny "živá" data, tj. data pro aktuálně vyráběnou sérii, a který zabezpečuje obnovování všech aktuálně zobrazených grafů a hodnot. Třetím procesem je hlavní větev programu, která zprostředkovává vytvoření připojení k databázi, spuštění GUI a možnost jeho ovládání.

5.2.4 MySQL databáze

Pro finální úložiště dat bylo vybíráno mezi těmito alternativami:

- zápis do textových souborů (vlastní formát / CSV / XML)
- využití databáze (MySQL, PostgreSQL, Oracle, Microsoft SQL Server...)

Zápis do souborů se nejevil jako praktický, z důvodu omezené možnosti manipulace s daty a malé přehlednosti. Proto bylo zvoleno ukládání dat do databáze, přičemž vyhrála MySQL,



Obrázek 5.6: Základní struktura programu

z důvodu dostačující funkcionality, velké rozšířenosti (jednodušší hledání řešení případných problémů...) a dostupnosti bezplatné verze.

Strukturu navržené databáze tvoří tyto tabulky:

Tabulka batches

Tato tabulka obsahuje základní informace o jednotlivých vyrobených sériích. Je tvořena následujícími sloupcemi:

- id - ID řádku, primární klíč
- name - název série (připraveno pro další rozšíření programu, kdy bude možné přímo v programu linky zadávat název série. Nyní se zadává pouze do počítače laserového popisovače)
- timestamp - časová známka unixového typu, čas kdy byla série započata
- live - boolean hodnota, označuje zda ještě probíhá výroba dané série (pokud ano, jsou v programu opakovaně obnovávana zobrazení grafů a hodnot)
- meas_17_min, meas_17_max, meas_18_1_min, meas_18_1_max, meas_18_2_min - hodnoty mezí pro jednotlivá měření

Nová série je do tabulky přidána vždy, když je detekována zkrze data v souborech zapsaných linkou. Přitom jsou vždy zrovna z prvního souboru načteny a uloženy (00000000.DAT) mezní hodnoty pro měření.

Tabulka batchesOverview

Tabulka slouží pro uložení vypočtených statistických ukazatelů pro již hotové série, aby nemusely být znovu počítány při každém požadavku o zobrazení dané série a mohly být rychleji zobrazeny.

Tabulka obsahuje následující sloupce:

- id - ID řádku, primární klíč
- batch - ID série, ke které hodnoty patří
- measName - název měřené hodnoty, ke které hodnoty patří (může být 'meas_17', 'meas_18_1', nebo 'meas_18_2')
- countOK - počet OK kusů v rámci dané měřené hodnoty
- counNOK - počet NOK kusů v rámci dané měřené hodnoty
- percentNOK - podíl zmetků z celkového počtu kusů v rámci dané měřené hodnoty
- OKAverage - průměr z OK hodnot
- OKMin - minimum z OK hodnot
- OKMax - maximum z OK hodnot

Data jsou do tabulky zapsána vždy hned po detekování ukončení série, aby příště již nemusela být počítána.

Tabulka measurements

Do této tabulky jsou ukládány změřené hodnoty pro jednotlivé vyrobené kusy. Tabulku tvoří tyto sloupce:

- id - ID řádku, primární klíč
- batch - ID série, ke které daný kus přísluší
- timestamp - časová známka unixového typu, čas, kdy byly měřené hodnoty pro daný kus uloženy do souboru
- meas_17 - hodnota iniciačního odporu
- meas_18_1 - hodnota zkratovacího odporu
- meas_18_2 - hodnota isolačního odporu

Na tabulce jsou kromě primárního klíče vytvořeny indexy na sloupcích batches a timestamp, protože bude často používáno vyhledávání podle těchto sloupců.

Navíc v rámci jedné série může být běžně vyrobeno až 15000 kusů, což odpovídá 15000 řádkům v tabulce measurements. Pokud by na sloupci batch nebyl vytvořen index a bylo podle něj vyhledáváno v rámci řádově sto tisíc a více řádků (musíme započítat i data z ostatních vyrobených sérií), musel by počítač projít všechny tyto řádky jeden po druhém, což už by znamenalo velmi značné zpomalení [8].

5.2.5 Threading

Běh několika simultánních procesů je - jak již bylo naznačeno - zajištěn pomocí tzv. vláken (anglicky *threads*). Tyto python nativně podporuje svým standardním modulem threading. Odvození vlastních (uživatelských) tříd z třídy Thread tohoto modulu umožňuje nadefinování operací, které se mají spustit při iniciaci threadu, při jeho spuštění a také je zde možné dodefinovat návěstí pro zastavení threadu.

Nové vlákno pak může být v programu iniciováno příkazem podobným tomuto:

```
thread = my_thread_class()
```

a spuštěno takto:

```
thread.start()
```

Po jeho spuštění začnou procesy vlákna probíhat ve vlastní (nové) větvi programu, přičemž hlavní větev pokračuje současně dál dalšími příkazy.

V některých situacích je však potřeba zajistit, aby jednotlivá vlákna současně NEprobíhala, nýbrž aby jedno vlákno počkalo na dokončení činnosti jiného až poté pokračovalo. Toto je možno zajistit na několika různých úrovních složitosti, z nichž nejjednodušší je nástroj jménem *threading locks*.

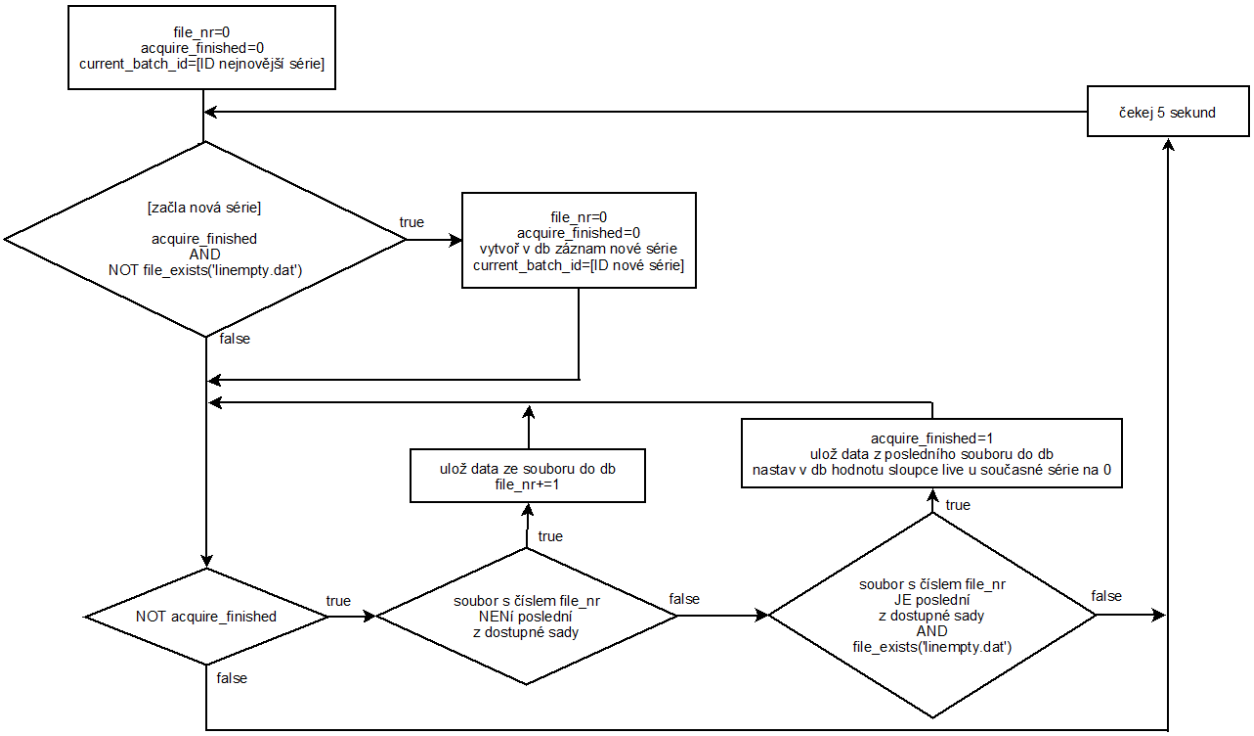
Threading lock se může nacházet ve dvou stavech - uzamčený (locked), nebo odemčený (unlocked). Vytvořen je v odemčeném stavu. Má dvě základní metody - acquire() ("osvojit") a release() (uvolnit). Pokud je lock odemčený, acquire() změní jeho stav na uzamčený a vrátí se. Pokud je uzamčený, průběh funkce acquire() se zablokuje až do té doby, než je stav locku uvolněn zavoláním funkce release() z jiného vlákna.[9]

Toto je v programu využito pro přistupování k databázi, kdy je mezi dvěma thready sdíleno jedno připojení. Je logické, že jedno připojení nemůže být využíváno více procesy současně, proto je třeba ohlídat, aby bylo vždy používáno maximálně jedním. Právě pro tuto možnost jsou threading locks ideálním řešením.

5.2.6 Modul data_acq

Modul data_acq (definován v souboru data_acq.py) se stará o načítání informací zprostředkovaných linkou textovými soubory (00000000.DAT, 00000001.DAT, LINEMPTY.DAT) a jejich ukládání do databáze.

Princip funkce modulu je znázorněn v diagramu na obrázku 5.7.



Obrázek 5.7: Princip funkce třídy data_acquisition_thread

V modulu jsou použity tyto hlavní proměnné:

- file_nr - číslo čteného datového souboru (0 odpovídá 00000000.DAT, 1 odpovídá 00000001.DAT, atd...)
- acquire_finished - proměnná typu boolean, má hodnotu 1, pokud již byla načtena všechny data v rámci právě načítané série
- current_batch_id - ID řádku právě načítané série v databázi v tabulce batches

Použití těchto proměnných je znázorněno v diagramu.

Jak je vidět v diagramu, celý proces běží v cyklech a vždy zjišťuje dostupnost zatím nezapsaných dat pro danou sérii, a všechny, až na poslední soubor - do toho by se mohlo ještě zapisovat, načte a uloží do databáze. Pokud zjistí přítomnost souboru LINEMPTY.DAT (linka je vyprázněná), vezme data i z posledního souboru a nastaví hodnotu acquire_finished (všechny hodnoty pro danou sérii byly už uloženy) na 1.

Pokud acquire_finished==1 a soubor LINEMPTY.DAT zmizí, znamená to, že začala nová série, program tedy vytvoří nový záznam v databázi v tabulce batches, jeho ID uloží do proměnné current_batch_id a vyresetuje hodnoty proměnných file_nr a acquire_finished.

Celá tato funkčnost se skrývá v modulu ve třídě data_acquisition_thread, která je odvozena z pythonové třídy threading.Thread() a spuštěna je tedy v hlavním programu tímto kódem:

```
import data_acq
(...)
data_acq.lock=lock
acq = data_acq.data_acquisition_thread(connection)
acq.start()
```

, přičemž lock je instance objektu threading.Lock() a connection je instance MySQL připojení, které jsou obě již dříve vytvořené.

Aby se zabránilo přistupování ke připojení z více vláken současně, je každé využití připojení v modulu zajištěno threading lockem (stručně popsáno v předchozí kapitole), jehož instance je předána příkazem

```
data_acq.lock=lock
```

. Kód využívající připojení k MySQL pak v modulu vypadá podobně, jako tento:

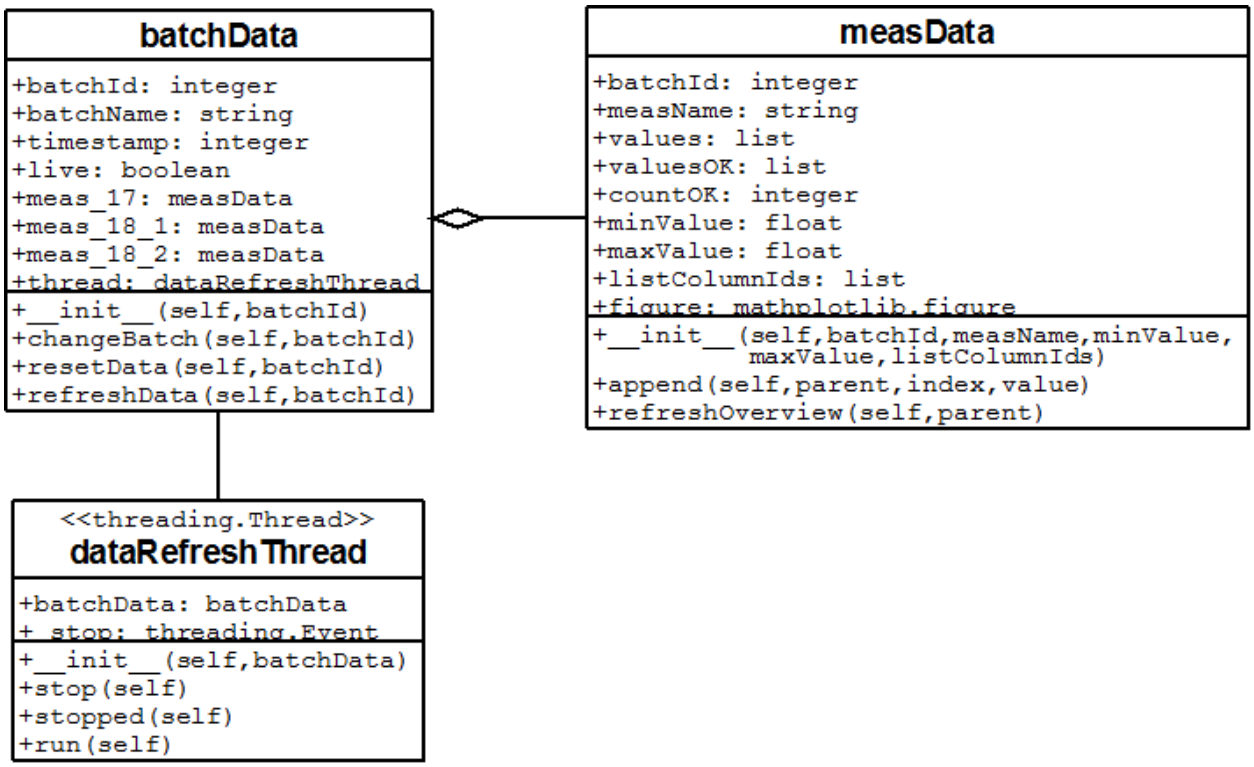
```
lock.acquire()
cursor=self.connection.cursor(MySQLdb.cursors.DictCursor)
cursor.execute('select * from batches order by live desc, timestamp
                desc limit 1')
row=cursor.fetchone()
lock.release()
```

5.2.7 Modul interface

Modul interface se stará o vytvoření instancí všech objektů potřebných pro grafické rozhraní, dále o načtení dat právě zvolené série, vypočtení hodnot všech zobrazovaných ukazatelů a jejich zobrazení společně s vykreslením grafů a - v případě zobrazení právě vyráběné série - o kontrolování dostupnosti nových data a aktualizaci zobrazení.

Pro vytvoření grafického rozhraní je využit standardní balíček wx (respektive jeho verze pro python wxPython [7]). Třídy grafického rozhraní jsou primárně nadefinovány v souboru gui.py, který je byl automaticky vygenerován programem wxFormBuilder [11]. Tento soubor je pak jako modul importován do souboru gui_my.py, kde jsou z jeho třídy vytvořeny třídy odvozené, u kterých jsou dodefinovány potřebné vlastnosti (funkce některých tlačítek,

sloupce seznamu v kartě Data atd...). Tento soubor je pak jako modul importován do modulu interface, kde jsou jeho třídy využívány.



Obrázek 5.8: Třídy modulu interface

Vlastními třídami definovámi v modulu jsou batchData, measData a dataRefreshThread (viz obr. 5.8).

Třída batchData je hlavní třídou modulu. Jsou v ní shromážděna všechna data pro aktuálně zobrazenou sérii. Kromě inicializační funkce obsahuje 3 další funkce. Funkce changeBatch slouží pro změnu zobrazené série. Funkce resetData slouží pro nastavení proměnných do výchozího stavu tak, aby mohla být kompletně načtena všechna data z databáze. Pokud je aktuálně vybraná série ještě vyráběna (live==true), vytvoří funkce instanci objektu dataRefreshThread, který opakovaně volá funkci refreshData. Pokud se jedná o již vyrobenou sérii, pouze se jednou zavolá refreshData.

Funkce refreshData načte z databáze data měření příslušící ke zvolené sérii a uloží jejich hodnoty do objektů measData. Pokud se data obnovují pro již zobrazenou sérii (live==true), nejsou načítána všechna data pokaždé znovu, ale jsou načtena jenom ta, která za uplynulou dobu přibyla. Po uložení všech hodnot je u všech instancí objektů measData zavolána funkce refreshOverview, čímž se obnoví zobrazení grafů a hodnot.

Třída measData obsahuje data, která přísluší k jednomu typu měřené hodnoty (iniciační odpor/zkratovací odpor/isolační odpor). Obsahuje vždy seznam všech načtených měřených hodnot (objekt values), seznam hodnot, které spadají do mezí pro měření (valuesOK), počet

OK hodnot (countOK) a hodnoty mezí pro měření (minValue, maxValue). Také obsahuje instanci objektu mathplotlib.figure [10], která je použita pro vykreslování grafů hodnot.

Její funkce append je použita pro přidání přidání měřené hodnoty.

Funkce refreshOverview umožňuje spočítat statistické ukazatele z právě uložených hodnot a obnovit hodnoty těchto ukazatelů zobrazených v GUI programu a znovu vykreslit grafy.

Pokud je právě zobrazovaná série "živá" a mají se přepočítávat statistické hodnoty jen kvůli několika málo novým hodnotám, nedává smysl přepočítávat např. průměr, minimální a maximální hodnotu z celého pole valuesOK, které může mít při konci série až 15000 hodnot. Proto je například průměr valuesOK počítán pomocí vážených průměrů, kdy výsledná hodnota je průměr z původní hodnoty průměru s váhou počtu kusů v původním poli a z přidávaných hodnot s váhami 1. Stejně tak nemá smysl procházet celé pole pro znovunalezení nejmenší a největší hodnoty, stačí zjistit, zda některá z přidávaných hodnot je menší (resp. větší) než minimum (resp. maximum) nalezené v původním poli a tuto pak označit za minumum (resp. maximum). Tím se obnovování zobrazení podstatně urychlí - výpočet nové průměrné hodnoty klesne pod jednu sekundu oproti několika sekundám při opětovném počítání z všech hodnot pole - a ztráta přesnosti je přitom zanedbatelná.

Třetí třídou modulu je dataRefreshThread, což je objekt vytvořený předefinováním standardní třídy threading.Thread. Při jeho iniciaci je vytvořeno vlákno, které po spuštění každých 10 sekund volá funkci batchData.refreshData, čímž se načítají nová data a aktualizuje se zobrazení. Pokud "živá" série skončí a je započata nová, přepne automaticky zobrazení na novou sérii tak, aby na obrazovce vždy (pokud není zobrazena stará série) bežela aktuální data.

5.2.8 Hlavní soubor programu - main.py

Hlavní soubor programu je již relativně jednoduchý, protože pouze jednoduše využívá složitou funkcionalitu skrytou ve výše popsaných modulech.

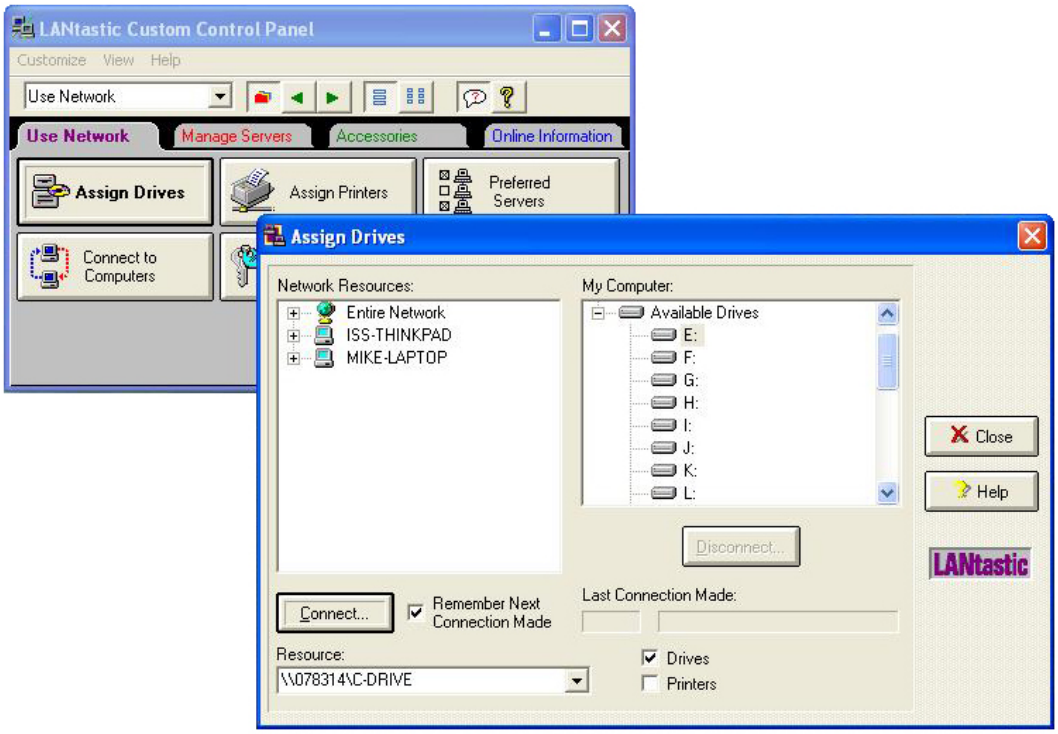
Na jeho začátku jsou importovány potřebné moduly, jimiž jsou MySQLdb pro práci s databází, threading pro práci s vlákny a data_acq a interface, které jsou popsány výše. Dále je zde vytvořeno připojení k MySQL databázi a je vytvořena instance threading locku, která je dále předána modulům data_acq a interface. Stejně tak je předána i instance připojení k databázi. Je iniciována instance třídy batchData v modulu interface, přičemž je vybráno zobrazení dat nejnovější dostupné série. Následně je zobrazeno hlavní okno grafického rozhraní, které je spuštěno funkcí MainLoop.

Poté co grafické rozhraní obdrží požadavek na ukončení programu (je zavřeno hlavní okno $\Rightarrow$ funkce MainLoop skončí), je ukončen thread data_acq (sběr dat ze souborů na lince) a je uzavřeno připojení k databázi. Poté program skončí.

5.3 Propojení přes ethernet přes LANtastic

Jak již bylo naznačeno dříve, propojení mezi počítačem linky a počítačem, na kterém poběží program na vyhodnocení dat, je uskutečněno pomocí ethernet portu. Na lince běží OS Microsoft DOS a na něm LANtastic server, proto je na vyhodnocovacím počítači nainstalován LANtastic Client, který umožňuje souborový systém linky připojit jako virtuální disk.

Disk C linky je tedy na počítači pro vyhodnocování připojen jako virtuální disk E a dále je k němu přistupováno jako k normální jednotce.



Obrázek 5.9: Ovládací prostředí programu LANtastic

Připojení je nastaveno tak, aby se při každém spuštění počátače znovu obnovilo.

Kapitola 6

Ověření v praxi

Navržený systém byl v průběhu měsíce května ve firmě uveden do provozu a prozatím se nevyskytly žádné problémy s jeho provozem.

Nasazení upraveného programu linky a implementace pascalového modulu sice předcházelo dlouhé odlaďování, protože apigrafový program nebylo možné vyzkoušet, ba ani zkompilovat jinde než přímo na PC linky (kvůli nutnosti použít hardwarový klíč), nicméně to aspoň zaručilo, že byly úpravy programu implementovány "kousek po kousku", a byly tedy všechny řádně odzkoušeny ve všech možných situacích, které můžou při provozu nastat.

Program pro vyhodnocování byl sice původně testován jenom pomocí simulačního programu, který zastupoval software linky a zapisoval náhodná data do stejně strukturovaného systému souborů jako modul statisti (kompletně vč. souboru LINEMPTY.DAT), nicméně jednoduše navržený systém zápisu dat umožnil jeho dobré otestování i tímto způsobem a po jeho nasazení do provozu fungoval tak, jak se předpokládalo.

Celkově tedy hodnotím nasazení systému do praxe jako úspěšné.

Kapitola 7

Závěr

Cíle bakalářské práce - sestavit základní problémy výměny dat v automatizované výrobě a popsat vybraný problémový případ a doporučit způsob řešení - byly splněny.

Hlavním přínosem této práce pro firmu Indet Safety Systems, a. s. je, že nyní může sledovat průběh měřených hodnot všech vyráběných kusů v rámci sérií a odhalit tak případné příčiny zmetkovitosti. Pokud by se např. stalo, že se hodnota některého z odporů pohybuje v těsné blízkosti některé z mezí a zmetkovitost je dána přeskočením hodnoty až za hodnotu meze, může firma učinit opatření, aby hodnotu odporů posunula více doprostřed vymezeného intervalu, čímž zvětší úspěšnost výroby.

Dalším přínosem pro firmu je také fakt, že byla zanalyzována a popsána jak funkce programovacího jazyka Apigraf, tak i programů firmy Mikron v něm napsaných, které ve firmě řídí velké množství dalších velmi podobných linek. Firma tím získala jednak lepší náhled na princip jejich funkce, ale také potenciál k určité samostatnosti, co se týče často potřebných drobných úprav programů, které byly prozatím prováděny přibližně jednou za rok osobně švýcarským technikem, což bylo pro firmu sice potřebné, ale také poměrně nákladné.

Možnosti dalších vylepšení

Řešení navrhnuté v této bakalářské práci lze pochopitelně ještě dále zdokonalit. Jedním z dalších kroků, které by bylo dobré udělat, je zprovoznění sériové komunikace (RS-232 [6]) mezi PC linky a počítačem laserového popisovače. Pak by bylo totiž možné upravit program linky tak, aby bylo číslo série, které je v současné době vždy třeba zadat ručně do PC laserového popisovače, zadáváno přímo do programu linky při zadávání požadavku na novou sérii. To by nejen mírně zjednodušilo práci obsluze, ale bylo by výhodné i pro identifikaci zapisovaných dat, kdy v současné době jsou jednotlivé série od sebe odlišitelné pouze časem začátku výroby. Po této úpravě by se však stalo rozlišení zcela jednoznačným, protože každá sada dat by byla jednoznačně identifikována číslem příslušné série, což by také zjednodušilo orientaci při výběru série při zobrazení starších dat.

Pro tuto úpravu by sice bylo třeba doplnit chybějící driver ke komunikaci s laserem, je však možné, že se jej podaří získat z jiné podobné linky, kde je stejný typ již použit.

Dalším možným zlepšením je upravit program tak, aby jednotlivým vyrobeným kusům přiřazoval identifikační čísla a údaj o tomto čísle pak posílal do laserového popisovače. Každý kus by tak mohl být jednoznačně identifikován, přičemž by bylo možné tento údaj přiřazovat i k zapisovaným datům měření. Pak by bylo možné dohledat přesné změřené hodnoty libovolného vyrobeného kusu, což by mohlo poskytnout určitou výhodu při případných reklamacích. Tuto možnost by však bylo třeba udělat vypínací, protože přidání ID čísla na load limiter je změnou oproti jeho původnímu výkresu a tato změna zatím nebyla dojednána se všemi zákazníky.

Všechny tyto předeslané úpravy jsou v plánu na uvedení do provozu v průběhu léta tohoto roku a přispěly by jednak k jakési "úplnosti" řešení a také k dalšímu, ještě hlubšímu pochopení jazyka Apigraf a programů v něm napsaných.

Literatura

[1] Balátě, J.: Automatické řízení. BEN 2003 Praha

[2] Lacko, B. - Beneš, P. - Maixner, L. - Šmejkal, L.: Automatizace a automatizační technika. Computer Press 2000 Praha

[3] Švarc, I.: Automatizace a automatické řízení. CERM 2002 Brno

[4] Aide sur le langage Apigraf. Saint Cyr sur Loire, France: Optimalog, [199?]. 307 s.

[5] MOTYČKA, Arnošt. Algoritmizace. První vydání. Brno: Konvoj, 1999. 75 s. ISBN 80-85615-80-0.

[6] Manual No 3C : Project IND-LL1. Boudry, Suisse: Mikron, 2008. 256 s.

[7] wxPython [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <http://www.wxpython.org/>.

[8] MySQL Documentation [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <http://dev.mysql.com/doc/>.

[9] Python v2.7.1 documentation [online]. 2011 [cit. 2011-05-19]. Threading - higher-level threading interface. Dostupné z WWW: <http://docs.python.org/library/threading.html#thread-objects>.

[10] Mathplotlib: python plotting - Documentation [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <http://matplotlib.sourceforge.net/>.

[11] Project wxFormBuilder [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <http://wxformbuilder.org/>

[12] Automa - časopis pro automatizační techniku [online]. 2007 [cit. 2011-05-19]. Foundation Fieldbus dominuje na poli průmyslových sběrnic. Dostupné z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=33942>.

[13] Navajo - otevřená encyklopedie [online]. 2011 [cit. 2011-05-19]. Automatizace. Dostupné z WWW: <http://automatizace.navajo.cz/>.

[14] Wikipedia, the free encyclopedia [online]. 15. 5. 2011 [cit. 2011-05-19]. Automation. Dostupné z WWW: <http://en.wikipedia.org/wiki/Automation>.

[15] Wikipedia, the free encyclopedia [online]. 10. 5. 2011 [cit. 2011-05-19]. Bus (computing). Dostupné z WWW: <http://en.wikipedia.org/wiki/Bus_%28computing%29>.

[16] Wikipedie, otevřená encyklopedie [online]. 18. 5. 2011 [cit. 2011-05-19]. Sběrnice. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Sb%C4%9Brnice>.