K sJIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
PŘÍRODOVĚDECKÁ FAKULTA

Katedra: Ústav aplikované informatiky **Obor**: Aplikovaná informatika

BAKALÁŘSKÁ PRÁCE

Vývoj desktopové aplikace v jazyce Java pro ovládání 3D posuvného systému

Desktop application development in Java for 3D control sliding system

Autor bakalářské práce:

Jakub Egner

Vedoucí bakalářské práce:

Zbyněk Kolář, Robert Bosch, spol. s r.o.

Garant bakalářské práce

Jaroslav Icha, RNDr.

České Budějovice 2013

Bibliografické údaje

EGNER,J.:2013: Vývoj desktopové aplikace v jazyce Java pro ovládání 3D posuvného systému[Desktop application development in Java for 3D control sliding system] – 39 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Tato práce se zabývá vývojem desktopové aplikace pro ovládání 3D posuvného systému v jazyce Java. Požadavek na aplikaci vznikl ve firmě Robert Bosch v Českých Budějovicích, které nevyhovuje dosavadní aplikace.

Cílem bakalářské práce je vytvoření takové aplikace, která bude využívat pro komunikaci s posuvným systémem průmyslovou sběrnici Profibus-DP. Cílem je větší přehlednost, rychlejší a bezpečnější ovládání všech řídících jednotek typu TLC53F najednou, zahrnující výběr typu pozicování a sledování módu, ve kterém se aktuálně jednotky nacházejí. Dalším požadavkem je zabudování komunikace s optometrem typu optoControl 2500, který sleduje aktuální vzdálenost osy X od překážky.

Klíčová slova

Desktopová aplikace, Java Native Interface, Java, Profibus-DP, DF PROFI II

Abstract

This work is interested in developing Java desktop application for 3D sliding system control. The company Robert Bosch in České Budějovice came with a request to build a new application because of their dissatisfaction with the old one.

The main goal of the work is to create such application where the industrial bus Profibus-DP will be used for communication with sliding system. The requirements are simplicity, faster and safer control over all control units of TLC53 type at once., including the selection of positioning type and monitoring mode in which the units are currently located. Another requirement is to communicate with the optometer type optoControl 2500, which monitors actual X-axis distance from the obstacle.

Keywords

Desktop application, Java Native Interface, Java, Profibus-DP, DF PROFI II

Poděkování

Rád bych poděkoval v první řadě svému odbornému konzultantovi za všechny jeho rady a postřehy při vývoji aplikace, bez kterých bych se neobešel. Dále bych chtěl poděkovat garantovi RNDr. Jaroslavu Ichovi a nakonec i své rodině za podporu.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

České Budějovice 12.3.2013

Jakub Egner

Obsah

Seznam obrázků

1. Úvod

Bakalářská práce se zabývá vytvořením aplikace pro ovládání posuvu pomocí tří řídících jednotek TLC53F (Twin line controler), kde každá z jednotek řídí jeden servomotor a komunikuje s řídícími jednotkami po průmyslové sběrnici Profibus-DP prostřednictvím karty DF PROFI II.

Požadavek na aplikaci vznikl ve firmě Bosch[1]. Firmě nevyhovuje aplikace od výrobce řídících jednotek, která se dosud používala. Aplikace nevyhovuje z důvodů nepřehlednosti v aplikaci a možnosti ovládat pouze jednu řídící jednotku a z toho plynoucí celkovou pomalou a složitou manipulaci s programem. Dalším důvodem je, že program komunikuje s řídícími jednotkami přes sériové rozhraní, které se nahrazuje sběrnicí Profibus-DP. K programu také neexistuje alternativa, která by splňovala požadavky firmy Bosch.

Název 3D posuvný systém vychází z umístění servomotorů, kdy jeden servomotor je řízen řídící jednotkou ve směru osy X, druhý ve směru osy Y a poslední ve směru osy Z. Posuvný systém se používá k testování snímačů klikové a vačkové hřídele do automobilů, kdy je senzor nejprve usazen do držáku a pak se posuvný systém nastaví na různé vzdálenosti k ozubenému kolu podle specifikace zákazníka, kde jsou posléze měřeny parametry senzoru a ty se následně vyhodnocují.

Programovací jazyk pro vytvoření aplikace byl zvolen jazyk Java. S tímto jazykem mám největší zkušenosti a ostatní aplikace ve firmě jsou také naprogramovány v jazyce Java.

Firma Bosch si od nově vytvořené aplikace slibuje zrychlení nastavení osového systému a jednoduché, přehledné ovládání.

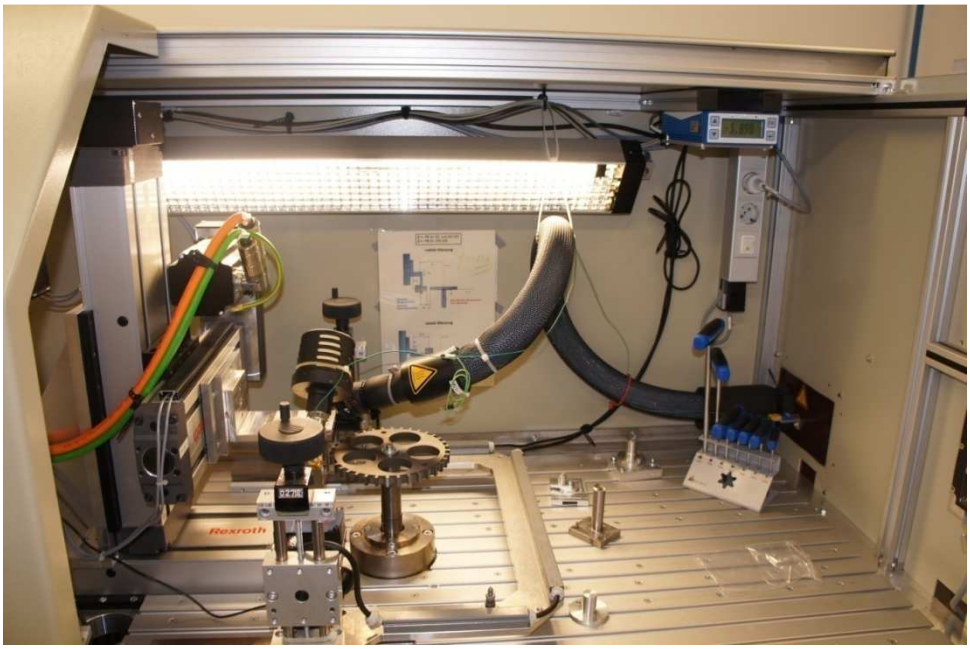
[1]

1 Robert Bosch: České Budějovice. [online]. [cit. 2013-03-28]. Dostupné z: http://www.bosch.cz/cs/cz/our_company_7/locations_7/menu_robert_bosch_spol_sro_ceske_budejovice/budejovice_menu_uvod.html

1.1 Cíle

Cílem bakalářské práce je vytvoření aplikace k řízení posuvného systému po sběrnici Profibus-DP, která implementuje komunikaci s optometrem značky optoControl 2500. Řízení bude umožňovat manipulaci s posuvným systémem tak, aby bylo možno zadávat jednotlivé vzdálenosti pro každou osu zvlášť. Následně po stisku jediného tlačítka se začnou polohovat všechny osy podle zadané vzdálenosti. Typ polohování bude možné volit mezi relativním a absolutním. V masce programu bude možné sledovat aktuální pozici jednotlivých os v milimetrech a také aktuální mód, ve kterém se řídící jednotky nacházejí. Důležitou věcí z hlediska bezpečnosti je zabudování měření mezery mezi osou X a ozubeným kolem, tím získáme přehled, jak daleko od ozubeného kola jsme.

Po vyhodnocení testů v provozu podle scénářů, které mohou při nastavení nastat, se odstraní případné nedostatky.



Obrázek 1- Osový systém pohled z boku

Obrázek 2- Osový systém pohled zepředu

1.2 Popis obsahu práce

Bakalářská práce je rozdělena na dvě hlavní části, které se dále dělí do menších podkapitol.
V první části je popsán celý posuvný systém a jeho jednotlivé části, aby bylo možné získat lepší představu, jak celé zařízení funguje a jaký úkol plní jednotlivé jeho části.
V následující části je pro bližší seznámení s principy komunikace s řídícími jednotkami analyzován původní program dodaný výrobcem, aby bylo zjištěno, co vše je potřebné pro vytvoření nové aplikace.

Dále následuje praktická část. Tato část se skládá z popisu vytvoření návrhu aplikace a programu v jazyce C, který komunikuje s kartou DF PROFI II, bez kterého by program vytvořený v Javě nemohl s kartou vůbec komunikovat.

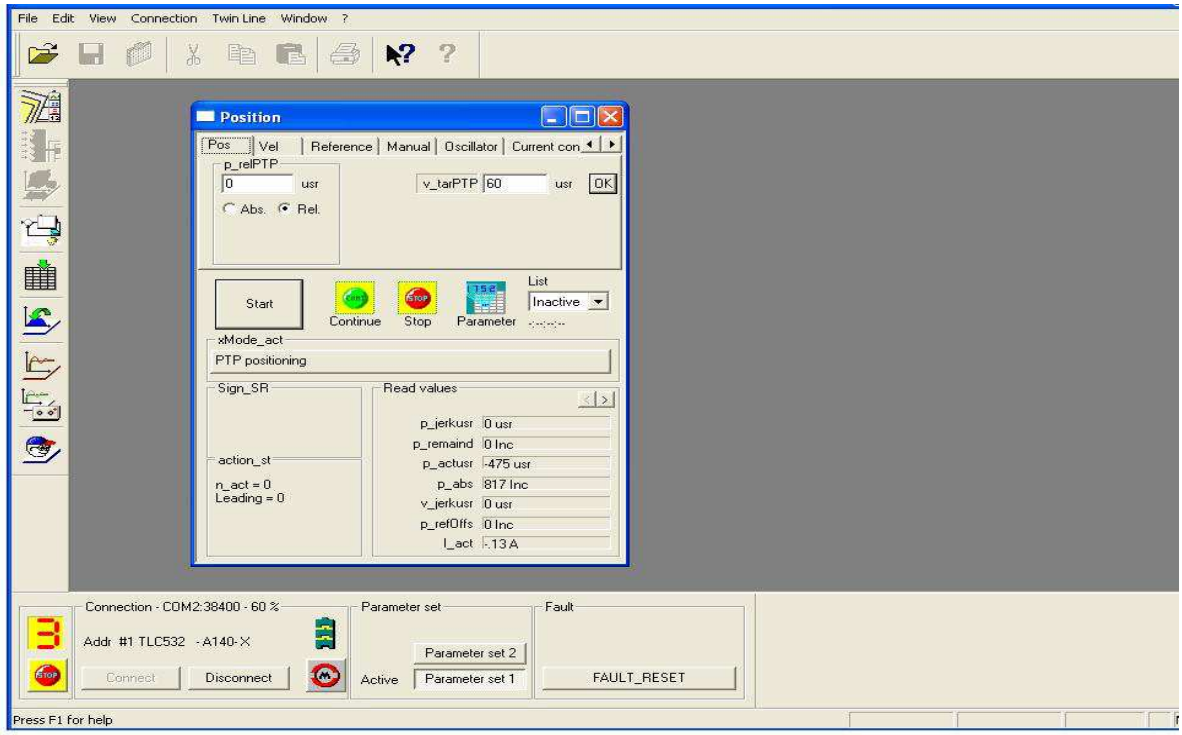
Následuje popsání propojení programu vytvořeného v jazyce C s aplikací vytvořenou v jazyce Java pomocí rozhraní JNI(Java Native Interface). Popis vývoje aplikace v jazyce Java zahrnuje i komunikaci s optometrem přes sériové rozhraní a získáním obrazu z webové kamery.

Poslední část se zabývá testováním aplikace v provozu s uvedením zjištěných nedostatků a vysvětlením kroků vedoucích k nápravě těchto chyb. Na závěr jsou uvedeny možné varianty rozšíření aplikace do budoucna.

2. Analýza původní aplikace k ovládání osového systému

2.1 Popis stávající aplikace

Doposud se používá volně stažitelný[2] servisní program „TwinLine Control Tool“, který je dodáván výrobcem řídících jednotek.



Obrázek 3- Hlavní okno TwinLine Control Tool s komunikací s řídící jednotkou osy X

TwinLine Control Tool pracuje ve spojení s řídícími jednotky řady Twin Line. Je používán pro své rychlé spuštění a snadné řešení problémů s jednotkami. Nepotřebuje žádnou konfiguraci před použitím a může být používán s jakoukoliv z Twin Line jednotek s RS232 rozhraním.

2

[2] TwinLine Control Tool: aplikace. [online]. [cit. 2013-03-28]. Dostupné z: http://www.mmnt.net/db/0/0/ftp.sdt.se/Documentation/BergerLahr/Software/TwinLine%20Control%20Tool%201.046

Funkce programu se automaticky přizpůsobují podle připojeného zařízení. To způsobuje, že některé funkce nejsou přístupny pro určitý druh zařízení.

Funkce programu, které mohou být vykonány se všemi druhy zařízení:

- Zadávání a zobrazování parametrů zařízení
- Archivování a duplikace dat zařízení
- Manuální pozicování motoru
- Nahrávání, hodnocení a archivování cest pohybu
- Off-line a on-line zpracování parametrů a polohování
- Diagnostika provozních poruch
- Uvedení do provozu asistentem pro rychlý start s Twin Line jednotkou

Funkce programu, které jsou závislé na typu zařízení:

- Optimalizace řízení odpovědi
- Programování pozicování
- Zpracování seznamu poloh

2.2 Důvody nahrazení stávající aplikace

Mezi hlavní důvody vedoucí k nahrazení stávající aplikace novým programem patří nepřehlednost aplikace. Aplikace nabízí mnoho funkcí, které ve firmě nenacházejí uplatnění. Aplikace komunikuje skrze sériové rozhraní, které se nahrazuje průmyslovou sběrnicí Profibus-DP.
Program TwinLine Control Tool dokáže najednou komunikovat pouze s jednou z řídících jednotek, což velmi znepříjemňuje nastavení osového systému, kdy je třeba měnit řídící jednotky a přepínání mezi nimi je velice zdlouhavé a komplikuje tak celý proces.

3. Analýza zařízení potřebných při vývoji

3.1 Řídící a výkonové jednotky TLC53F

Obrázek 4- Řídící jednotky TLC55x

Řídící jednotky jsou vyrobeny firmou Berger Lahr, která je dnes známa pod jménem Schneider Electric[3].

Základem řídících jednotek je mikropočítač s funkcemi pro řízení pohonu, jako například polohování z bodu do bodu, řízení otáček, elektronická hřídel nebo převodovka a dalších.

Typ výkonové jednotky ve firmě Bosch je určen pro AC-servomotory a nazývá se TLD 134 a její parametry jsou uvedeny v tabulce níže.

3

3 Schneider Electric. [online]. [cit. 2013-03-28]. Dostupné z: http://www.schneider-electric.com/site/home/index.cfm/cz/

Výkon	Proud	Napájení	Výstupní proud trvalý (A_{ef})	Výstupní proud špičkový (A_{ef})
1,5 kW	3A	400Vac	3	8

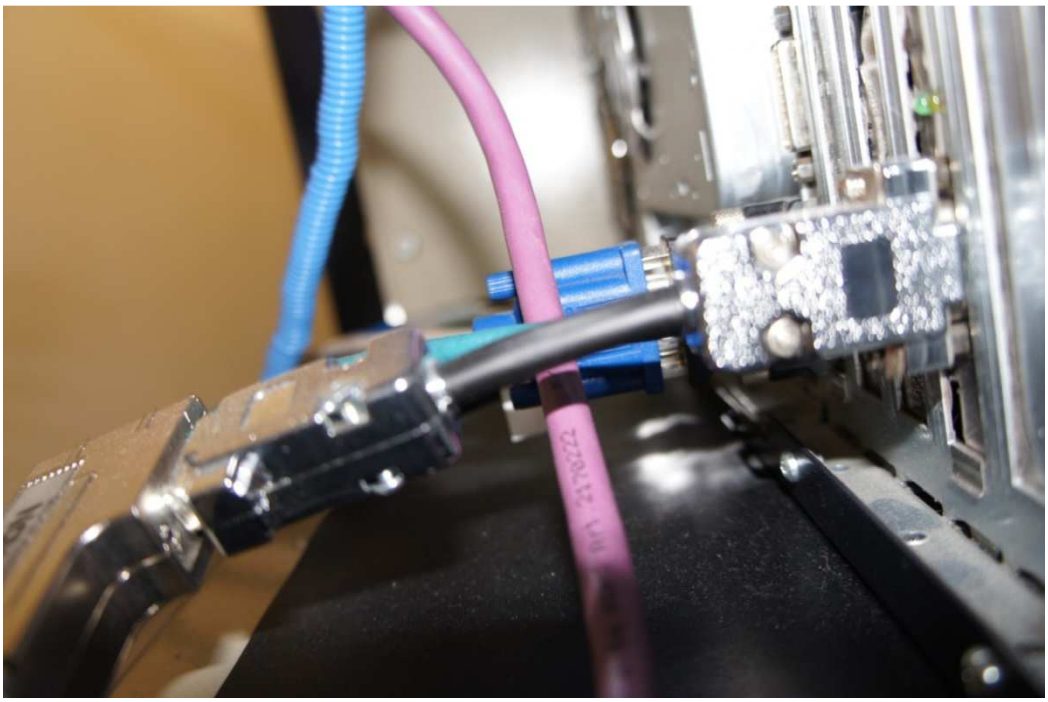
Tabulka 1- Výkonová jednotka TLD 134

Řídicí jednotka má digitální vstupy a výstupy, jejichž funkce je možno nastavit do jednoho ze tří režimů podle způsobu aplikace. Hardwarovou sestavu lze pak dále konfigurovat přídavnými moduly.

Řídicí jednotka je určena pro začlenění do distribuovaného systému řízení s on-line komunikací po některé ze sběrnic jako například RS485, Profibus DP, CAN-bus (protokol CAN-Open), Interbus S, Modbus. [6]

Ve firmě se používají tři řídící jednotky TLC53F. Každá z řídících jednotek řídí jeden ze servomotorů, takže každá z os se může pohybovat nezávisle na ostatních.

3.2 Průmyslová sběrnice Profibus



Obrázek 5- Připojení karty DF PROFI II ke sběrnici Profibus-DP

Profibus je celosvětově nejrozšířenější sběrnicový systém. Přenosovým médiem je stíněná dvoulinka nebo optický kabel.

Je postavena na základech otevřeného komunikačního modelu ISO/OSI a je určena pro všechny oblasti automatizace.[5] Například pro automatizaci výrobních linek (výroba automobilů, plnicí linky, skladové systémy), pro domovní automatizaci (klimatizace, vytápění), procesní automatizaci a pro řízení výroby a distribuce energie.

Historie doposud snad nejúspěšnější sběrnice pro průmyslové použití začala ve skutečnosti v roce 1987, kdy firmy Bosch, Klöckner & Möller, Siemens a část univerzit předložili projekt Profibus německé vládě.[5]

Cílem tohoto projektu bylo vytvořit architekturu komunikačního systému, který by na jedné straně respektoval potřebu připojit na sběrnici malá zařízení a současně vytvořil otevřené rozhraní pro komunikaci různých automatizačních zařízení (programovatelné automaty, operátorské panely, snímače, akční členy atd.).[5]

Ve firmě Bosch je použitá varianta Profibusu s názvem Profibus-DP (Distributed peripherals). Je nejrozšířenější variantou standartu Profibus. Varianta Profibus-DP nachází své uplatnění díky rychlé cyklické výměně dat na nejnižší úrovni komunikačního protokolu tzn. v komunikaci mezi programovatelnými řídicími automaty a jejich decentralizovanými vstupy a výstupy (v/v).
Výměna dat probíhá mezi programovatelným automatem (master) a jednotlivými vstupně/výstupními zařízeními (slave).

3.3 Rozšiřující karta DF PROFI II

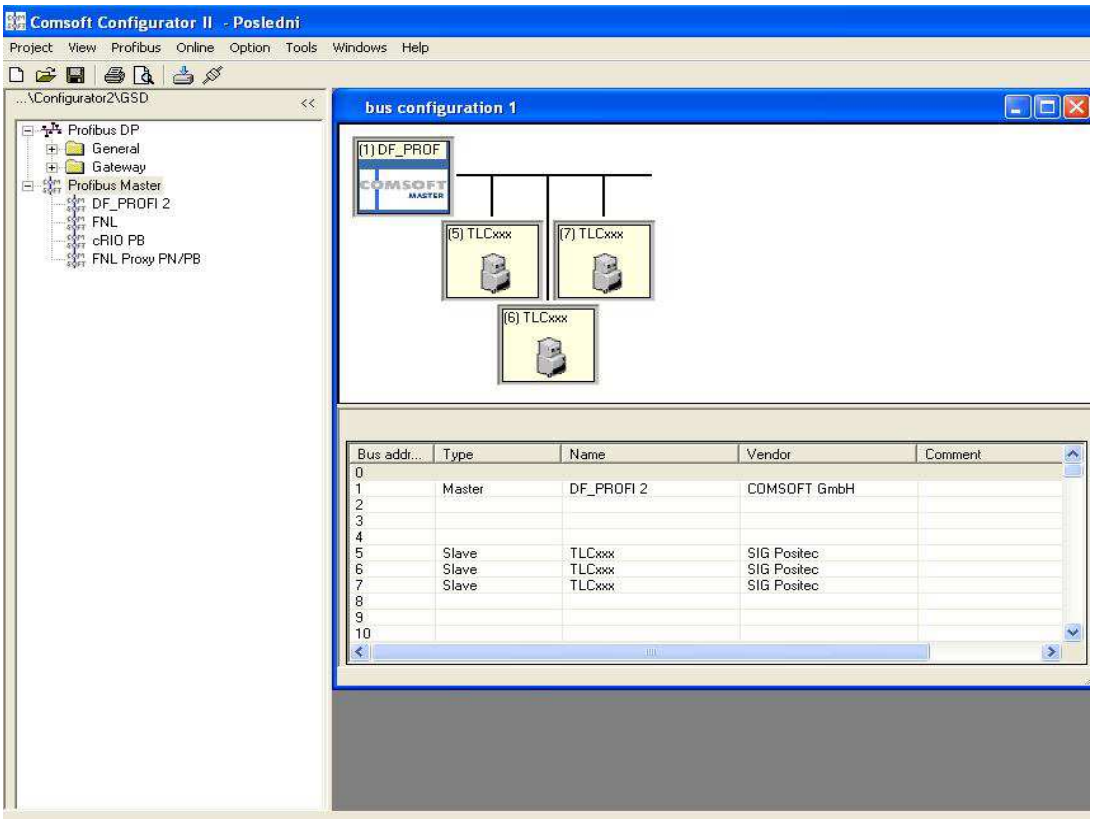
Obrázek 6- Karta DF PROFI II

V našem případě se používá běžné PC jako aktivní řídící master. Aby se běžné PC stalo řídící stanicí je nutností použít speciální rozšiřující karty, které mají vlastní procesor a Profibus mastera implementují.

Ve firmě Bosch využívají kartu DF PROFI II[4] od firmy National Instruments [5], která je zapojena do konektoru PCI sběrnice.

Konfigurace sběrnice Profibus-DP je nutná pro pozdější nastavení karty DF PROFI II do master módu a ostatních zařízení připojených ke sběrnici do módu slave. V našem případě jsou zařízení, které jsou v módu slave řídící jednotky TLC53F. Pro tyto potřeby nastavení je zapotřebí software COMSOFT Configurator II[6].

Při použití softwaru COMSOFT Configurator II je zapotřebí ke všem zařízením zajistit příslušné GSD (Generic Station Description) soubory, které definují komunikační charakteristiku.
Program si načte zařízení a přiřadí mu adresu, na které s ním bude možné komunikovat.



Obrázek 7- Software Comsoft Configurator II

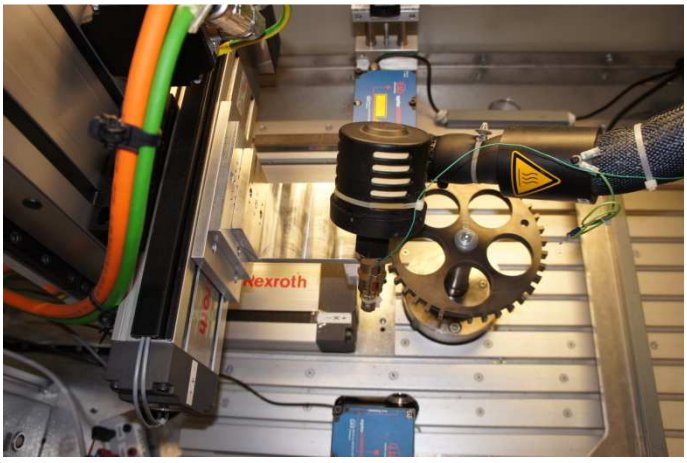
4

4Karta DF PROFI II. [online]. [cit. 2013-03-28]. Dostupné z: http://www.comsoft.de/uploads/media/COMSOFT_Flyer_DF_PROFI_II_e.pdf
5 National Instruments. [online]. [cit. 2013-03-28]. Dostupné z: http://czech.ni.com/
6 Comsoft Configurator II. [online]. [cit. 2013-03-28]. Dostupné z: http://www.ni.com/white-paper/6959/en

3.4 OptoControl 2500

Obrázek 8- OptoControl 2500 displej

OptoControl 2500 je laserový mikrometr vyrobený firmou MicroEpsilon[7] s integrovanou kamerou CCD s vysokým rozlišením pro měření geometrických veličin.[7] OptoControl měří rozměry cíle nebo pozici hrany cíle podle stínového principu. Paralelní světelná závora se vyrábí zdrojem laserového světla. Čtecí kamera v přijímači měří obrys cíle s vysokou přesností pomocí vytvořeného stínu.



Obrázek 9- OptoControl 2500 přijímač a vysílač
5

V našem případě je využit k měření vzdálenosti osy X od ozubeného kola. Jeho činnost je velice důležitá, protože vzdálenost od ozubeného kola se nastavuje až v desetinách milimetru.

7 MicroEpsilon. [online]. [cit. 2013-03-28]. Dostupné z: http://www.micro-epsilon.com/index.html

Tato vzdálenost není již rozeznatelná pouhým okem. Případný střet osy s ozubeným kolem by mohl způsobit poškození měřeného senzoru.

4. Návrh řešení aplikace

4.1 Metodika

Pro vývoj aplikace byla použita metodika označovaná jako vodopádový model.
Struktura vodopádového modelu:

1. definice problému
2. specifikace požadavků
3. návrh
4. implementace
5. testování
6. provoz a údržba

První dva kroky byly prováděny ve spolupráci se zadavatelem (firmou Bosch), který definoval problém a specifikoval požadavky na aplikaci. V průběhu komunikace se zadavatelem se postupně dospělo k přesnému vymezení požadavků a uvedení do celé problematiky. Po bližším seznámení byla následně specifikace upravena.

Pro získání názorné představy o požadované funkcionalitě programu byl vytvořen diagram použití.

Následně byla provedena analýza doposud používaného původního programu. Na jejím základě byly zjištěny funkce, které bylo potřeba zahrnout do nově vytvářené aplikace. Dále bylo nutné nalézt příslušné funkce v manuálu pro řídící jednotky.

Podle zjištěných informací a vytvořeného návrhu bylo vytvořeno řešení, které bylo následně představeno zadavatelům. Dokončení aplikace bylo otestováno v provozu, aby byly vyzkoušeny všechny různé situace, které mohou nastat při chodu programu. Po testování následovalo opravení zjištěných nedostatků.

4.2 Analýza zdrojového kódu ke kartě DF PROFI II

4.2.1Popis ukázkového programu

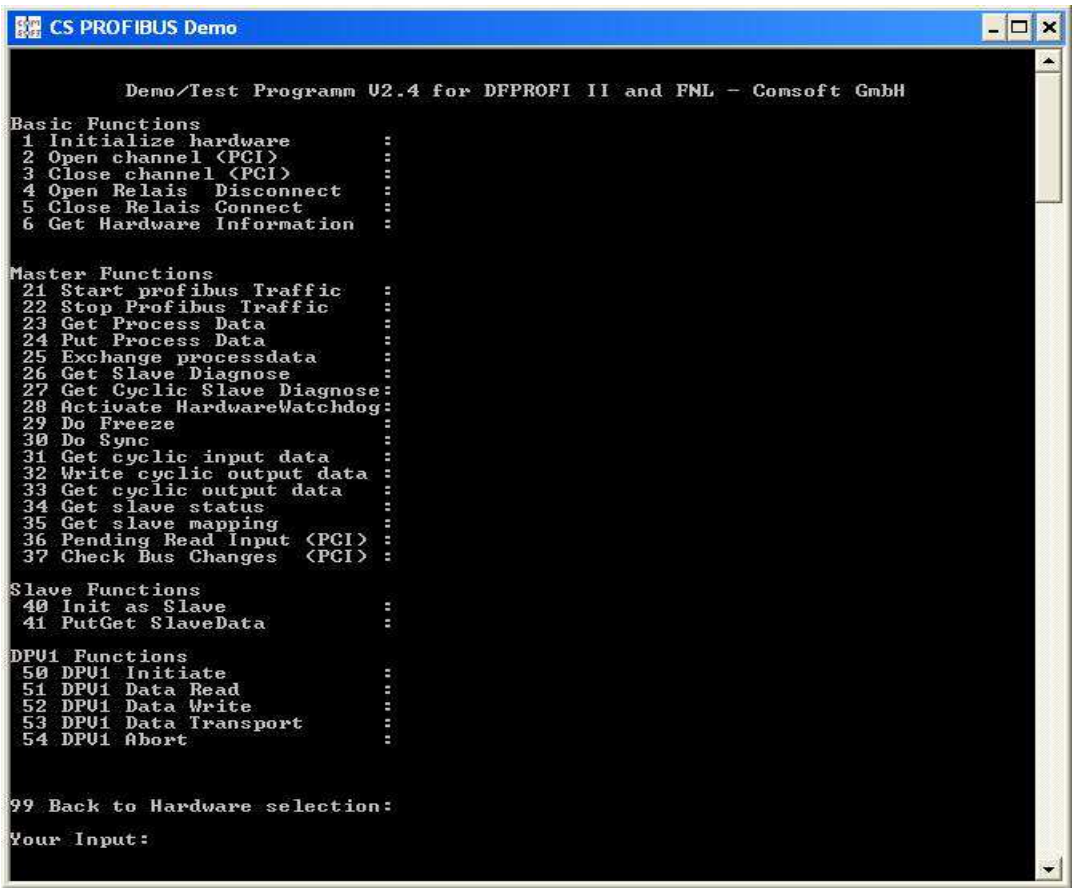
K vytvoření vlastního programu na ovládání řídících jednotek se vycházelo z ukázkových programů pro kartu DF PROFI II dodaných firmou National Instruments.

Konkrétně se jedná o tyto programy: CS PROFIBUS Demo a DFPrfi_II_DP_MASTER_PCI_DEMO.

Oba programy jsou konzolové aplikace, napsané v jazyce C, ve kterých je ukázáno, jak probíhá komunikace po sběrnici Profibus-DP.

Z ukázkových programů lze po bližším zkoumání zjistit, jak správně inicializovat kartu DF PROFI II a načíst k ní příslušné ovladače. Dále jak otevřít komunikační kanál mezi kartou DF PROFI a řídícími jednotkami a po otevření kanálu jakým způsobem posílat příkazy na řídící jednotky.

Programy obsahují plno dalších funkcí, ale ne všechny byly nutné pro vytvoření nové aplikace.



Obrázek 10- CS PROFIBUS Demo ukázka nabízených funkcí

Před použitím programu je potřeba mít nastavenou kartu do módu master.
Z komentářů a pomocí debugu programu zjišťujem, že prvně je nutné definování vlastností hlavní struktury, která reprezentuje cílové zařízení, se kterým program komunikuje což v našem případě je karta DF PROFI II. Dále je nutné implementování inicializace knihovny *dfxx.dll* a vytvoření hardwarového přístupu ke kartě. Následuje otevření komunikačního kanálu a pak už samotný začátek cyklické komunikace, která je standardní Profibusovou komunikací. Po ukončení komunikace je potřeba uzavřít komunikační kanál.

5. Softwarové technologie použité při vývoji aplikace

5.1 Java Native Interface (JNI)

Java Native Interface je rozhraní, které umožňuje propojení kódu běžícího na virtuálním stroji Javy s nativními programy a knihovnami, které nejsou napsány v jazyce Java. [8]
JNI je nutné, pokud chceme využít specifická hardwarová zařízení, která nejsou přístupná přes virtuální stroj a pro která výrobce neposkytuje javovské rozhraní.

Jiným důvodem pro použití může být, že nativní kód se vykonává rychleji. Java Native interface představuje obousměrné spojení mezi nativním kódem a kódem napsaným v jazyce Java.

5.1.1 Nevýhody používání JNI

• Ztrácíme jednu ze základních výhod jazyku Java, a to podporu nezávislosti na platformě.

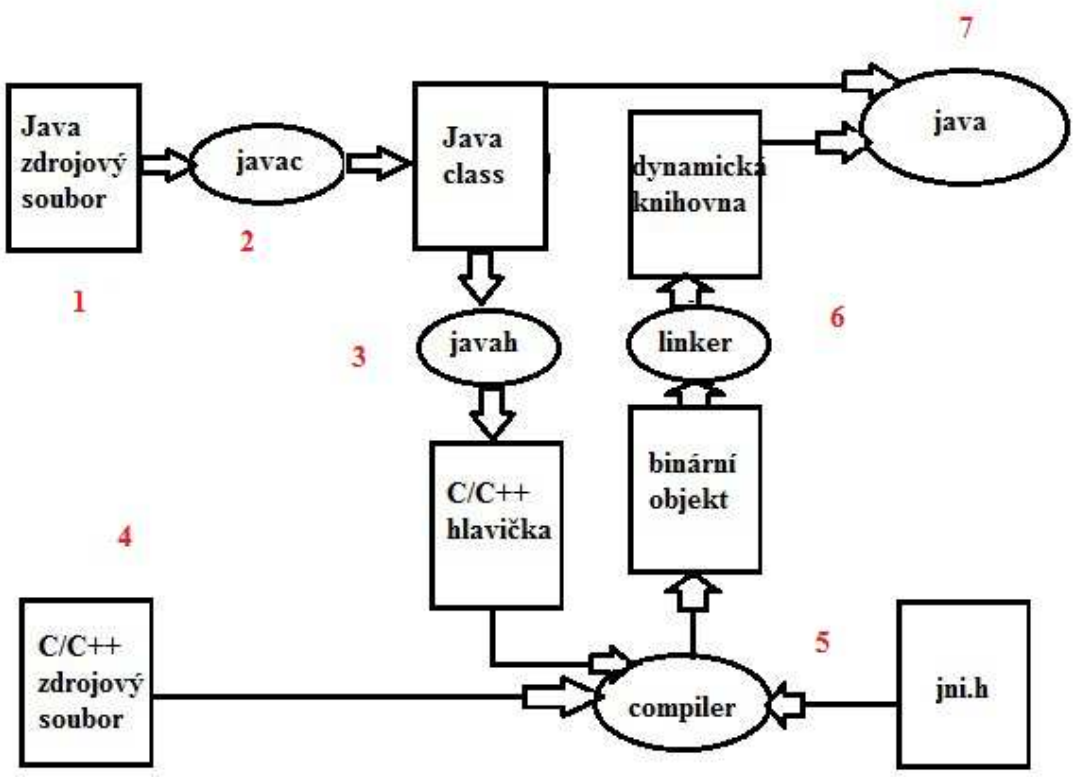
• Další nevýhodou je, že na rozdíl od Javy, která umí být typově bezpečná, tak jiné nativní jazyky jako např. C nebo C++ to neumí. Z toho důvodu může i drobná chyba v implementaci nativní metody způsobit pád celé aplikace a navíc tyto chyby je velmi těžké dohledávat a ladit.

• Poslední z hlavních nevýhod může být, že JNI neposkytuje žádné prostředky pro automatické uvolnění nepotřebných zdrojů z paměti. Odpovědnost za jejich uvolnění ve chvíli, kdy už nejsou potřebné, nese nativní kód.

5.1.2 Fungování JNI

Aby bylo možné z Javy volat nativní knihovny, je potřeba udělat následující kroky.

1. Vytvoření javovské třídy, která deklaruje nativní metody ve stylu
   např. *public native int Initialize();*
2. Přeložení této třídy do byte kódu
3. Vytvoření hlavičkového souboru (pro C/C++) se signaturou příslušných metod, které jsou připraveny pro následnou implementaci
4. Samotná implementace metod v jazyce (C/C++) používáme vytvořený hlavičkový soubor z předchozího kroku, kde je již vytvořena signatura metod a soubor pro definování typů ze složky *include/jni.h.*
5. Zkompilování souboru s C/C++ kódem
6. Použití linkeru pro vytvoření nativní knihovny
7. Spuštění knihovny v kódu v Javě, která načítá vytvořenou nativní knihovnu

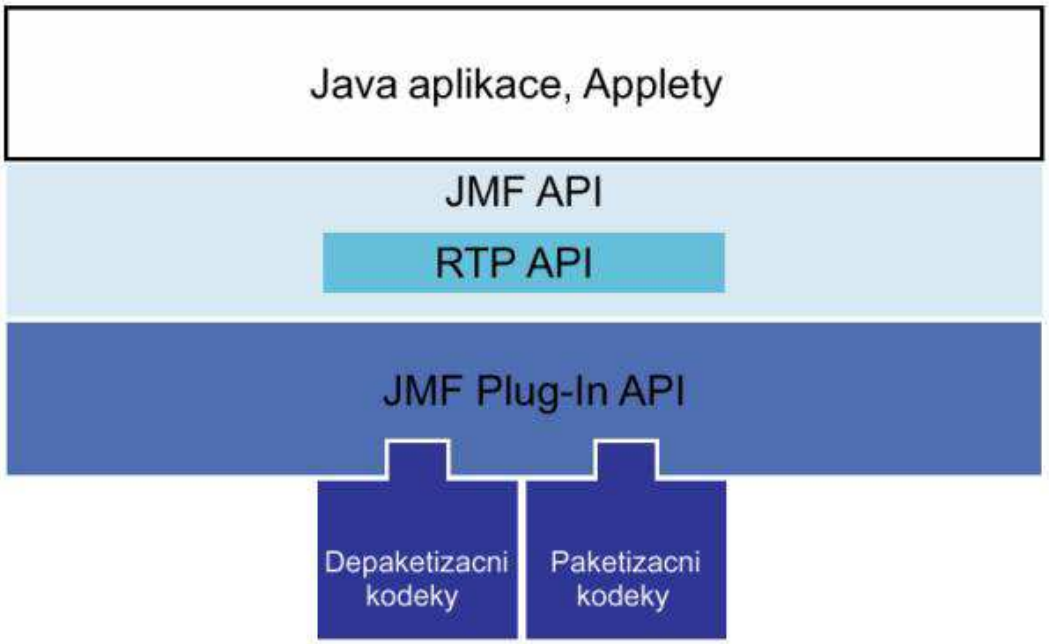


Obrázek 11- Fungování JNI

5.3 JMF(Java Media Framework)

JMF je API umožňující zpracovávat a přehrávat audio, video a jiná časově založená data a začleňovat je do Java aplikací a appletů. Aktuální verze JMF je 2.1.1e, která je dostupná ve třech verzích, a to platformě nezávislá, pro Linux a pro Windows na adrese http://www.oracle.com/technetwork/java/javase/download-142937.html.

Tento framework byl potřeba pro práci s webkamerou, protože při vývoji aplikace vznikl rozšiřující požadavek na implementaci webkamery Logitech Webcam Pro 9000 do aplikace.



Obrázek 12- Java Media Framework architektura

5.4 Sériová komunikace

Pro implementaci sériové komunikace s optometrem byla použita již vytvořená a dostupná knihovnu RXTX. Knihovna RXTX je nativní knihovna, která poskytuje sériovou a paralelní komunikaci pro Java Development Toolkit (JDK).
RXTX je poskytován pod GNU LGPL licencí a poskytuje stejnou podporu jako javax.comm library.

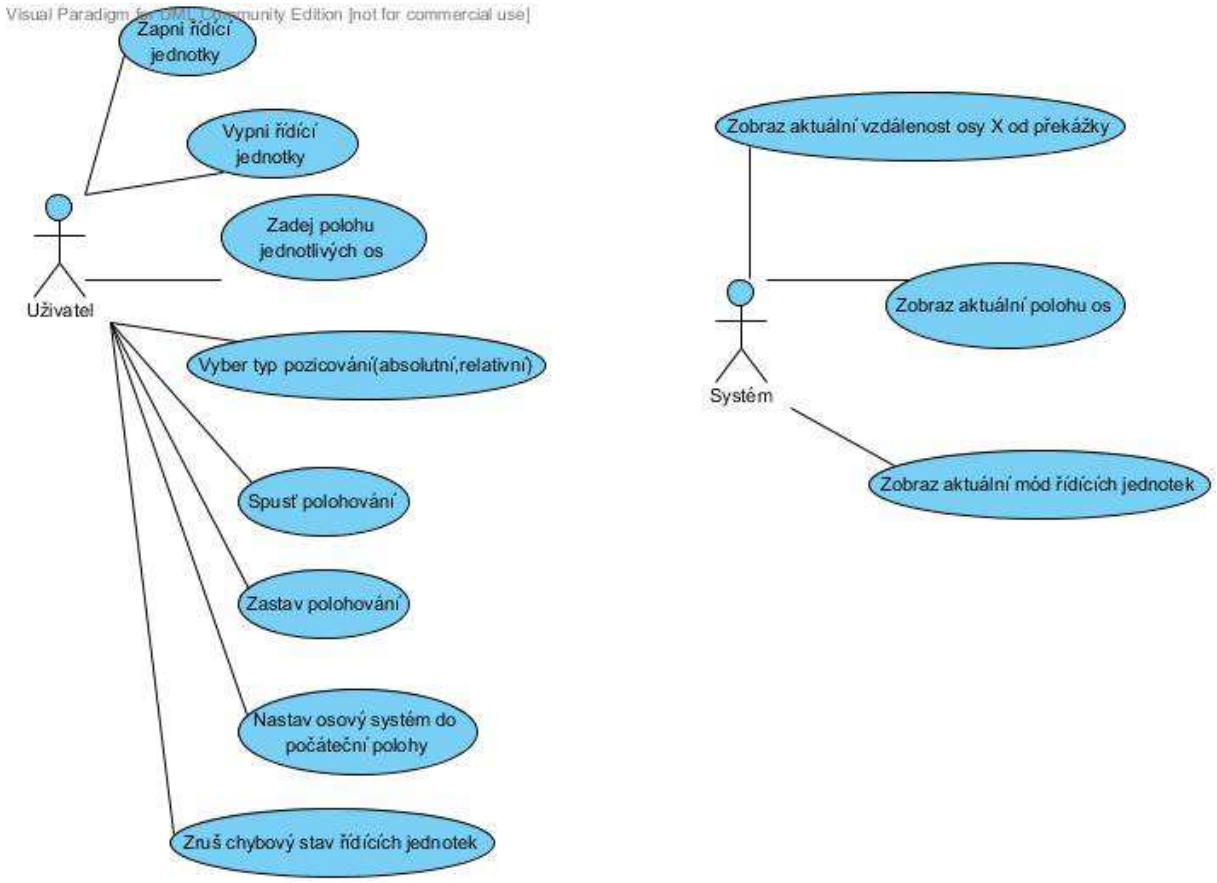
5.5 ANT (Another Neat Tool)

Tento nástroj je nazýván také jako sestavovací utilita, která umožňuje napsat sestavovací schémata pro aplikace.[4] Soubory jsou založeny na formátu XML, takže jsou plně přenositelné.
Činnost se řídí souborem *build.xml*, ve kterém se definuje vše k provedení a tak nám šetří čas při opakujících se činnostech jako je třeba build aplikace a vytvoření jaru pro distribuci.

Nástroj ANT je již implementován v prostředí Eclipse, které bylo během vývoje také použito.

6. Postup vývoje aplikace

6.1 Návrh aplikace



Obrázek 13- Případ užití

Podle stanovených požadavků na aplikaci, dostupných manuálů k řídícím jednotkám, sběrnici Profibus a nakonec i analýzy dosavadního programu TwinLine Control Tool byl v prvním kroku vytvořen diagram případů užití, pro lepší představu požadované funkcionality programu.

Mezi vybrané funkce patří zapnutí a vypnutí řídících jednotek, výběr typu polohování mezi absolutním polohováním, kdy se osa pohybuje podle své polohy po inicializace, a relativním polohováním kdy se osy pohybují vůči aktuální pozici. Další funkce jsou spuštění polohování osového systému a jeho následné zastavení. Zastavení osového systému je velice důležité, aby v případě chybně zadané vzdálenosti nedošlo k nárazu do ozubeného kola. Neméně důležitou funkcí je funkce pro nastavení osového systému do počáteční polohy, kterou je potřeba spustit vždy, když byl osový systém odpojen od zdroje elektrické energie a je znova nastartován a

potřebujeme se pohybovat absolutně. Důvodem je, že jednotky po vypnutí ztrácejí informaci o své aktuální poloze. Poslední funkcí je funkce na zrušení chybového stavu řídících jednotek, chybové stavy jsou Quick Stop a Fault a jsou označeny číslem 7 a 9. Do prvního stavu se dostaneme v případě, že se jednotky dostanou na koncová čidla a nemohou se dále pohybovat.

Do druhého stavu se dostaneme po otevření prostoru měřícího stavu. V tomto případě se automaticky nastaví stav Fault, který zablokuje osový systém a není možné s ním pohybovat, dokud nedojde k zrušení tohoto stavu. Nastavení do stavu Fault je prováděno z důvodu bezpečnosti, aby po ukončení manipulace s osovým systémem se nemohl již osový systém pohybovat.

Funkce, které plní program sám bez vnějšího požadavku od uživatele, je zobrazení aktuální vzdálenosti osy X od ozubeného kola, kterou program odečítá z optometru.
Další funkcí je zobrazení aktuální polohy osového systému, kdy je zobrazena aktuální poloha os (počítáno od počáteční polohy). Poslední funkcí je zobrazování aktuálního módu řídích jednotek, kdy je zobrazen číselně aktuální stav odpovídající manuálu.

6.2 Vytvoření komunikace s kartou DF PROFI II

Prvním z mnoha úkolů, pro dosažení stanoveného cíle bylo programové spuštění karty DF PROFI II a otevření komunikačního kanálu tak, aby bylo možné posílat příkazy na řídící jednotky pro ovládání, a následně po skončení komunikace uzavřít komunikační kanál.

Pro tuto potřebu bylo nutné na základě dostupných manuálu ke kartě DF PROFI II a z ukázkových programů v jazyce C pochopit, jak karta funguje a co je potřeba pro správnou inicializaci karty.
Na základě studia přiložených programů a dlouhého zkoušení se dospělo ke zjištění, že v prvním kroku, který je potřeba udělat, je načtení příslušných ovladačů ke kartě DF PROFI II s následným načtením dynamické knihovny *dfxx.dll*, která spravuje kartu. Dále bylo potřeba vytvořit standardní proud pro hardwarový přístup.

Poté následuje už samotné otevření komunikačního kanálu a začátek komunikace se zařízeními připojenými na sběrnici Profibus-DP.

Ve vytvořené aplikaci se vše popsané nachází pod funkcí ***public native int*** *Initialize();*.

6.3 Komunikace s řídícími jednotkami

Po vytvoření inicializace je další krokem komunikace s řídícími jednotkami TLC53, které jsou přiřazeny sběrnici v módu slave.
Z programu COMSOFT Configurator II zjistíme, jaké adresy byly přiřazeny jednotlivým řídícím jednotkám.
To je důležité zjistit proto, abychom mohli komunikovat s každou jednotkou zvlášť, jako v případě metody *public native int SettingAbsPosition(int adresaOsy,int hodnota);*.
Metoda slouží k absolutnímu pozicování příslušné osy, má dva parametry a to adresu osy zadávanou v hexadecimálním tvaru a samotnou číselnou hodnotu, na kterou se má osa posunout.
Program v jazyce C si převezme parametry, nejdříve podle velikosti zadané hodnoty rozdělí na příslušný počet bytů a předá je i s adresou a kódem příkazu funkci *WriteCommand.*Tato funkce se postará o připravu všeho, co je nutné pro potřebu komunikace (typ komunikace, číslo kanálu atd.) a zajišťuje i samotné poslání řídící jednotce. O konfiguraci dat, která se odesílají, se stará metoda *ConfigureOutput.* Metodě je předána adresa řídící jednotky na kterou se data budou posílat. Data, která se zasílají jsou v datové struktuře *ProcessImage.*

6.4 Vytvoření knihovny z vytvořeného programu

6.4.1 Vytvoření třídy deklarující nativní metodu

Nativní metody se v třídě s názvem *FunkceOvladani* označují klíčovým slovem *native* a jejich signatura se ukončuje pomocí středníku. Podobně, se píše metoda v rozhraní. Důležitou věcí je, že před použitím příslušných metoda je nutné nejprve nahrát knihovnu, která obsahuje už

napsané implementace jednotlivých metod. Nahrání knihovny se provádí pomocí metody *System.loadLibrary()*, kde jako parametr musí být uveden název knihovny. Pro nově vytvářenou aplikaci je volání knihovny ze statického bloku uvedeno níže.

```
{
        System.loadLibrary("Profi");
}
```

V této třídě jsou uvedeny všechny metody, které se volají z vytvořeného kódu v jazyce C. Patří mezi ně metoda na inicializaci karty DF PROFI II a vytvoření komunikačního kanálu a metoda pro nastavení polohy jednotlivých os.

6.4.2 Přeložení třídy do byte kódu

Po definování potřebných metod třídu přeložíme, aby bylo možné pokračovat dále. O přeložení se většina vývojových prostředí postará automaticky nebo je možnost z příkazové řádky zavoláním překladače.

javac FunkceOvladani.java

Výsledkem pak bude soubor *FunkceOvladani.class* již v bytekódu.

6.4.3 Vytvoření souboru se signaturou metod

Pro implementaci metody v jazyce C je potřeba znát její signaturu. Ta odpovídá definici ze zdrojového kódu Javy. Tato hlavička se od původní odlišuje a pro získání jejího tvaru potřebného pro implementaci v jazyce C je potřeba použít nástroj *javah* z příkazové řadky, který využívá soubory s bytekódem. Proto bylo potřeba třídu nejdříve přeložit. Zavolání tak vypadá takto.

javah –jni FunkceOvladani

Po zpracování získáme hlavičkový soubor *FunkceOvladani.h* ve kterém jsou uvedeny signatury metod tak, jak je potřebujeme. Ukázka signatury jedné z metod.

```
JNIEXPORT jint JNICALL Java_FunkceOvladani_Initialize
  (JNIEnv *, jobject);
```

Nativní metoda obdrží od JNI dva argumenty, i když původní metoda v Javě byla bez parametrů. Prvním z nich je *JNIEnv* ukazatel na rozhraní JNI a druhým je *jobject*, který představuje referenci na javovský objekt, který metodu zavolal.

Tyto dva parametry obsahuje každá nativní metoda volaná prostřednictvím JNI.

6.4.4 Implementace nativní metody

Před psaním implementace nativních metod je nutné udělat include vytvořeného souboru *FunkceOvladani.h* se signaturami metod a dodržet volací konvenci jazyka C.

```
#include "FunkceOvladani.h"

JNIEXPORT jint JNICALL Java_FunkceOvladani_Initialize (JNIEnv *, jobject){
        InitializeDfxxDll(&hConsoleOutput);
        InitPCIHardware(&ComPartner);
        OpenChannel(&ComPartner) ;
        StartProfibus(&ComPartner,&ManageProfBusReq);

}
```

6.4.5 Překlad nativní knihovny

Po vytvoření zdrojového kódu, kód zkompilujeme do nativní knihovny, která se poté v programu použije. Tento krok je závislý na použitém kompilátoru. V tomto případě byl použit *cl.exe* kompilátor, který vytváří dynamicky linkované knihovny (dll). Vždy je potřeba nutné správně uvést cesty ke vkládaným souborům. Dále musíme přidat soubor *jni.h*, který najdeme v podadresáři include adresáře, ve kterém je nainstalována Java SDK. Další cestu je potřeba uvést k podadresáři, který odpovídá operačnímu systému jako například *C:\include\win32*. Celý příkaz vypadá takto.

```
cl –o Profi.dll Profibus.cpp /I C:\include\java6\jni.h /I C:\include\win32 /link /DLL
```

6.5 Vytvoření návrhu GUI pro aplikaci

Nezávisle na vytváření programu v jazyce C, byl navržen vzhled aplikace, aby obsahoval všechny požadované funkce a splňoval požadavky na přehlednost a jednoduchost. Po schválení bylo vytvořeno v Eclipse grafické uživatelské rozhraní.

Uživatel má možnost z rozhraní zapnout nebo vypnout všechny řídící jednotky, uvést osy do počáteční polohy, polohovat jednotlivé osy, vybírat typ polohování, zastavení pohybu os a rušit chybový stav řídících jednotek.

Ostatní komponenty pomáhají zjišťovat aktuální stav. Patří mezi ně zobrazení aktuálního stavu jednotek, kdy se číselně zobrazuje aktuální mód, ve kterém se jednotky nacházejí. Dále ukazatel vzdálenosti osy X od ozubeného kola odečítaný z optometru a aktuální poloha jednotlivých os v mm.

Obrázek 14- Návrh GUI aplikace

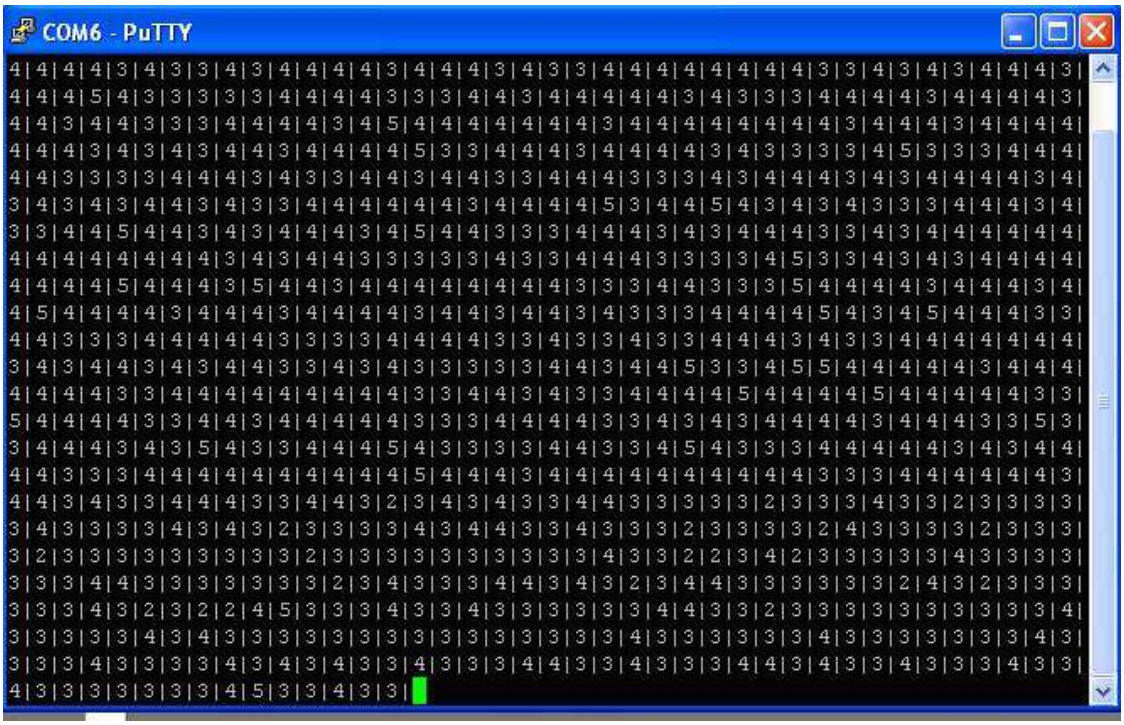
6.6 Integrace komunikace s optometrem

Poslední věcí ze specifikace zadání je integrování komunikace s optometrem. Získávání vzdálenosti z optometru je velice důležité, protože hodnota nám udává mezeru mezi senzorem a ozubeným kolem. A tak máme přehled o vzdálenosti, o kterou se ještě můžeme pohybovat. Toto odečítání bylo zakomponováno do aplikace, abychom nemuseli neustále odcházet od PC a sami si kontrolovat vzdálenost z displeje.

Optometr je připojen pomocí převodníku USB na RS232 který vytvoří v PC virtuální sériový port. Nejdříve se musel vyřešit problém, jak komunikovat se zařízením připojeným jako sériové zařízení. Po delším zkoumání bylo zjištěno, že vše potřebné již existuje napsané v jazyce Java.

Z internetových stránek[7] byl stažen příslušný jar soubor s nativními knihovnami poskytujícími sériovou a paralelní komunikaci. Po rozbalení byla v projektu přidána reference na nativní knihovnu a mohl jsem začít programovat komunikaci s optometrem. Obzvláště důležité bylo správné nastavení rychlosti přenosu, počtu stop bitů, parity a nakonec počet datových bitů tak aby odpovídalo nastavení optometru.

Z manuálu k optometru a po použití programu *Putty*, bylo zjištěno, že optometr automaticky posílá data na výstup, takže není potřeba příkaz, kterým by si data vyžádal. V manuálu se dočteme, že vždy je na výstup poslána sekvence tří bytů L-byte, M-byte a H-byte, které se poznají podle hodnot na posledních dvou bitech. Toto umožnilo rozpoznat jaký byte právě je na výstupu a sestavit z toho celou hodnotu. [6]



Obrázek 15- Komunikace s optometrem přes program Putty

Na základě vztahu z manuálu[8], ***MV(mm) =DV * 34.4386 / 65519 - 0.2221***, který je přepočtem z digitální hodnoty na naměřenou hodnotu, se získalo číslo odpovídající tomu, které je právě zobrazeno na displeji.

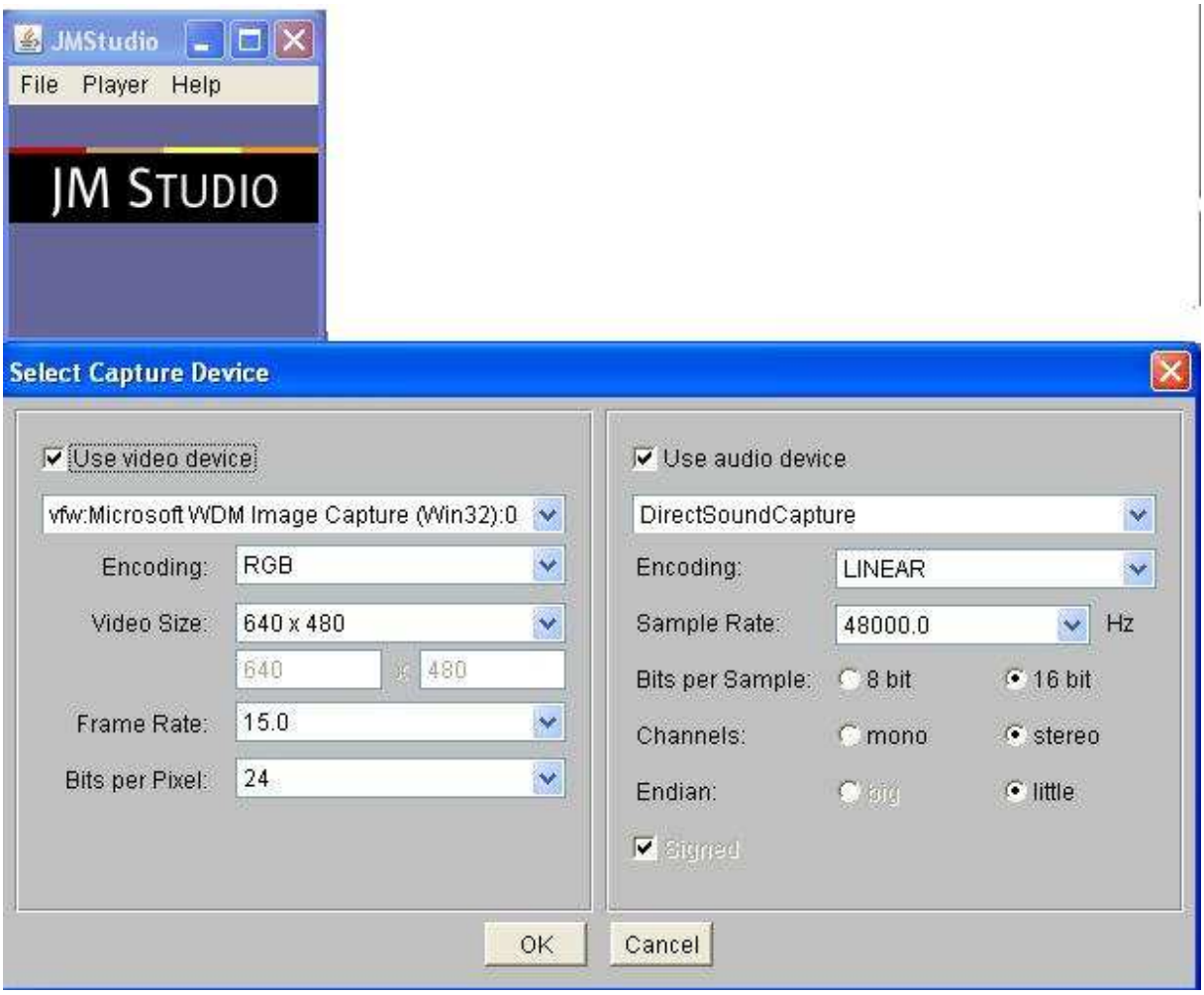
MV- celková naměřená hodnota

DV- digitální hodnota

7 Jar pro Sériovou komunikaci. [online]. [cit. 2013-03-28]. Dostupné z: http://www.kuligowski.pl/java/rs232-in-java-for-windows,1

6.7 Integrace obrazu z webové kamery do GUI programu

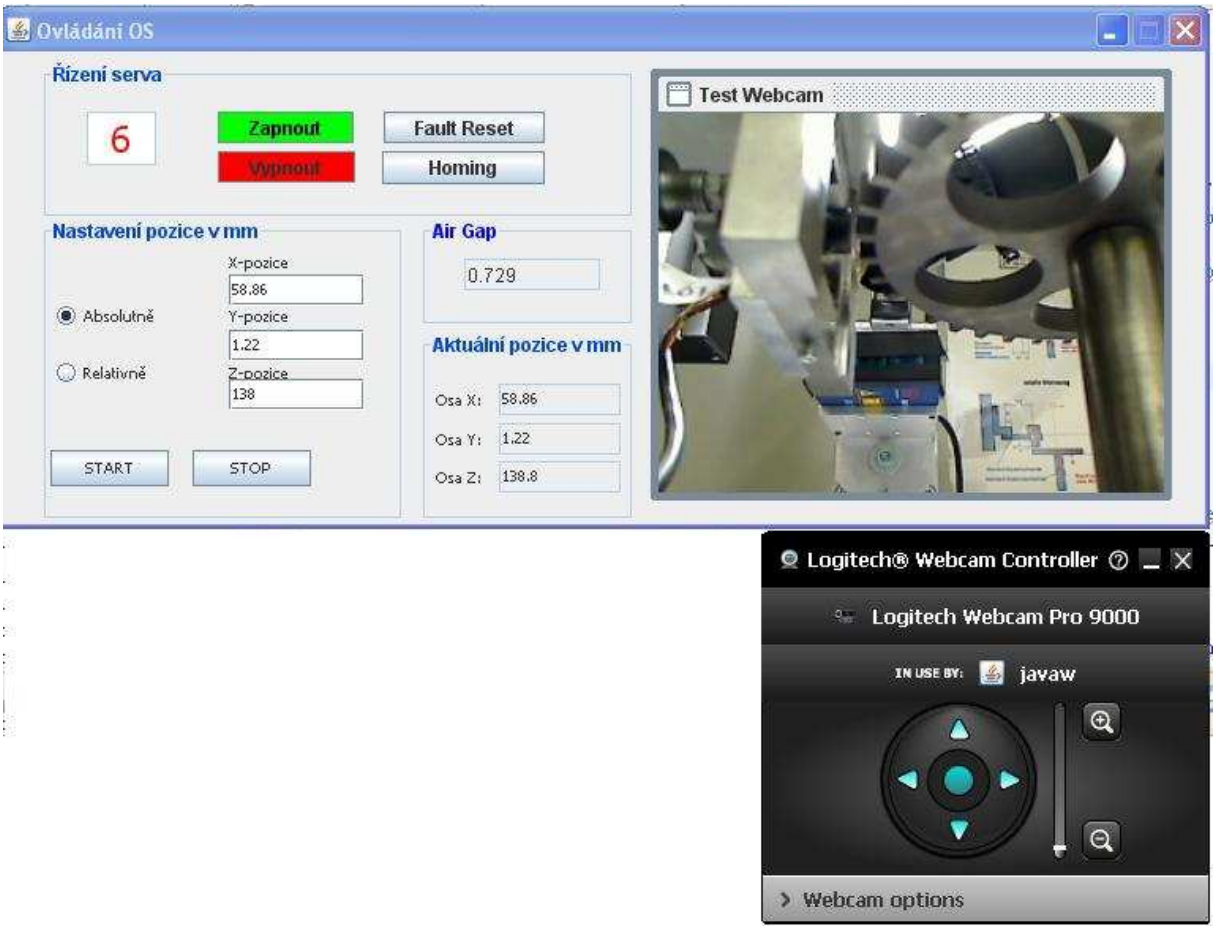
Těsně před dokončením programu vznikl požadavek na implementaci kamery Logitech Webcam Pro 9000 do aplikace. Důvodem bylo získání lepšího přehledu a představy o vzdálenosti osového systému od ozubeného kola.



Obrázek 16- Nastavení webové kamery v JMStudio

Pro integraci obrazu z webové kamery byl použit Java Media Framework (JMF). Po instalaci poslední verze JMF balíku se vytvořil adresář, ve kterém se nachází tři podadresáře včetně adresáře *lib*, kde se nachází soubory s příponou *jar*, které je potřeba načíst do aplikace.
Dále se vytvoří ikona s názvem *JMStudio*, které je důležité pro zachytávání obrazu z webkamery. Z položky si vybereme položku *JMFRegistryEditor*, kde najdeme všechny kodeky a webovou kameru po instalaci ovladačů.
Nic už nebrání použití webové kamery v programu podle ukázek na internetu[9].

7



Obrázek 17- GUI aplikace s integrovanou kamerou

6.8 Nasazení aplikace a testování

Po vytvoření aplikace následovalo dlouhé období testování. Před testováním byla aplikace představena zadavatelům ve firmě Bosch a teprve po jejich schválení následovalo finální povolení k testování. Vše probíhalo ve spolupráci se zaměstnanci, kteří budou aplikaci každodenně používat. Testování v první fázi probíhalo bez senzoru a ozubeného kola tak, aby v případě chyby programu nedošlo k jejich poškození. Teprve po úspěšném testu se začalo s

8 OptoControl 2500: MicroEpsilon. [online]. [cit. 2013-03-28]. Dostupné z: http://www.micro-epsilon.com/download/manuals/man--optoCONTROL-2500--en.pdf

9 WebCam. [online]. [cit. 2013-03-28]. Dostupné z: http://puneetk.com/http://puneetk.com/wp-content/uploads/2010/02/MyCam.txt

testováním v provozu za opravdových pracovních podmínek a požadavků, kdy se testovaly různé situace, které mohou nastat.
První věcí, která se testovala, bylo polohování absolutní a relativní. Ověřovalo se, zda se osový systém pokaždé dostane na stejnou vzdálenost a jestli odpovídá aktuální vzdálenosti, kterou ukazuje program.
Důležitou věcí pro testování bylo ověření funkce tlačítka Stop, které musí být schopné zastavit pohyb osového systému například v případě špatně zadaných hodnot, a tak zabránit poškození jednotlivých částí měřícího zařízení.

Jediný zjištěný nedostatek, který byl opraven, byl převod z jednotek *usr*, se kterými pracují řídící jednotky na *mm*, kde vztah je *1usr=100mm*. Jinak se další nedostatek zatím nenašel, který by bylo třeba opravit.

Teprve po zhruba měsíc a půl dlouhém testování se dá říci, že aplikace neobsahuje žádné nedostatky z hlediska funkčnosti a může se bez problémů používat dále.

7. Návrhy pro budoucí řešení

Jeden z návrhů na rozšíření aplikace přišel už během vývoje a tím bylo rozšíření aplikace o obraz z webové kamery Logitech Webcam Pro 9000, která snímá prostor u ozubeného kola. Další větší rozšíření aplikace se zatím neplánuje, aplikace už funguje v plném provozu a změny, které se na ni zatím provedly, byly jen drobného charakteru. Jediná změna, která se do budoucna plánuje je výměna současného optometru OptoControl 2500 za jeho novější verzi a to OptoControl 2600, kvůli jeho větší přesnosti.

Návrhem do budoucna je implementace vytvořené aplikace do programu pro měření senzorů, kdy by nebylo potřeba spouštět dvě aplikace odděleně. S tím se nabízí další možné zlepšení a to vytvoření automatické aplikace, kdy by lidská obsluha programu byla minimální, tvořila by jí pouze výměna senzorů v držáku. Programu by tak stačilo zadat pouze vstupní hodnoty zapsané například do souborů a podle toho by si sám nastavil osový systém a provedl měření.

Toto řešení je ovšem složitější a je při něm větší šance na selhání a tím poškození měřícího zařízení. Rovněž vzniká otázka, zda by byla časová úspora taková, aby se to vůbec vyplatilo.

Posledním možným rozšířením aplikace by mohlo být přidání některých funkcí. Například přidání možnosti pro změnu rychlosti pohybu os.

8. Závěr

Výsledkem práce je kompletní desktopová aplikace. Aplikace splnila všechny cíle, které byly vytyčeny firmou Robert Bosch. Vývoj programu trval kvůli svému rozsahu a náročnosti déle než tři měsíce usilovné každodenní práce bez započtení období testování. Práce na aplikaci byla velice zajímavá a náročná. Při práci na programu bylo použito mnoho různých technologií a zařízení takže práce byla i velice rozmanitá.

Na začátku jsem věnoval delší čas studiu manuálů k jednotlivým zařízením, ale hlavně seznámení s rozhraním JNI, které jsem do té doby neznal a které bylo základem pro vytvoření požadované aplikace. Důležitá byla komunikace se zadavatelem, z důvodu bližšího objasnění požadovaného cíle a seznámení se s měřícím zařízením a jeho principem.

Vytvoření takto rozsáhlé aplikace mi přineslo mnoho zkušeností, které určitě využiju v budoucnu. Také jsem si dokázal, že jsem schopen se vypořádat s úkolem, který byl velice náročný, pomocí znalostí získaných při studiu.

9. Seznam použitých zdrojů

Herout,P.: Učebnice jazyka C nakladatelství KOPP, České Budějovice, duben 2009, VI. vydání, ISBN 978-80-7232-383-8, 280 stran

Herout,P.: Učebnice jazyka Java nakladatelství KOPP, České Budějovice, září 2008, IV. upravené vydání, ISBN 978-80-7232-355-5, 381 stran

Doležel L.: Seriál Java Native Interface dostupné z http://www.abclinuxu.cz/serialy/java-native-interface

[4]HEROUT, Pavel. *Java a XML*. České Budějovice: nakladatelství KOPP, 2007. ISBN 978-80-7232-307-4.

[5]Profibus: ČVUT. [online]. [cit. 2013-03-28]. Dostupné z: http://www1.fs.cvut.cz/cz/U12110/site/profibus/

[6]Regulační pohony: Twin Line. [online]. [cit. 2013-03-28]. Dostupné z: http://www.regulacni-pohony.cz/data/el_twin.html

[7]OptoControl 2500: MicroEpsilon. [online]. [cit. 2013-03-28]. Dostupné z: http://www.micro-epsilon.com/download/manuals/man--optoCONTROL-2500--en.pdf

[8]Java Native Interface: wikipedia. [online]. [cit. 2013-03-28]. Dostupné z: http://cs.wikipedia.org/wiki/Java_Native_Interface

TwinLine Control Tool: manuál. [online]. [cit. 2013-03-28]. Dostupné z: http://www.global-download.schneider-electric.com/mainRepository/EDMS_CORP3.nsf/1a75da9e3565db2c852575bb004d391d/A5E9480AC709C53A8525774A005A4CE4/$File/tlct-gb.pdf

Programmming in C/C++ with the Java Native Interface. [online]. [cit. 2013-03-28]. Dostupné z: http://home.pacifier.com/~mmead/jni/cs510ajp/index.html#Example

Přehrávání a střih videozáznamů v jazyce JAVA: ZČU. [online]. [cit. 2013-03-28]. Dostupné z: http://www.kiv.zcu.cz/~lobaz/mhs/semestralky/2006/video_Java/

Java Media Framework. [online]. [cit. 2013-03-28]. Dostupné z: http://grack.com/downloads/school/enel619.10/report/java_media_framework.html

Sériová komunikace: RXTX. [online]. [cit. 2013-03-28]. Dostupné z: http://kishor15389.blogspot.cz/2012/06/install-javaxcommor-rxtx-in-eclipse.html

Diplomová práce ČVUT: Tran_Duy_Khanh. [online]. 2009 [cit. 2013-03-28]. Dostupné z: http://measure.feld.cvut.cz/cs/system/files/files/cs/vyuka/zaverecne_prace/DP_2009_Tran_Duy_Khanh_locked.pdf

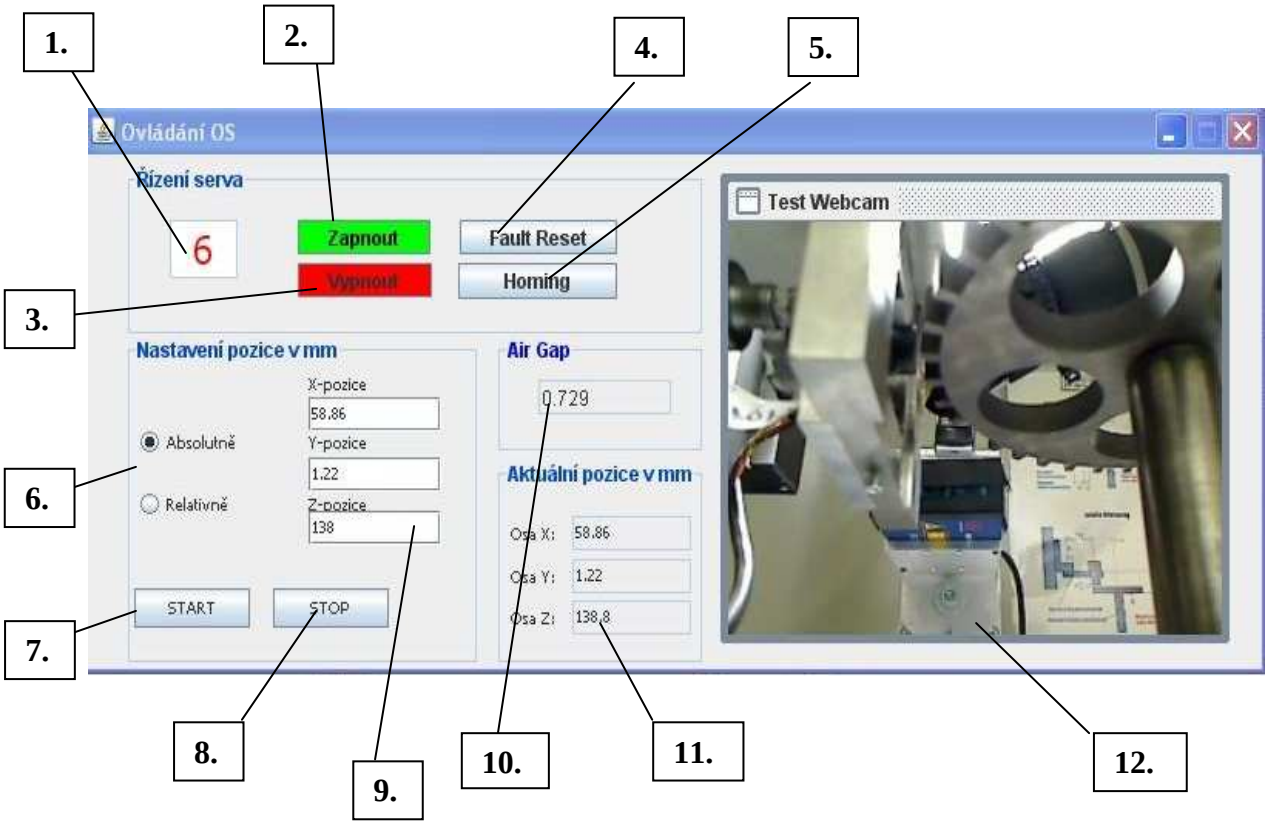
Manuál TLC 53x: BergerLahr. [online]. [cit. 2013-03-28]. Dostupné z: ftp://ftp.sdt.se/Documentation/BergerLahr/Manuals/Twinline/TLC5xx/TLC53x/English/TLC53x-gb.pdf

Manuál ProfiBus Network: Siemens. [online]. [cit. 2013-03-28]. Dostupné z: http://stest1.etnetera.cz/ad/current/content/data_files/automatizacni_systemy/prumyslova_komunikace/profibus/sysman_profibus-network-manual_2009-04_en.pdf

10. Přílohy

10.1 Příloha A- Manuál k použití aplikace

10.1.1 Popis uživatelského rozhraní



Obrázek 18- GUI programu s popisem funkcí

Jednotlivé módy řídících jednotek

1. **Zobrazuje aktuální mód řídící jednotky osového systému**

- **Mód 3 –** Jednotka je v pořádku a je připravena k zapnutí (po zapnutí hlavního vypínače na měřícím stavu)
- **Mód 4 –** Jednotka je zapnuta (po stisku aktivačního tlačítka na panelu)
- **Mód 6 –** Motory jsou odblokované a připravené k nastavení pozice osového systému (stisk tlačítka „Zapnout“ viz Obr. 1)
- **Mód 7 –** Osy se nacházejí mimo toleranční rozsah nebo byly zastaveny po stisku tlačítka „Stop“ v aplikaci viz Obr. 1. Pro opětovnou aktivaci je třeba provést vymazání chyby stiskem tlačítka „Fault Reset“ viz Obr. 1 a pokud bylo důvodem vyjetí osami mimo toleranční rozsah tak následuje stisk tlačítka „Homing“ viz Obr. 1.
- **Mód 9 –** Nastává při otevření ochranných dveří. Do módu 4 se opět dostaneme po stisku tlačítka „Fault Reset“ viz Obr. 1 a stisku aktivačního tlačítka na panelu.

2. **Zapnutí motorů os**
3. **Vypnutí motorů os**
4. **Reset chybového stavu (mód 7 a mód 9)**
5. **Inicializace osového systému**
6. **Volba mezi absolutním a relativním pozicováním**
7. **Zahájení posunu**
8. **Zastavení posunu**
9. **Vzdálenost v mm na které chceme aby se osy dostaly v případě volby absolutní pozicování v případě relativního o kolik se osy mají pohnout**
10. **Aktuální vzdálenost osy X od ozubeného kola**
11. **Aktuální poloha jednotlivých os**
12. **Obraz z webkamery, který ukazuje vzdálenost senzoru od ozubeného kola**

10.1.2 Popis postupu práce s programem

1) Po zapnutí programu, probíhá inicializace. Po úspěšné inicializaci se nám objeví informace, že hodnoty do textových polí se zadávají v mm a v případě potřeby pohybu do záporného směru osy je potřeba zadat před hodnotu znaménko „-“.

2) Po potvrzení tlačítkem „Ok“ se nám objeví hlavní okno aplikace.

3) Pomocí tlačítka „Zapnout“ přejdeme do **módu 6** a dojde k zobrazení aktuální pozice. Osový systém je připraven pro nastavení pozice měřeného senzoru vůči ozubenému kolu. Po zapnutí motorů provedeme kontrolu v aplikaci, je-li nastaven **mód 6**. Poté stiskem tlačítka „Homing“ provedeme inicializaci osového systému a jsme připraveni pro nastavení pozice. Do vstupních polí pro osy X,Y,Z zadáme požadovanou polohu v „mm“ a vybereme volbu absolutní nebo relativní polohování. Poté stiskem tlačítka „Start“ uvedeme osy do pohybu. V průběhu pohybování je možnost zastavení pohybu os tlačítkem „Stop“. Nastavenou polohu ověříme v aplikaci v panelu „Aktuální pozice mm“.

Po dokončení polohování vypneme motory tlačítkem „Vypnout“ a můžeme přejít k požadovanému měření.

Poznámka:
Hodnoty požadované pozice zadáváme na dvě desetinná místa „xxx.xx“ v milimetrech.Směr posunu nám určuje znaménko před zadaným číslem.Znaménkem „+/-“ xxx.xx určujeme směr pohybu (směr pohybu osy je vyznačen na každé z os).