Programová realizace řízení motorů MAXON

David Malaník

2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ABSTRAKT

Technologické a aplikační možnosti motorů MAXON jsou ověřeny i mimo zeměkouli. V rámci řešení STOČ bylo provedeno v návaznosti na řešení laboratorního modelu pro výuku technických prostředků vytvoření softwarové aplikace pro rozšíření firmware EPOS pro ovládání a sledování laboratorní úlohy s motorky. Aplikace byla napsána v jazyce c++ s využitím knihovny wxWidgets. Firmware EPOS je včleněn do aplikace úlohy a umožňuje z obrazovky PC zadávat parametry chodu motorků pro nastavený počet otočení nebo úhlového natočení, pro trvalý chod s nastavením otáček a pro spojený hnací a brzdicí režim s regulací PID. Významný pedagogický přínos má část řízení vzájemně spojených motorků (hnací a brzdicí) ve funkcích autotunning parametrů PID regulátoru, sledování kvality regulace podle parametrů z automatického nastavení a také podle vlastních konstant regulátoru.

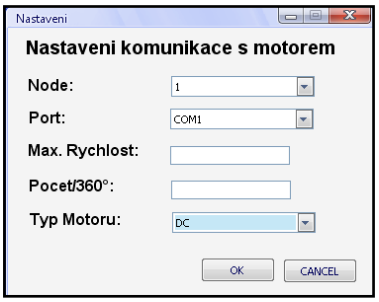
Výsledkem této práce je zde představený program. Jedná se o modulární systém, který je možné přizpůsobit podle individuálních potřeb výuky, případně vývoje. Lze tedy jednoduše přidávat funkce a procedury, které vyplynou z technických realizací navazující činnosti.

Klíčová slova: řízení motorů, motory MAXON, řídící jednotky EPOS, programování c++, knihovna wxWidgets, identifikace PID složek regulátorů, implementace dll knihoven.

OBSAH

1 DOKUMENTACE PROGRAMU

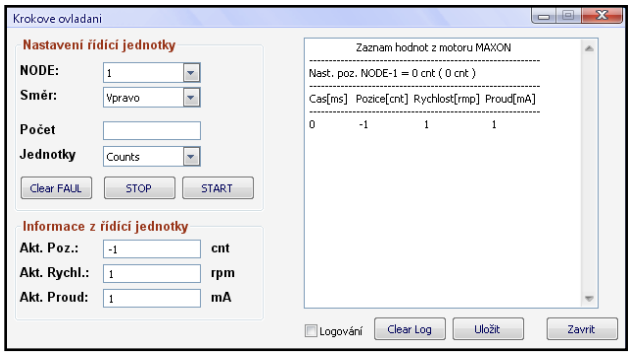
1.1 Menu Nastavení - Komunikace



Obrázek 1 – Nastavení komunikace

V dialogovém okně ***Komunikace*** je možnost nastavení komunikačního protokolu, jak znázorňuje Obrázek 1. Zde je možné vybrat sériový port na který je připojena hlavní(NODE-1) komunikační jednotka. Dále je nutné nastavit parametry motorku: Node(číslo uzlu, ke kterému je motorek připojen), maximální rychlost motorku(z manuálu výrobce), počet snímačů na pozční sondě motorku(z manuálu výrobce) a vybrat typ motorku. Po stisku ***OK*** dojde k uložení zadaných dat do **xml** souboru s názvem daného uzlu v adresáři ***Nastavení***.

1.2 Menu Ovládání - Krokové



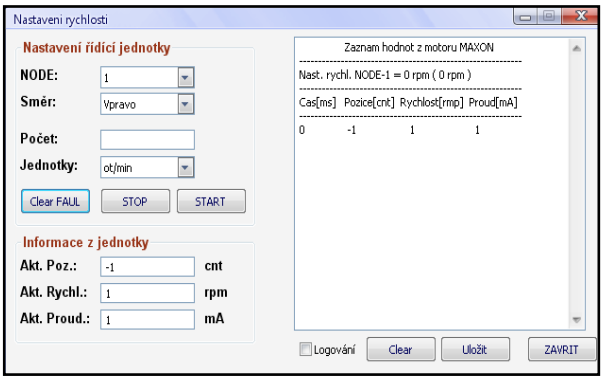
Obrázek 2 – Krokové ovládání

V této části programu je ovládání krokového(resp. pozičního) profilu motorků. Jak ukazuje Obrázek 2, je možno vybrat motorek(pomocí čísla NODE) se kterým se bude pracovat. Dále lze zvolit směr otáčení, počet jednotek a jednotky. Na výběr jsou následující jednotky: counts(počet impulsů se sondy na motorku např. 2048imp/360° dle sondy), Otáčky(počet otáček) a úhel. Úhlové jednotky jsou ale zatíženy chybou, protože při nich dochází k výpočtům s desetinnými čísly a řídící jednotka pracuje pouze s celými čísly. Při zadávání hodnot celočíselného typu je implementována kontrola, která při zadání neceločíselné hodnoty uživatele upozorní a neprovede požadovanou úlohu. Pokud je motorek z jakéhokoliv důvodu v chybovém stavu(signalizuje červené kontrolka na řídící jednotce), lze jej přes tlačítko ***Clear FAUL*** převédst opět do provozního režimu.

Pomocí tlačítka ***START*** se potom započne přesun na zvolenou pozici. S programem lze během přesunu pracovat(program není operací přesunu zastaven a lze tedy ovládat druhý motorek). Současně se spustí logování aktuálních dat do pravé části okna. Vybraný motorek lze kdykoliv zastavit tlačítkem ***STOP***. Pokud uživatel požaduje záznam hodnot z motorku před započetím přesunu je možné pomocí zaškrtávacího políčka ***logování*** zapnout záznam hodnot kdykoliv. Záznam lze také kdykoliv zastavit odškrtnutím tohoto zaškrtávacího tlačítka. Zaznamenané hodnoty lze vymazat stiskem tlačítka ***Clear Log***. Nakonec lze záznam uložit do **txt** souboru pomocí tlačítka ***Uložit***. Uživatel je potom vyzván k výběru umístění souboru a k volbě názvu.

POZOR PŘI ZAVŘENÍ DIALOGOVÉHO OKNA DOJDE K ZASTAVENÍ MOTORKŮ A K UKONČENÍ JEJICH OVLÁDÁNÍ!!

1.3 Menu Ovládání - Rychlostní



Obrázek 3 – Rychlostní ovládání

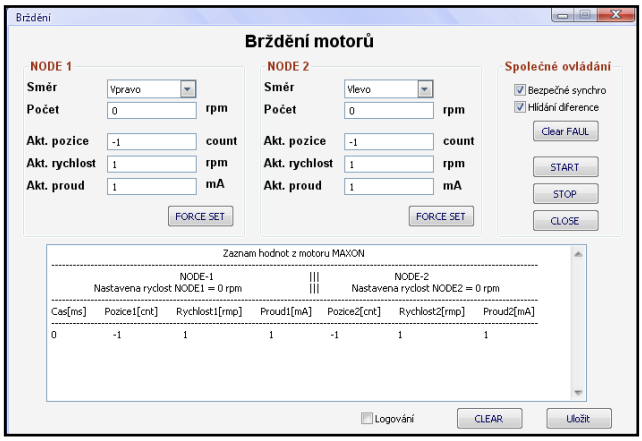
V této části programu je ovládání rychlostního profilu motorků. Jak ukazuje Obrázek 3, je možno vybrat motorek(pomocí čísla NODE) se kterým se bude pracovat. Dále lze zvolit směr otáčení, počet jednotek a jednotky. Na výběr jsou následující jednotky: ot/min(rpm) a ot/s(rps). Při zadávání hodnot celočíselného typu je implementována kontrola, která při zadání neceločíselné hodnoty uživatele upozorní a neprovede požadovanou úlohu. Kontrola je také implementována na omezení rychlosti motorku, pokud uživatel zadá vyšší rychlost než motorek zvládá je na to upozorněna a ke startu motorku nedojde. Pokud je motorek z jakéhokoliv důvodu v chybovém stavu(signalizuje červené kontrolka na řídící jednotce), lze jej přes tlačítko ***Clear FAUL*** převédst opět do provozního režimu.

Pomocí tlačítka ***START*** se potom započne rozběh na zvolenou rychlost. S programem lze během rozběhu pracovat(program není provozem motorku zastaven a lze tedy ovládat druhý motorek). Současně se spustí logování aktuálních dat do pravé části okna. Vybraný motorek lze kdykoliv zastavit tlačítkem ***STOP***.

Pokud uživatel požaduje záznam hodnot z motorku před započetím přesunu je možné pomocí zaškrtávacího políčka ***logování*** zapnout záznam hodnot kdykoliv. Záznam lze také kdykoliv zastavit odškrtnutím tohoto zaškrtávacího tlačítka. Zaznamenané hodnoty lze vymazat stiskem tlačítka ***Clear Log***. Nakonec lze záznam uložit do **txt** souboru pomocí tlačítka ***Uložit***. Uživatel je potom vyzván k výběru umístění souboru a k volbě názvu.

POZOR PŘI ZAVŘENÍ DIALOGOVÉHO OKNA DOJDE K ZASTAVENÍ MOTORKŮ A K UKONČENÍ JEJICH OVLÁDÁNÍ!!

1.4 Menu Ovládání - Brzdění



Obrázek 4 – Brzdění motorků

V této části programu je ovládání motorků ve vazbě. Jak ukazuje Obrázek 4, je možné vybrat směr otáčení každého motorku. O to aby se nemohly otáčet proti sobě se stará funkce ***Bezpečné synchro***, kterou lze ale vypnout nebo zapnout pomocí zaškrtávacího políčka ***Bezpečné synchro***. Dále lze zvolit rychlost otáčení každého motorku. Program pomocí vnitřní funkce hlídá aby uživatel zadal celé číslo a aby největší zadaná rychlost byla menší než maximální rychlost pomalejšího motorku. Pokud nebude výše uvedená podmínka splněna, bude na to uživatel upozorněn a motorky se nenastartují. Program dále pomocí funkce ***Hlídání diference*** počítá diferenci mezi oběma rychlostmi a pokud je diference vyšší než 20% je uživatel opět upozorněna a ke startu nedojde. Tuto funkci lze vypnout nebo zapnout pomocí zaškrtávacího políčka ***Hlídání diference.***

Pokud je některý motorek z jakéhokoliv důvodu v chybovém stavu(signalizuje červené kontrolka na řídící jednotce), lze oba motorky přes tlačítko ***Clear FAUL*** převédst opět do provozního režimu.

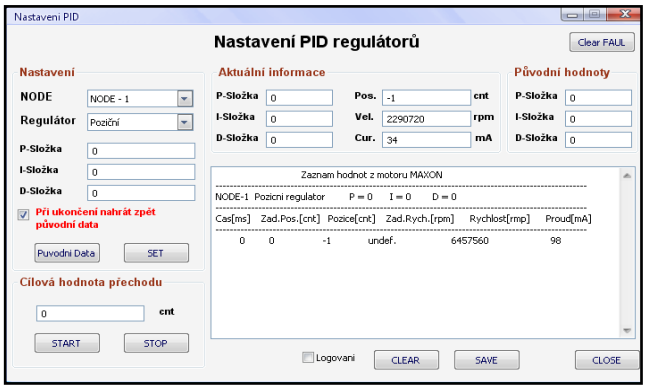
Pro potřeby simulace poruch motorkem ve vazbě je možné za provozu nastavit hodnotu rychlosti otáčení pro kterýkoliv motorek bez nutnosti zabrzdění

motorků. To je možné provést nastavením rychlosti do patřičného políčka a stisknutím tlačítka ***FORCE SET*** u odpovídajícího motorku.

Pomocí tlačítka ***START*** se potom započne rozběh motorků na zvolené rychlosti. S programem lze během přesunu pracovat(program není provozem motorků). Současně se spustí logování aktuálních dat do pravé části okna. Motorky lze kdykoliv zastavit tlačítkem ***STOP***. Pokud uživatel požaduje záznam hodnot z motorků před startem, je možné pomocí zaškrtávacího políčka ***logování*** zapnout záznam hodnot kdykoliv. Záznam lze také kdykoliv zastavit odškrtnutím tohoto zaškrtávacího tlačítka. Získané hodnoty lze vymazat stiskem tlačítka ***Clear Log***. Nakonec lze záznam uložit do **txt** souboru pomocí tlačítka ***Uložit***. Uživatel je potom vyzván k výběru umístění souboru a k volbě názvu.

POZOR PŘI ZAVŘENÍ DIALOGOVÉHO OKNA DOJDE K ZASTAVENÍ MOTORKŮ A K UKONČENÍ JEJICH OVLÁDÁNÍ!!

1.5 Menu Ovládání - PID



Obrázek 5 – Nastavení PID

V této části programu je možné nastavovat PID(resp. PI) složky jednotlivých regulátorů(pozičního a rychlostního) na řídící jednotce **EPOS,** jak ukazuje Obrázek 5. V sekci nastavení je možné vybrat jednotlivý motorek(pomocí jeho NODE) a regulátor, který se bude rekonfigurovat. Dále je možné navolit vlastní složky PID(resp. PI dle typu regulátoru). Složky jsou opět celočíselné a program upozorňuje na zadání neceločíselné hodnoty. Hodnoty lze poté pomocí tlačítka ***SET*** nastavit do paměti řídící jednotky. Pomocí tlačítka ***Puvodni Data*** je možné nahrát původní hodnoty. Aktuální informace jsou zobrazovány v kontejneru ***Aktuální informace***. V kontejneru ***Původní hodnoty*** jsou zobrazeny hodnoty složek regulátorů, které byly načteny při startu dialogového okna. Pokud by se motorky dostaly do faul stavu(indikuje ho červená kontrolka na

řídící jednotce), je možné je převést zpět do provozního režimu stiskem tlačítka ***Clear FAUL***.

Pomocí zaškrtávacího políčka je možné zrušit bezpečnostní funkci, která při zavření okna automaticky nahraje původní hodnoty do jednotek.

Dále je možnost nastavit cílovou hodnotu přechodu, která má jednotky buď cnt(při pozičním regulátoru), nebo rpm(při rychlostním regulátoru).

Pomocí tlačítka ***START*** se potom započne rozběh motorku na zvolenou cílovou hodnotu. S programem lze během přesunu pracovat(program není provozem motorků). Současně se spustí logování aktuálních dat do pravé části okna. Při zvoleném pozičním regulátoru se v poli záznamu ***Zad. Rychlost*** zobrazuje řetězec ***undef.*** Motorek lze kdykoliv zastavit tlačítkem ***STOP***. Pokud uživatel požaduje záznam hodnot z motorků před startem, je možné pomocí zaškrtávacího políčka ***logování*** zapnout záznam hodnot kdykoliv. Záznam lze také kdykoliv zastavit odškrtnutím tohoto zaškrtávacího tlačítka. Získané hodnoty lze vymazat stiskem tlačítka ***Clear Log***. Nakonec lze záznam uložit do **txt** souboru pomocí tlačítka ***SAVE***. Uživatel je potom vyzván k výběru umístění souboru a k volbě názvu.

POZOR PŘI ZAVŘENÍ DIALOGOVÉHO OKNA DOJDE K ZASTAVENÍ MOTORKŮ A K UKONČENÍ JEJICH OVLÁDÁNÍ!!

2 REALIZACE PROGRAMU

Samotný program byl realizován v jazyce c++, jako vývojové prostředí byl použit program wxDevcpp verze 6.10.2. Aby byla zajištěna jednoduchá portace programu na platformy UNIX/LINUX, MAC byla použita knihovna wxWidgets, která v podstatě zajistí stejný vzhled aplikace na výše uvedených platformách.

Celé jádro programu tvoří pouzdro kolem vytvořené komunikační knihovny a je snadno modifikovatelné v případě požadavku na nové funkce programu. Samotné jádro vytváří hlavní okno aplikace a nabídku daného okna. Jednotlivé prvky nabídky jsou tvořeny samostatnými dialogovými okny a jsou schopné plně autonomní činnosti. Aplikace umožňuje jednoduché přidávání dalších modulů dle potřeb výuky, případně výzkumu uplatnění motorů MAXON.

Samotná zjednodušená komunikační knihovna zapouzdřuje v podstatě celou řídící jednotku, včetně nastavení komunikačního portu, informací o motoru a slouží k nastavování jednotlivých motorů. Knihovna je vždy přilinkována ke každému dialogovému oknu a své konfigurační údaje o daném motorku si vybírá z konfiguračních **xml** souborů. Tyto soubory jsou tedy k chodu programu naprosto nezbytné.

Každé dialogové okno nejprve načte data z konfiguračních souborů. Poté se zapne timer, který obnovuje každé okno s periodou 200ms. Periodu je možné změnit před kompilací programu u každého okna zvlášť. Jediným omezením periody je komunikace po sériovém portu, při příliš krátké periodě(menší než odezva rozhraní) dochází k nevykonání požadavků na řídící jednotku a mohou nastat hazardní stavy. Timer se stará o překreslování aktuálních údajů a v záznamech působí jako časová osa na kterou jsou data navázána. Uvnitř každého okna je vytvořena obsluha všech jeho ovládacích a vykreslovacích prvků. Tím je zaručena nezávislost jednotlivých oken na hlavním. Pokud by se tedy na vývoji podílelo více programátorů, není problém aby každý vytvářel několik dialogových oken pro ovládání vybraných činností bez nebezpečí kolize vzniklých např. použitím globálních proměnných.

Ačkoliv jsou dialogové okna ve své podstatě naprosto autonomní, je možná návaznost činností, které umožňují. Příkladem je návaznost dialogových oken obsluhujících nastavení PID složek regulátorů a brzdění motorků ve vazbě. V jednom okně je možné nastavit PID hodnoty daného regulátoru a pomocí druhého modulu je

možné zaznamenat chování jak při náběhu na žádanou hodnotu tak i při simulované poruše. Poruchu, lze vytvořit tak, že se za provozu provede změna otáček na hnaném motorku. Tím dojde k simulované poruše a lze zaznamenat chování sledovaných hodnot.

Činnost jedné řídící jednotky není(do jisté míry) ovlivněna probíhající komunikaci s druhou jednotkou. Lze tedy nastavit zvolený režim na jedné jednotce a ve stejném dialogovém okně v téměř stejném čase nastavovat režimy druhé jednotky.

Pro operace s **xml** soubory byla vytvořena samostatná třída, která zapouzdřuje funkcionalitu nad těmito datovými typy. Je vytvořena přímo na míru vytvářenému řešení.

Pro potřeby uchování zachycených údajů nebyl zvolen formát **xml** kvůli importu dat do programu **Excel** a také kvůli přehlednosti zachycených dat při jejich prohlížení např. v poznámkovém bloku. Pro tuto činnost byl zvolen formát **txt.** Jako oddělovače jednotlivých hodnot byly použity tabulátory. To usnadňuje import zachycených dat např. do programu **Excel** a následnou vizualizaci vývoje žádané hodnoty např. pomocí grafu. Z podstaty použitého datového typu(wxString) vyplývá v této realizaci jedno omezení. Nelze zaznamenat údaje jejichž velikost bude větší než 4GB(přibližně 4 000 000 000 znaků). Pokud by ale tato situace přece jen nastala, je program koncipován tak, aby šel velice jednoduše upravit. Nicméně hodnota 4GB je v současné době vysoce naddimenzována a nepředpokládá se její překonání při práci s tímto programem.

ZÁVĚR

Cílem této práce bylo vytvořit náhradu dodávaného ovladače motorů MAXON. K řízení motorů slouží řídící jednotky EPOS. Výrobce dodává ke svým řídícím jednotkám komunikační knihovnu pro vytváření vlastních aplikací. Proto je možné vytvářet realizace ovládání motorků na míru danému použití.

Dodávaný obslužný software byl zhodnocen jako zbytečně komplikovaný na ovládání a pro potřeby výuky(kdy na něm mají pracovat hlavně studenti) jen těžko využitelný. Proto bylo přistoupeno k realizaci vlastního vývoje, který by sloužil pro výukové účely a případně pro vědecké účely na Univerzitě Tomáše Bati ve Zlíně. Výsledkem měl zajistit lehce škálovatelný program, který by nahradil prostředí dodávané výrobcem. Velký důraz byl kladen právě na jednoduchost programu a na jeho flexibilitu podle aktuálních potřeb. Pro vývoj byl zvolen rozšířený programovací jazyk c++. Aby bylo možné používat program napříč platformami, je napsán v univerzální knihovně wxWidgets.

Výsledkem tohoto projektu je program, který je velmi jednoduchý a názorný na obsluhu. Je tedy ideální pro nasazení při výuce(neklade nároky na vysokou odbornost uživatele). Navíc je velmi lehce přizpůsobitelný jakýmkoliv potřebám, které vyplynou při jeho používání(ovšem v rámci možností řídících jednotek).

Od dalšího semestru se zvažuje využití programu a modelu s motorky k výuce na fakultě aplikované informatiky. Měl by sloužit k získání praktických poznatků z teorie automatického řízení a technických prostředků automatizace. V budoucnu by se měl použít jako testovací modul pro návrh solárního panelu, který by se pomocí programu natáčel za sluncem.

Další možností využití poznatků získaných vývojem tohoto řešení by byl vývoj vlastního ovladače řídících jednotek(a potažmo i motorů) pro prostředí Control Web. Tímto postupem by se výrazně ulehčila vizualizace získaných dat a přibyla by i možnost ovládání jednotek přes www rozhraní realizovaným prostředím Control Web.