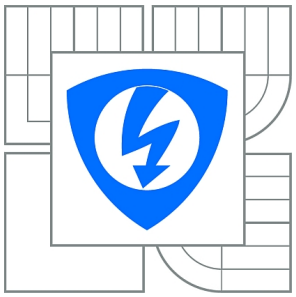
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMUNIKACE MEZI PLC B&R A PC

COMMUNICATION BETWEEN B&R PLC AND PC

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

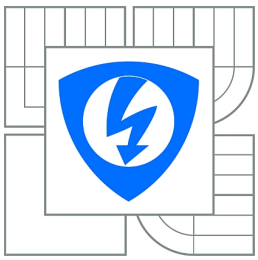
AUTOR PRÁCE — PAVOL ŠLICHTA

AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR BURLAK

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Pavol Šlichta	***ID:***	146111
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Komunikace mezi PLC B&R a PC

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření knihovny, která umožní výměnu dat mezi PLC a uživatelskou aplikací vytvořenou v prostředí NI LabVIEW nebo MATLAB/Simulink. Zadání lze shrnout do následujících bodů:
1. Seznamte se s programovatelnými automaty firmy B&R a s možnostmi komunikace mezi PLC a PC.
2. Navrhněte a realizujte univerzální knihovnu určenou pro NI LabVIEW a MATLAB/Simulink zprostředkovávající výměnu hodnot proměnných mezi PC a PLC aplikacemi. Dále se zaměřte na detekci nedodržení vzorkovací periody.
3. Vytvořené funkce vhodně otestujte při regulaci fyzikálních soustav

DOPORUČENÁ LITERATURA:

Vlach, J. a kol.: Začínáme s LabVIEW, Praha, ISBN 978-80-7300-245-9
B&R www.br-automation.com

Termín zadání:	10.2.2014	***Termín odevzdání:***	26.5.2014

Vedoucí práce: Ing. Vladimír Burlak
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cieľom tejto bakalárskej práce je naštudovanie softvérovej podpory firmy B&R a na jej základe vytvorenie funkcií, ktoré budú zapisovať/čítať dáta z/do PLC. V úvodnej časti sú uvedené základné možnosti komunikácie z PLC. Dôležitou časťou práce je vysvetlenie PVI objektovej hierarchie, ktorá je nutná pre pripojenie klienta. Hlavným výsledkom bakalárskej práce sú implementované funkcie v Simulinku/NI LabVIEW. V závere práce je demonštrovaná funkčnosť riešenia ukázaná na regulácii fyzikálnej sústavy.

Kľúčové slová

PLC, B&R, Simulink, NI LabVIEW, Komunikácia, PVI Manager, PSD

Abstract

Objective of this thesis is to examine the software support for company B&R and creation of functions that handle input and output for the PLC. At first we look at basic communication techniques for the PLC. Important part of the thesis describes PVI which is esential for connecting the client. The main goal of the thesis is to implement these functions in Simulink/NI LabView. Functionality of this solution is showcased in the example at the end of the thesis.

Keywords

PLC, B&R, Simulink, NI LabVIEW, Communication, PVI Manager, PSD

Bibliografická citace:

ŠLICHTA, P. *Komunikace mezi PLC B&R a PC*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 85s. Vedoucí bakalářské práce byl Ing. Vladimír Burlak.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Komunikace mezi PLC B&R a PC 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. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: ……

podpis autora

Poďakovanie

Ďakujem vedúcemu bakalárskej práce Ing. Vladimírovi Burlakovi za účinnú metodickú, pedagogickú, odbornú pomoc a ďalšie cenné rady pri spracovávaní mojej bakalárskej práce.

V Brně dne: ……

podpis autora

Obsah

Zoznam obrázkov:

Zoznam tabuliek:

1 ÚVOD

V úvode tejto práci bude čitateľ zoznámený s komunikáciou medzi PC a riadiacimi systémami od firmy B&R.

Na komunikáciu sa využíva komunikačné rozhranie Automation Net PVI. Rozhranie PVICOM sprostredkováva komunikáciu medzi užívateľskou aplikáciou MS Windows a PVI Managerom. Ďalej bude bližší popis možností vytvorenia komunikácie a bude vybraná najlepšia možnosť vzhľadom na požiadavky zadania práce.

Výsledkom práce je vytvorenie programov, pričom prvý bude vytvorený pre Simulink a druhý pre LabVIEW. Programovací jazyk v ktorom budem programovať je C++. Tento výber som si zvolil z dôvodu podpory tohto jazyka funkciami PVICOM.

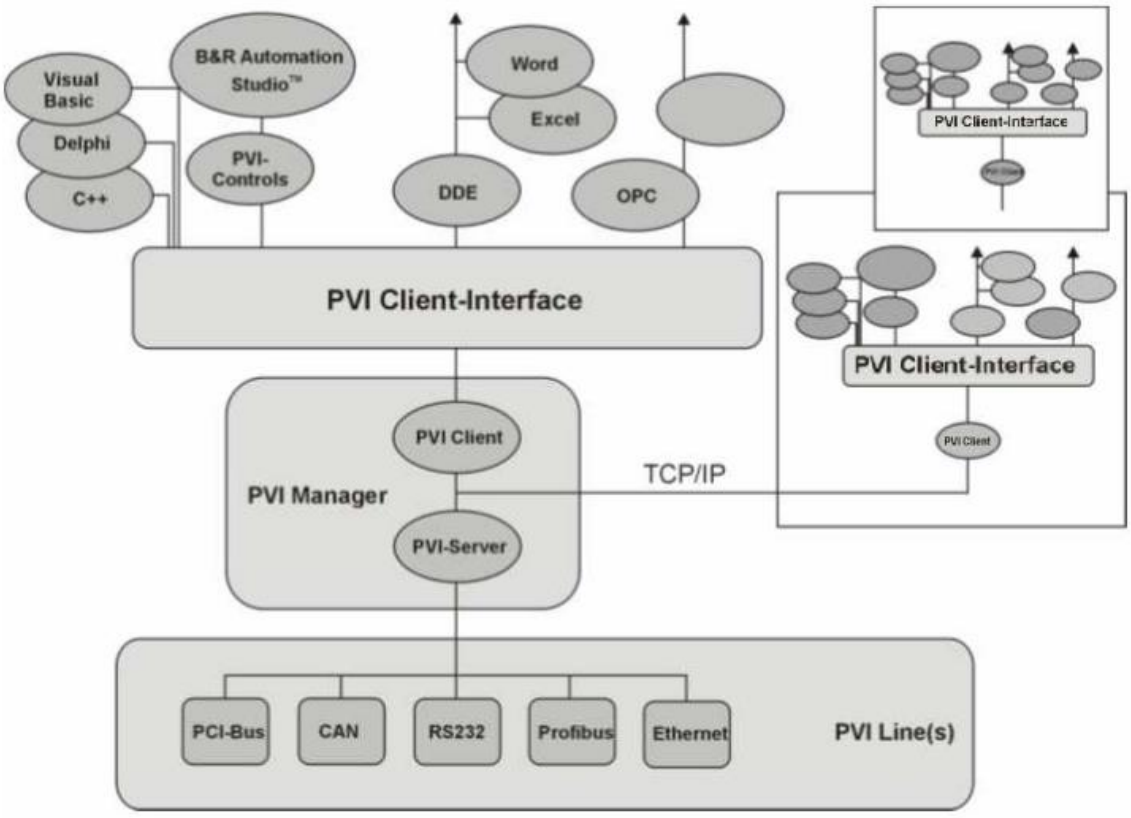
V programe pre Simulink budú vytvorené funkcie pre zápis/čítanie premennej a zaistenie behu simulácie v reálnom čase. Beh simulácie v reálnom čase bode riadený čítaním premennej z PLC. Pre pripojenie sa k premennej bude vytvorená užívateľsky prístupná maska, do ktorej sa budú zadávať reťazce potrebné pre pripojenie sa k premennej. Pre ľahšiu implementácie vytvorených funkcií bude vytvorený *m-file* ktorý implementuje vytvorené funkcie do prostredia Simulink. Vytvorené funkcie budú otestované na riadení fyzikálneho modelu.

V programe pre LabVIEW budú vytvorené knižnice pre 32 bitovú verziu systému a pre 64 bitovú verziu systému. Beh simulácie v reálnom čase bude riadený v PC.

2 KOMUNIKÁCIA MEDZI PC A RIADIACIM SYSTÉMOM

2.1 Štruktúra Automation Net PVI

Pre komunikáciu medzi ľubovoľnými riadiacimi systémami od firmy B&R (napr. IPC, PLC) a užívateľskou aplikáciou, ktorá je určená pre platformu MS Windows, poskytuje výrobca riadiacich systémov jednotné komunikačné rozhranie Automation Net PVI (Process Visualization Interface). Toto rozhranie budeme ďalej označovať len ako PVI. PVI poskytuje širokú škálu funkcií, od jednoduchého prenosu hodnôt premenných medzi užívateľskou aplikáciou (napr. vizualizáciou) a riadiacim systémom, až po nahranie celého projektu do riadiaceho systému a prostredníctvom PVI Service aj sledovanie diagnostických informácií. Pre dosiahnutie maximálnej flexibility bola štruktúra PVI rozdelená na tri základné časti, ktorými sú PVI Manager, PVI Lines a PVI Client (užívateľská aplikácia, ktorá komunikuje prostredníctvom rozhrania PVICOM). Uvedená štruktúra je zobrazená na obrázku 2.1. [1]



Obrázok 2.1: Štruktúra komunikačného rozhrania PVI [2]

2.1.1 PVI Manager

PVI manažér je hlavnou zložkou PVI a zodpovedá za riadenie všetkých typov procesov od najjednoduchších až k zložitým. Základný objekt PVI je automaticky riadený PVI Manažérom. Tento objekt je vždy prítomný počas behu PVI Manageru. Aplikácia nemusí nastavovať alebo odstraňovať tento objekt. Názov základného objektu je "Pvi". PVI Manager taktiež zodpovedá za správny tok a smer dát. Stará sa o správne načasovanie posielania a prijímania dát z aplikácie do PLC a naspäť. Zvláštna pozornosť je venovaná asynchrónnemu riadeniu, aby bolo možné koordinovať oneskorené dáta popri spracovávaní iných úloh. [1] [3]

Niektoré vlastnosti PVI Managera:

- objektovo orientované hierarchické riadenie všetkých procesných dát
- riadenie procesných dát založené na načasovaní a smere
- konverzia dát, linearizácia, hysterézia
- dynamický popis zapojení pre objekty PVI

2.1.2 PVI Monitor

Slúži pre zobrazenie prevádzkových vlastností a používa sa ku konfigurácii diagnostických funkcií a globálnych vlastností pre PVI Manager. Komunikácia medzi PVI Monitorom a PVI Managerom sa môže vykonávať buď lokálne (Monitor a Manager bežia na rovnakom počítači), alebo vzdialene (Monitor a Manager nebežia na rovnakom počítači). Pri používaní diaľkového typu komunikácie je PVI Monitor pripojený k správcovskému zariadeniu pomocou TCP/IP protokolu. [1]

2.1.3 PVICOM

Ako už bolo vyššie spomenuté, PVICOM predstavuje rozhranie pre komunikáciu medzi užívateľskou aplikáciou pre MS Windows a PVI Managerom. Metódy, ktoré tvoria uvedené rozhranie, sú sústredené v DLL (Dynamic Link Library) „PviCom.dll“ (pre 32 bitovú verziu Windows) a „PviCom64.dll“ (pre 64 bitovú verziu Windows). Toto rozhranie sprostredkováva komunikáciu typu klient/server na najnižšej úrovni a preto umožňuje rýchlu výmenu dát, nevyhnutnú pre túto prácu. Rovnako ako pri PVI Monitore aj PVI Client aplikácie, ktoré využívajú PVICOM rozhranie, môžu komunikovať nasledujúcimi spôsobmi:

Miestna komunikácia - PVI Manager a PVICOM aplikácia sú umiestnené na rovnakom počítači. Výmena dát prebieha pomocou zdieľanej časti pamäti. [3]

Vzdialená komunikácia medzi PVI Manager a PVICOM aplikáciou môžu byť umiestnené na samostatných počítačoch. Výmena dát prebieha cez TCP/IP sieť. [3]

2.1.4 PVI Line

Základnou úlohou PVI Line (linkovanie) je prepojenie PVI objektov (objektov služby) s objektmi mimo PVI. PVI Line je tiež zodpovedná za komunikáciu s PLC a určuje komunikačný protokol, ktorý sa bude používať pre komunikáciu. Protokoly, ktoré môžeme využiť pre komunikáciu sú:

- INA2000 (Industrial Network Architecture System 2000)
- SNMP (Simple Network Management Protocol)
- Modbus
- NET2000
- CANdirect
- MININET

Komunikáciu po ethernete podporujú viaceré protokoly. Nachádzajú sa tu protokoly SNMP a Modbus, ktoré podporujú iba ethernet. Ďalej je tu protokol INA2000, ktorý okrem ethernetu podporuje aj iné rozhrania (ako sú napr. RS232, RS422, CAN, Profibus FDL). [3]

Pre komunikáciu som vybral komunikačný protokol INA2000. Tento protokol je z môjho pohľadu najuniverzálnejší, pretože podporuje komunikáciu s viacerými rozhraniami. [3]

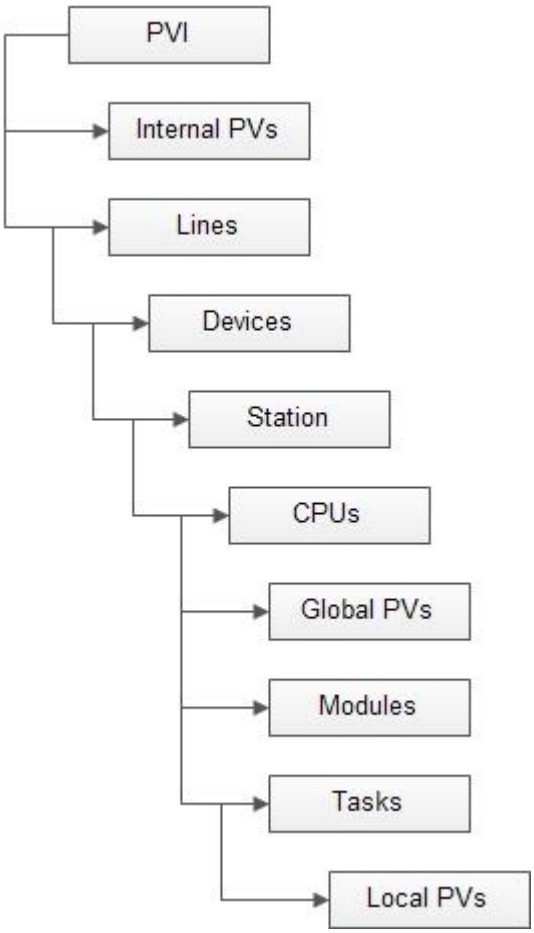
2.2 Rozhranie PVICOM

2.2.1 Pripojenie PVI Client aplikácie k PVI Managerovi

Pre komunikáciu s PVI Managerom je dôležité najprv vytvoriť a nastaviť správnu komunikačnú inštanciu. K vytvoreniu komunikácie sa používajú dve funkcie PviInitialize a PviXInitialize. Na rozdiel od funkcie PviInitialize, môžeme funkciu PviXInitialize volať viackrát, pričom každé nové volanie tejto funkcie vytvorí novú komunikačnú inštanciu. To umožňuje aplikácii PVICOM, aby mohla komunikovať s niekoľkými PVI Manažérmi na rôznych počítačoch. Ak je komunikačná inštancia nastavená ako PviXInitialize, tak všetky používané funkcie PVICOM musia byť volané ako PviX. Pri použití vyššie spomenutých funkcií sa používajú funkcie na deinicializáciu zavedených inštancií. Tieto funkcie sú PviDeinitialize a PviXDeinitialize. [2]

Funkcia PviInitialize alebo PviXInitialize inicializuje PVICOM rozhranie, zavádza PVICOM komunikačné inštancie a registruje komunikačnú inštanciu (klienta) s PVI Managerom (servera). Hneď ako sa zaregistrujete, je otvorené komunikačné

spojenie medzi klientom a serverom. Treba pripomenúť, že registrácia klienta je iba zahájená, preto návrat z funkcie nezaručuje stabilnú komunikáciu klient/server. [2]
PVICOM komunikačná inštancia môže signalizovať aktuálny stav komunikačného spojenia pomocou globálnych udalostí z aplikácie. K tomu však potrebujeme použiť globálne udalosti, ktoré aktivujeme funkciami PviSetGlobEventMsg alebo PviXSetGlobEventMsg. [2]



Obrázok 2.2: PVI objektová hierarchia [2]

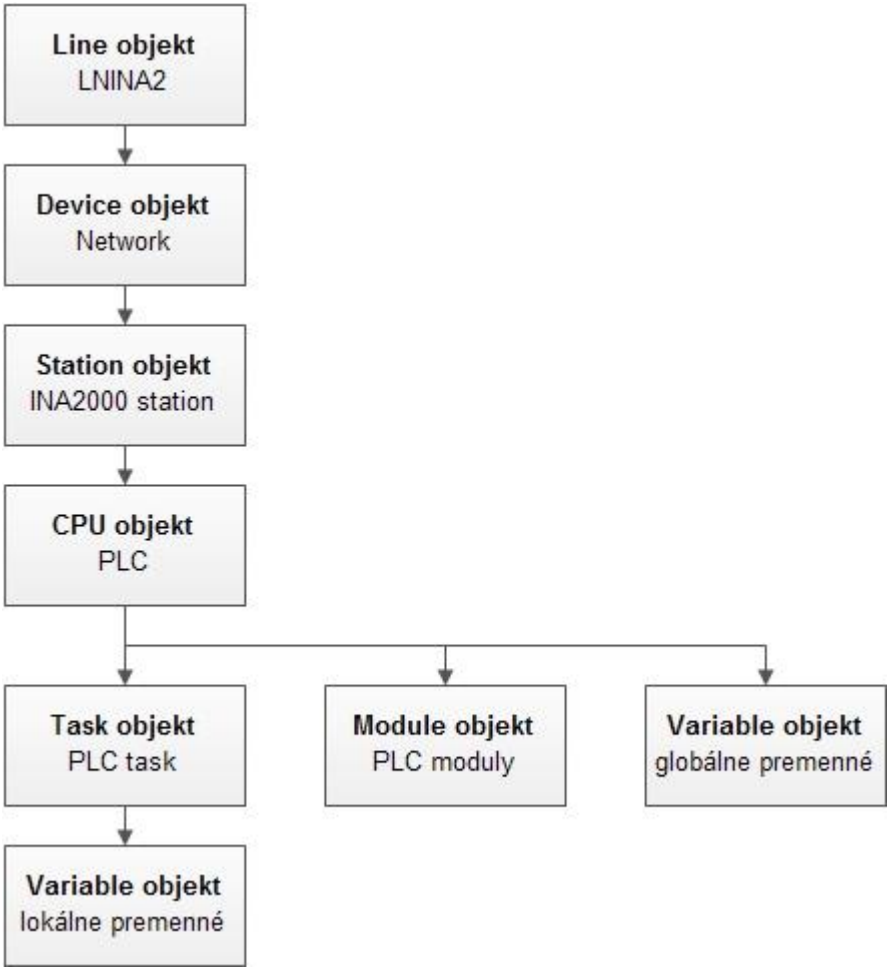
Procesné objekty sú rozdelené do rôznych typov objektov. Každý typ objektu reprezentuje určitú logickú alebo fyzickú súčasť komunikačného pripojenia alebo objektu v linke alebo na PLC. Typy procesných objektov sú: base objekt, line objekt, device objekt, station objekt, CPU objekt, module objekt, task objekt a variable objekt. Z týchto objektov sa skladá objektová štruktúra a pomocou nich sa vytvára komunikácia. Každý druh objektu má v stromovej štruktúre svoju úlohu priradenú podľa umiestnenia. Base objekt je na najvyššej úrovni stromovej štruktúry. Tento objekt je vytváraný implicitne. Line objekt definuje použitie PVI linky. Objekty typu device,

station a CPU sa používajú k vytvoreniu komunikácie s PLC a k adresovaniu PLC v sieti. Objekty modulu, task a variable sú súhlasné objekty s objektmi v PLC. Task označuje úlohu alebo proces a variable označuje premennú v PLC. [1]

Pri komunikácii medzi PC a PLC sa využíva knižnica PviCom64.lib pre 64 bitovú verziu systému alebo PviCom.lib pre 32 bitovú verziu systému. Hlavnou úlohou PVI Managera je najprv nastaviť komunikačnú žiadosť. Komunikačnou žiadosťou nastavíme spojenie medzi PVICOM a PVI Managerom. Toto spojenie vytvoríme volaním funkcie PviXInitialize(). Touto funkciou nastavujeme spôsob komunikácie (miestna alebo vzdialená) a časový limit (timeout) pre komunikáciu v sekundách. Každé zavolanie funkcie PviXInitialize() vytvorí nové komunikačné pripojenie k PVICOM a umožní tak komunikovať s viacerými manažérmi na rôznych PC . Ak pripojenie prebehne správne, tak komunikačnú žiadosť môžeme deinicializovať pomocou funkcie PviXDeinitialize(). PVICOM rozhranie pracuje na princípe klient/server, pričom PVI Manager predstavuje server a PVICOM aplikácia predstavuje klienta. Ak sa server a klient nachádzajú na jednom zariadení, tak sa jedná o lokálnu komunikáciu. Keď sa server a klient nenachádzajú na rovnakom zariadení, tak sa jedná o vzdialenú komunikáciu. Pri miestnej komunikácii sa používa na výmenu údajov lokálna zdieľaná pamäť. Ak používame vzdialený prístup, tak sa využíva komunikačný protokol TCP/IP. PVICOM rozhranie dokáže aplikácii nastaviť niekoľko komunikačných inštancií (žiadostí), čo umožňuje výmenu dát medzi niekoľkými PVI Mnagermi v rovnakom čase. Typ komunikácie určujeme podľa čísla portu a IP adresy. Simulácia má číslo portu 11160 a IP adresu 127.0.0.1, pričom je táto komunikácia lokálna. Vzdialená komunikácia prebieha vtedy, ak sa IP adresa zhoduje z IP adresou PLC a číslo portu je 11159. Vzdialená komunikácia klient/server prebieha pomocou komunikačného protokolu TCP/IP.

2.3 Linka INA2000 pre komunikáciu s PLC

Ako už bolo hore uvedené, komunikácia medzi PC a PLC bude prebiehať po ethernete, preto je výhodné použiť protokol INA2000 (Industrial Network Architecture). Tento protokol podporuje všetky zbernice, ktorými sú vybavené riadiace systémy B&R okrem RS485. Ďalšiu výhodou protokolu predstavuje bezplatná licencia pre jeho používanie. [1]



Obrázok 2.3: Usporiadanie procesných objektov INA2000 [1]

3 NÁVRH RIEŠENIA

3.1 Koncepcia riešenia

V zadaní je požiadavka vytvoriť programové prostriedky, ktoré by umožnili prenášať hodnoty premenných medzi aplikáciou vytvorenou v prostredí Matlab Simulink/NI LabVieW a procesmi, ktoré sú spustené v riadiacich systémoch B&R, pripojených k PC prostredníctvom siete ethernet.

Funkcie PVICOM podporujú niekoľko programovacích jazykov, ktorými sú C/C++ a Visual Basic.

Pre programovanie som si vybral programovací jazyk C++. Tento jazyk som si vybral aj preto, pretože sa dá jednoducho použiť pre vytvorenie DLL a následne zavolať z prostredia Matlab Simulink/NI LabVieW.

Pre zaistenie behu funkcií v reálnom čase bude potrebné riadiť periódu vzorkovania. Táto perióda vzorkovania bude v Simulinku prebiehať z PLC a v LabVIEW z PC (z časových dôvodov). Čítanie/zápis prebiehajú cyklicky. Každá funkcia čítanie/zápis bude mať svoju inicializáciu a deinicializáciu, kvôli tomu bude potrebné v Simulinku vytvoril bloky perióda, čítanie a zápis, ktoré budú obsahovať vlastnú inicializáciu, deinicializáciu a cyklicky sa opakujúcu operáciu.

Pre LabVIEW bude vhodnejšie oddeliť inicializáciu, deinicializáciu a vlastnú cyklicky sa opakujúcu operáciu.

V Simulinku vytvorím tri bloky. Blok pre zápis premennej, čítanie premennej, a blok pre kontrolu reálneho času v simulácii.

V NI LabVIEW sa budú volať funkcie čítanie a zápis, pričom perióda vzorkovania sa tam bude realizovať pomocou bloku *wait for* umiestnenom v cyklickej slučke *while*.

3.2 Synchrónny a asynchrónny režim

Synchrónny režim funguje tak, že pošle požiadavku a následne čaká na odpoveď. Pri tomto režime sa nedá posielať viacej požiadaviek behom spracovávania tých minulých, čo je veľká nevýhoda. Použitie tohto režimu v PVICOM aplikácii je jednoduchšie ako pri použití asynchrónneho režimu, pretože sa aplikácia nemusí starať o priraďovanie požiadaviek alebo odpovedí užívateľa. [3]

Asynchrónny režim funguje tak, že behom spracovávania požiadavky je možné poslať ďalšiu požiadavku a popri tom vykonávať iné úlohy. Asynchrónny režim v porovnaní so synchrónnym režimom je doplnený o funkcie, ktoré obsahujú slovo Response (odpoveď) alebo Request (požiadavka) napr. PviResponseWrite a PviRequestWrite. Na rozdiel od synchrónneho režimu sa tento režim môže používať v slučke. V aplikácii sa dá použiť asynchrónny aj synchrónny režim súčasne. [3]

Pre riešenie tejto práce som si zvolil synchrónny režim z dôvodu postupného (synchrónneho) volania blokov v Simulinku.

3.3 Použité funkcie

PviXInitialize:

Pre komunikáciu medzi PC a PLC budem využívať funkciu PviXInitialize(). Túto funkciu budem využívať preto, lebo umožňuje vytvorenie niekoľkých komunikačných inštancií jej opätovným volaním. Znamená to, že budeme môcť komunikovať s viacerými premennými a PLC súčasne. [1]

PviXDeinitialize:

Funkcia PviXDeinitialize slúži k ukončeniu komunikačného spojenia medzi PVI Managerom a PVICOM komunikačnej inštancie. Chyba sa najčastejšie objavuje pri deinicializácii nesprávneho identifikátoru (handlu). Iná návratová hodnota ako 0 signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. [1]

PviXWrite:

Táto funkcia slúži na zápis hodnoty z PC do premennej v PLC. Ak návratová hodnota tejto funkcie je iná ako 0, tak tým signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. Chyba je hlásená vtedy, ak dôjde k chybe pri vykonávaní požiadavky na zápis alebo v prípade, že spojenie medzi komunikačnou inštanciou a PVI Mnagerom je prerušené (timeout komunikácia) alebo ak dôjde k chybe aplikácie. Iná návratová hodnota ako 0 signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. [1]

PviXRead:

Táto funkcia odošle požiadavku na čítanie pripojeného objektu (premennej). Prepojený objekt zašle žiadosť priradenému procesnému objektu. Pripojený objekt sa nastavuje pomocou LinkID. Ak návratová hodnota tejto funkcie je iná ako 0, tak tým signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. [1]

PviXCreate:

Táto funkcia slúži k nastaveniu cesty k objektu (premennej). Ako príklad môžem uviesť tabuľku číslo 3.1. Ak návratová hodnota tejto funkcie je iná ako 0, tak tým signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. [1]

PviXLink:

Táto funkcia odošle požiadavku PVI Managerovi, aby pripojil objekt (premennú). Ak návratová hodnota tejto funkcie je iná ako 0, tak tým signalizuje chybu. Funkcia vracia hodnotu, ktorá vyjadruje kód chyby. [1]

3.4 Vytvorenie cesty k premennej a jej následné pripojenie

Pre pripojenie sa k premennej v PLC je potrebné s použitím vyššie uvedených funkcií vykonať inicializáciu pripojenia.

Pre vytvorenie pripojenia je potrebné najprv použiť funkciu *PviXInitialize*, ktorá vytvorí komunikačnú inštanciu medzi klientom a PVI Managerom (server). Pri každom zavolaní funkcie *PviXInitialize* sa vytvorí identifikátor (handle), ktorý je pre každé volanie tejto funkcie iný. Po naviazaní komunikácie treba vytvoriť cestu k premennej pomocou funkcie *PviXCreate*.

Funkcia *PviXCreate* pre vytvorenie objektu obsahuje medzi vstupnými parametrami identifikátor, cestu k objektu (reťazec), popis objektu a parametre objektu. Pre pripojenie objektu sa používa funkcia *PviXLink*, medzi ktorej vstupné parametre patrí identifikátor, cesta k objektu a parametre objektu. Táto funkcia neobsahuje popis objektu.

Funkcia, ktorá ukončí komunikačnú inštanciu medzi klientom a serverom je *PviXDeinitialize*. Táto funkcia má jeden vstupný parameter a tým je identifikátor, na základe ktorého sa ukončí správna komunikácia.

Príklad nadviazania komunikácie s objektom nachádzajúcim sa v PLC je uvedený v tabuľke číslo 3.1. Pre vytvorenie objektu je potrebné použiť funkciu *PviXCreate* v každom kroku v tabuľke. Po vytvorení objektu, treba nakoniec použiť funkciu *PviXLink*, ktorej sa predá reťazec vytvorený funkciou *PviXCreate* s cestou k objektu (posledný riadok tabuľky).

Cesta k objektu	PVI objekt
@/Pvi/skus	PVI Line
@/Pvi/skus/AR	PVI Device
@/Pvi/skus/AR/PLC1	PVI Station
@/Pvi/skus/AR/PLC1/Cpu	PVI CPU
@/Pvi/skus/AR/PLC1/Cpu/skuska	PVI Task
@/Pvi/skus/AR/PLC1/Cpu/skuska/x	PVI Variable

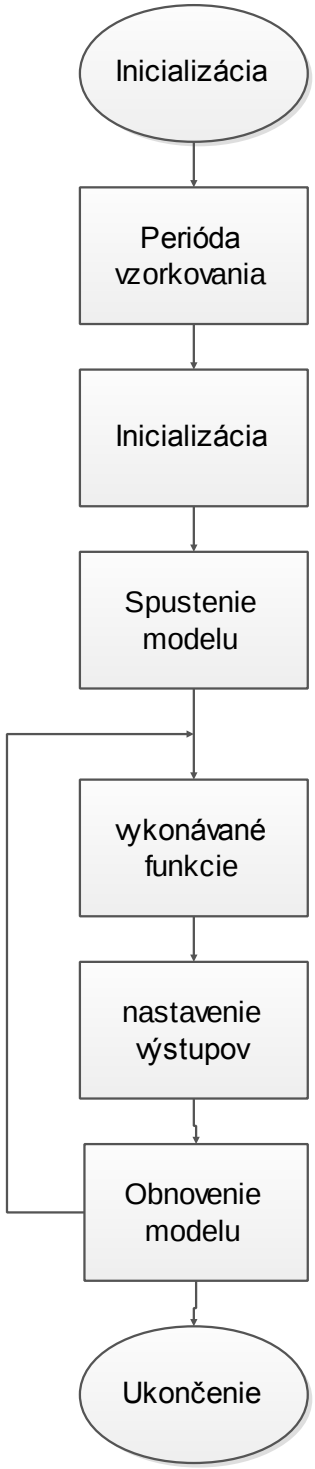
Tabuľka 3.1: Príklad nadviazania komunikácie s premennou *x* v PLC1

4 PRAKTICKÁ ČASŤ

4.1 Použitie v Simulinku

4.1.1 Štruktúra S-Funkcie

S-funkcia má danú štruktúru bez ohľadu na to aký programovací jazyk použijeme pri jej programovaní. Pri zavolaní S-funkcie sa vykonávajú vnútorné funkcie ako je napr. inicializácia, deinicializácia a iné len jeden krát. V inicializačnej časti sa dá nastaviť perióda vzorkovania, počtu vstupov/výstupov a v mojom prípade sú tu funkcie, ktoré čítajú vstupné premenné z masky. V tejto časti sa tiež nastavuje komunikačná inštancia ktorá je pre každý blok špecifická. Ďalej sa tu nachádza cyklická časť, v ktorej sa cyklicky vykonávajú potrebné funkcie napr. k výpočtom. Doba priechodu cyklickou časťou je nastavená periódou vzorkovania. Nakoniec sa tu nachádza deinicializačná časť, v ktorej sa dá ukončiť komunikácia pomocou špecifickej komunikačnej inštancie ktorá je predaná funkcii PviXdeinitialize. Pre vytvorenie programu je veľmi výhodná štruktúra S-funkcie, ktorá je zobrazený na obrázku číslo 4.1. [4]



Obrázok 4.1: Vývojový diagram simulačného cyklu [4]

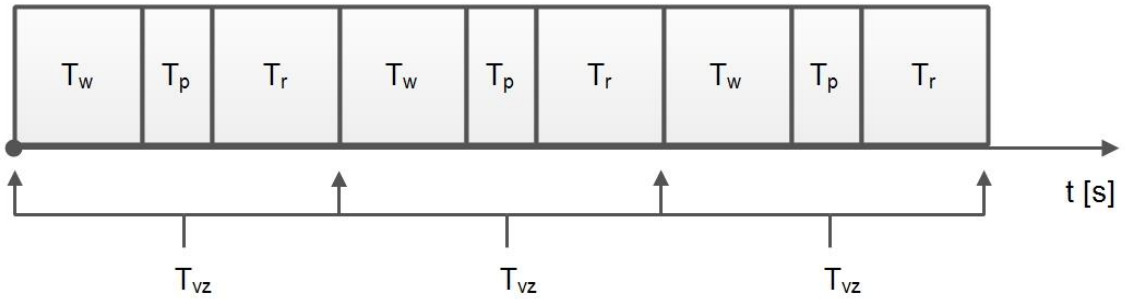
4.1.2 Vytvorenie blokov v Simulinku

V Simulinku som vytváral potrebné bloky pomocou S-funkcie. S-funkciu môžeme vytvoriť niekoľkými spôsobmi. Program, ktorý bol otestovaný a fungoval, bol naprogramovaný v C++, preto som sa rozhodol pre S-function level-2, ktorá podporuje C++ kód. S-funkcia podporuje C++, ale tento kód musí byť najprv preložený funkciou *mex,* ktorá je implementovaná v Matlabe. Funkcia *mex* používa na preklad buď implementovaný kompilér od *MathWorks,* alebo sa dá použiť prekladač od iného výrobcu, ktorý je v počítači implementovaný. V mojom prípade som na kompilovanie použil mnou implementovaný prekladač Microsoft Visual C++ 2010. Po preklade *mex* funkciou sa vytvorí nový súbor, ktorý má rovnaký názov ako prekladaný súbor a príponu s názvom *mexw64* pre 64bitový Windows, *mexw32* pre 32 bitový Windows. Príklad prekladu C++ súboru: *mex zapis.cpp* [4]

V každom bloku je ošetrené spracovanie chýb (error handling). Pri výskyte chyby sa užívateľovi zobrazí okno, v ktorom je napísané, v ktorom bloku a kroku nastala chyba. Pre splnenie zadania tejto práce bolo potrebné zapisovať hodnotu do PLC, čítať hodnotu z PLC a nakoniec dodržovať periódu vzorkovania. Pre splnenie týchto podmienok som vytvoril tri bloky.

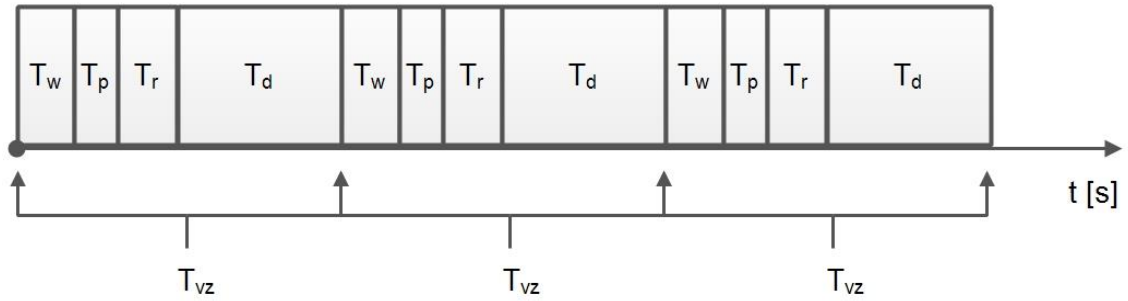
4.1.3 Zaistenie behu simulácie v reálnom čase

Simulink je simulačný nástroj, ktorý nebeží v reálnom čase. Z tohto dôvodu môže napríklad počítať 1 minútu skutočného času ako 1 ms v simulácii a preto bolo potrebné vytvoriť blok perióda, ktorý zariadi, aby sa simulačný čas priblížil čo najviac reálnemu času. Tento blok musí obsahovať každý model, v ktorom sa používajú funkcie zápisu alebo čítania. Dobu zápisu do premennej označím ako T_w, dobu čítania premennej označím ako T_r a čas spracovávania inštrukcií procesora označím ako T_p. Perióda vzorkovania je označená ako T_{vz}. Bez bloku perióda (v ktorom by sme definovali periódu vzorkovania) Simulink vzorkuje rýchlejšie ako je potrebné viď obrázok číslo 4.2.



Obrázok 4.2: Dĺžka periódy vzorkovania bez bloku perióda

Po pridaní bloku perióda si nastavíme periódu vzorkovania (T_{vz}), ktorá zariadi aby sa vzorkovacia perióda rovnala skutočnej požadovanej vzorkovacej perióde. Simulink vzorkuje rýchlejšie ako má a preto potrebujeme zavedením novej premennej $T_{d,}$ ktorú prestavuje blok perióda, spomaliť tento cyklus na požadovaný čas. Príklad funkcie je zobrazený na obrázku číslo 4.3.



Obrázok 4.3: Dĺžka periódy vzorkovania s implementovaným blokom perióda

Ako bolo už vyššie spomenuté, tak v bloku prerióda dochádza k čítaniu hodnoty typu unsigned integer, pretože tento dátový typ viac vyhovuje nižšie popísanej podmienke. Táto premenná sa inkrementuje napríklad v 10 ms cyklickej slučke v PLC. Inkrementovanie tejto premennej je najdôležitejšou časťou ovládania presnosti periódy vzorkovania.

Nedodržanie periódy vzorkovania sa kontroluje v kóde pomocou podmienky (1). Táto podmienka funguje tak, že ak používame na riadenie fyzikálnej sústavy program, ktorý je umiestnený v 100 ms cyklickej slučke, tak inkrementovaním hodnoty v 10 ms slučke dokážeme kontrolovať presnosť 100 ms slučky. Pokiaľ sa vykoná jeden cyklus 100 ms slučky, tak sa v 10 ms slučke inrementuje hodnota 10 krát.
Po každom úspešnom ukončení cyklu sa uchováva premenná, ktorú označím ako $H(k-1)$. Ak od aktuálnej hodnoty premennej, ktorú označím H odčítam $H(k-1)$ a nedostanem rozdiel 10, tak sa na druhom výstupe bloku objaví impulz.

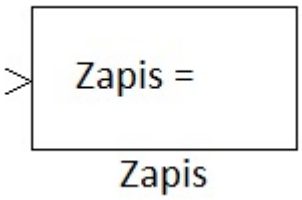
Podmienka:

$$(H - H(k-1)) \geq \left(\left(\frac{T_{vz}}{0{,}01}\right) - 1 + 0{,}1\right) \qquad (1)$$

$$T_{vz} = zadaná\ perióda\ vzorkovania$$

Podmienka funguje iba v tom prípade, ak je výsledok podielu $\frac{T_{vz}}{0{,}01}$ celé číslo. Kontrolovať inkrementovanú hodnotu je možné na prvom výstupe z bloku a nedodržanie periódy vzorkovania je možné kontrolovať na druhom výstupe z bloku.

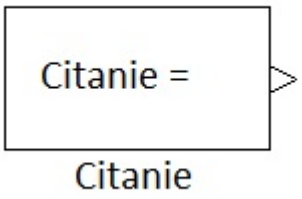
4.1.4 Blok pre zápis hodnoty do PLC



Obrázok 4.4: Vzhľad bloku pre zápis premennej v Simulinku

Tento blok slúži na zápis hodnoty do PLC. Používajú sa tu funkcie, ktoré sú popísané v kapitole 3.4 a nakoniec najdôležitejšia funkcia pre zápis hodnoty PviXWrite. Hodnoty, ktoré tento blok zapisuje do PLC sú v Simulinku vo formáte double. V programe, ktorý sa používa v PLC, aby bola zabezpečená kompatibilita formátov, musí byť hodnota, do ktorej zapisujeme, vo formáte LREAL (Long Real).
V bloku pre zápis hodnoty do PLC sa názov premennej zobrazí za znakom „=“. Tento blok v simulácii vykoná jeden zápis za periódu.

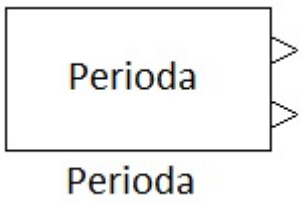
4.1.5 Blok pre čítanie hodnoty z PLC



Obrázok 4.5: Vzhľad bloku pre čítanie premennej v Simulinku

Tento blok slúži na čítanie hodnoty z PLC. Používajú sa tu funkcie, ktoré sú popísané v kapitole 3.4 a nakoniec najdôležitejšia funkcia pre čítanie hodnoty PviXRead. Hodnoty, ktoré tento blok číta z PLC sú v PLC vo formáte LREAL (Long Real). V matlabe sa potom premenná ukladá vo formáte double. V bloku pri čítaní hodnoty z PLC sa názov premennej zobrazí za znakom „=“. Tento blok v simulácii vykoná jedno čítanie za periódu.

4.1.6 Blok perióda



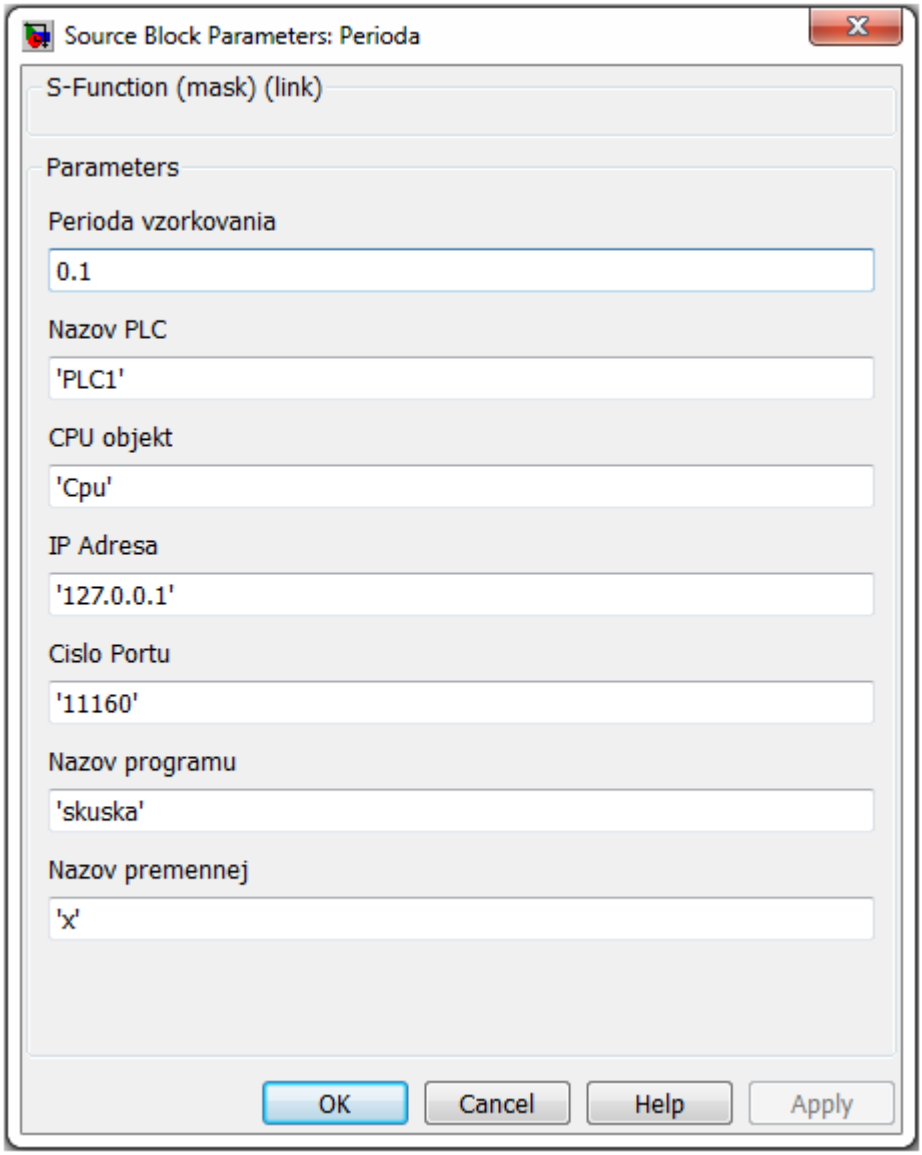
Obrázok 4.6: Vzhľad bloku perióda v Simulinku

Tento blok slúži na to, aby sa čas vykonávania simulačného cyklu rovnal skutočnému času, čo je bližšie vysvetlené v kapitole 4.3. Používajú sa tu funkcie, ktoré sú popísané v kapitole 3.4 a nakoniec najdôležitejšia funkcia pre čítanie hodnoty PviXRead. Tento blok obsahuje dva výstupy. Na prvom výstupe užívateľ môže kontrolovať inkrementovanú hodnotu a na druhom výstupe je jednotkovým impulzom zobrazené nedodržanie periódy vzorkovania. V Simulinku má každý blok svoju prioritu. Pre blok perióda je dôležité nastaviť najnižšiu prioritu, aby sa volal ako posledný.

Je potrebné aby sa v každom modely v ktorom sa používajú bloky čítanie/zápis nachádzal blok perióda pre zaistenie behu simulácie v reálnom čase.

4.1.7 Uživateľská maska

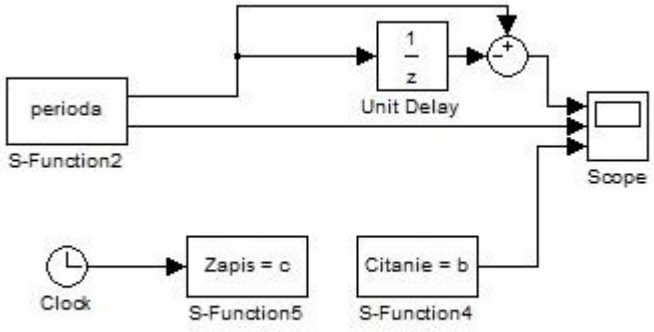
Užívateľskú masku som vytvoril preto, aby užívateľ nemusel zložito meniť reťazce v S-funkcii a náhodou nepoškodil kód. Získané reťazce z masky sa predávajú vo vnútri S-funkcie v inicializačnej časti. Vo vnútri S-funkcie sú niektoré parametre pevne nastavené a užívateľ k ním nemá prístup. Prvky, ktoré môže užívateľ v maske zadávať sú zobrazené na obrázku číslo 4.7.



Obrázok 4.7: Maska bloku perióda

4.1.8 Detekcia nedodržania periódy vzorkovania

V tomto prípade bude potrebné vytvoriť funkciu, v ktorej bude zabezpečené, aby užívateľ vedel, kedy nebola dodržaná perióda vzorkovania. Táto indikácia je potrebná pri identifikácii sústavy a nasledovnom spracovávaní dát pre výpočet regulátora. Indikácia nedodržania periódy vzorkovania je implementovaná v bloku s menom *perioda*. Tento blok obsahuje dva výstupy. Jeden výstup je hodnota inkrementovanej premennej napr. v 10 ms slučke a druhý výstup zobrazuje nedodržanie vzorkovacej periódy. Na základe indikácie nedodržania vzorkovacej periódy je možné vypočítať predpokladanú hodnotu meranej veličiny. Túto hodnotu je možné spočítať z hodnôt inkrementovanej hodnoty. Tento výpočet sa dá jednoducho naprogramovať tak, že od aktuálnej hodnoty odpočítam minulú hodnotu. Príklad naprogramovania výpočtu v Simulinku je zobrazený na obrázku číslo 4.8.



Obrázok 4.8: Príklad výpočtu oneskorenia v Simulinku

4.1.9 Meranie vlastností vzorkovacej periódy

Pri meraní vlastností vzorkovacej periódy som najprv skúšal komunikovať so similátorom. Po základnom meraní času simulácie som však zistil, že niečo nie je v poriadku. Simulácia bola nastavená na 20 sekúnd, ale v skutočnosti trvala 25 sekúnd. Po opakovanom meraní som dostával rovnaké výsledky. Bolo nutné sa pripojiť k PLC, aby som zistil či je chyba v kóde alebo v simulátore. Po prepojení PLC a PC som zistil chybu. Chyba bola v simulátore.
Na to, aby som mohol merať reálny čas periódy vzorkovania (nie simulačný), tak som musel na meranie času implikovať do S-funkcie funkciu *timeGetTime* z C++. [5]
Blok perióda obsahoval pri testovaný tri výstupy, pričom na treťom výstupe bol skutočný čas periódy vzorkovania.

Skutočný čas periódy vzorkovania som exportoval zo Simulinku do matlabu pomocou bloku *simout*. Dáta som analyzoval a výsledky sú zobrazené v nasledujúcich grafoch a tabuľkách. Pri plnom zaťažení CPU som nezistil žiadne ovplyvnenie periódy vzorkovania. Priorita MATLABU a programu, ktorý zaťažil procesor, bola rovnaká. Z grafov vyplýva, že ako najlepšie periódy vzorkovania vyšli T_{vz} = 10 ms a T_{vz} = 100 ms.

Výpočet strednej hodnoty:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

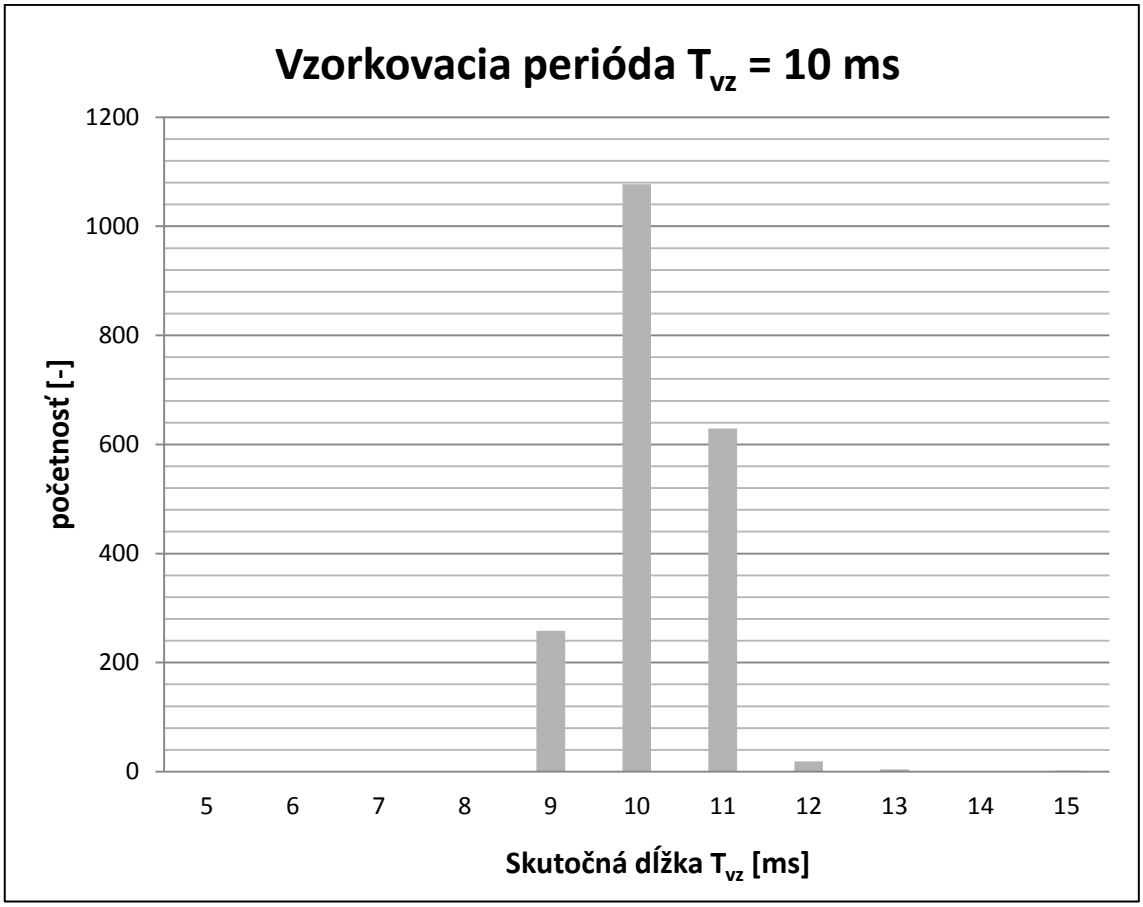
Výpočet rozptylu:

$$\sigma = \frac{1}{N}\sum_{i=0}^{N} (x_i - \overline{x})^2 \tag{3}$$

Výpočet smerodatnej odchýlky:

$$s = \sqrt{\frac{1}{N}\sum_{i=0}^{N} (x_i - \overline{x})^2} \tag{4}$$

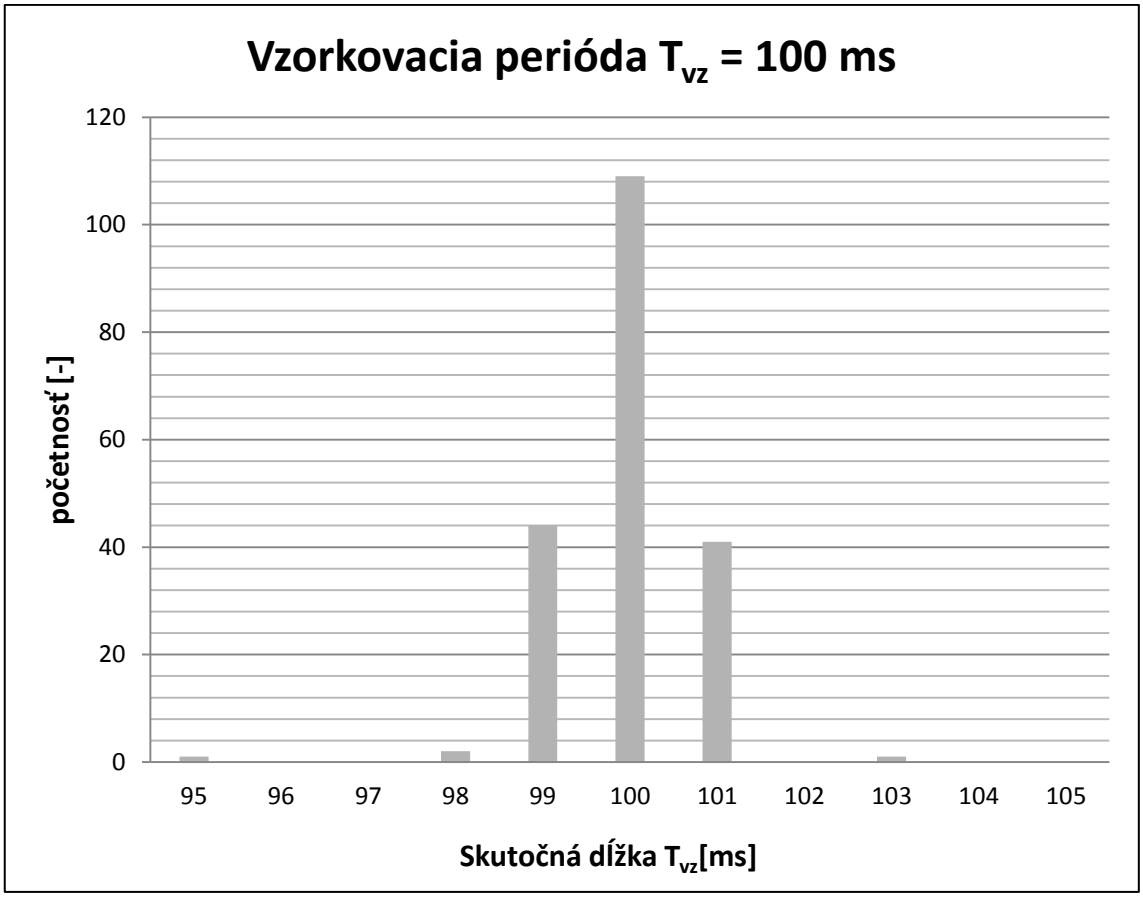
Kde N značí počet prvkov a *x* jednotlivé prvky.



Obrázok 4.9: Meranie periódy vzorkovania T_{vz} = 10 ms

Stredná hodnota [ms] (2)	10,254
Rozptyl [ms] (3)	0,9485
Smerodajná odchýlka [ms] (4)	0,9739

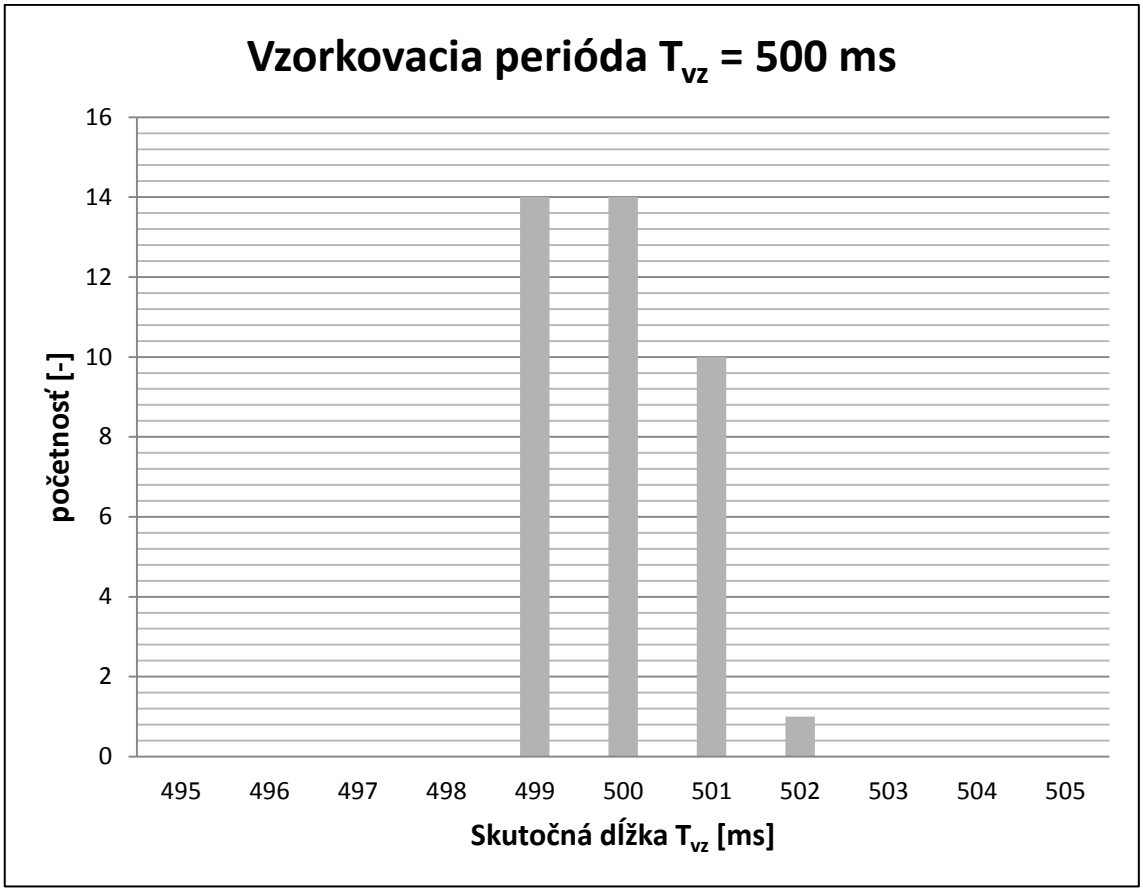
Tabuľka 4.1: Vypočítané hodnoty pre T_{vz} = 10 ms



Obrázok 4.10: Meranie periódy vzorkovania T_{vz} = 100 ms

Stredná hodnota [ms] (2)	99,98
Rozptyl [ms] (3)	1,2396
Smerodajná odchýlka [ms] (4)	1,1134

Tabuľka 4.2: Vypočítané hodnoty pre T_{vz} = 100 ms

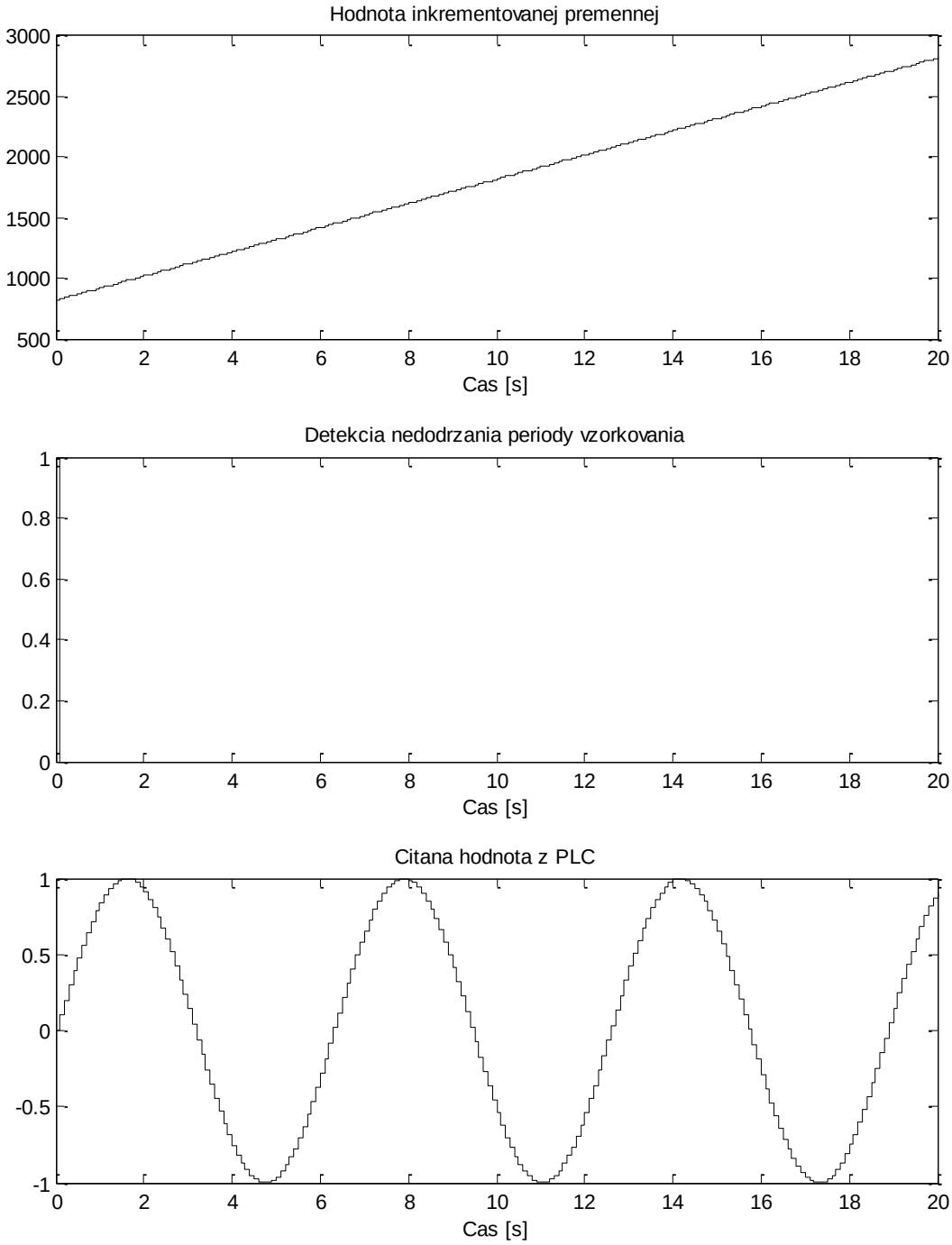


Obrázok 4.11: Meranie periódy vzorkovania T_{vz} = 500 ms

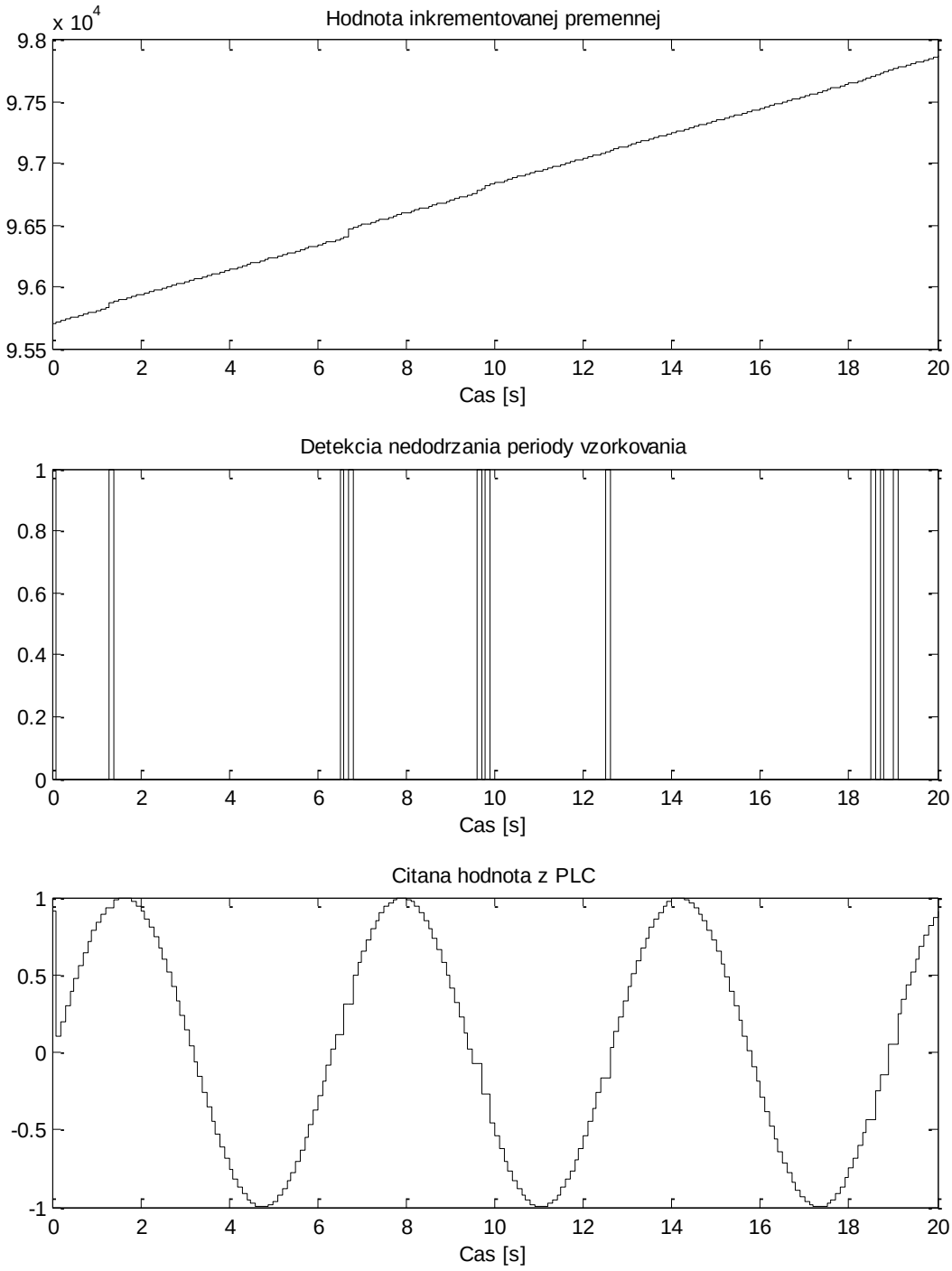
Stredná hodnota [ms] (2)	99,985
Rozptyl [ms] (3)	0,7194
Smerodajná odchýlka [ms] (4)	0,8482

Tabuľka 4.3: Vypočítané hodnoty pre T_{vz} = 500 ms

Pri testovaní som zistil, že pri zapnutí bloku *scope* v Simulinku, použití automatického zaostrenia v *scope* alebo pri pohnutí bloku, perióda vzorkovania vykazuje chybu. Táto chyba sa objavuje aj pri zapnutí modelu. Ako názorný príklad poukazujem na graf číslo 4.5 s periódou bez zásahu v Simulinku a graf číslo 4.6 so zásahom v Simulinku. Počas testu bola T_{vz} nastavená na 100 ms.



Obrázok 4.12: Ukážka periódy vzorkovania bez zásahu v Simulinku



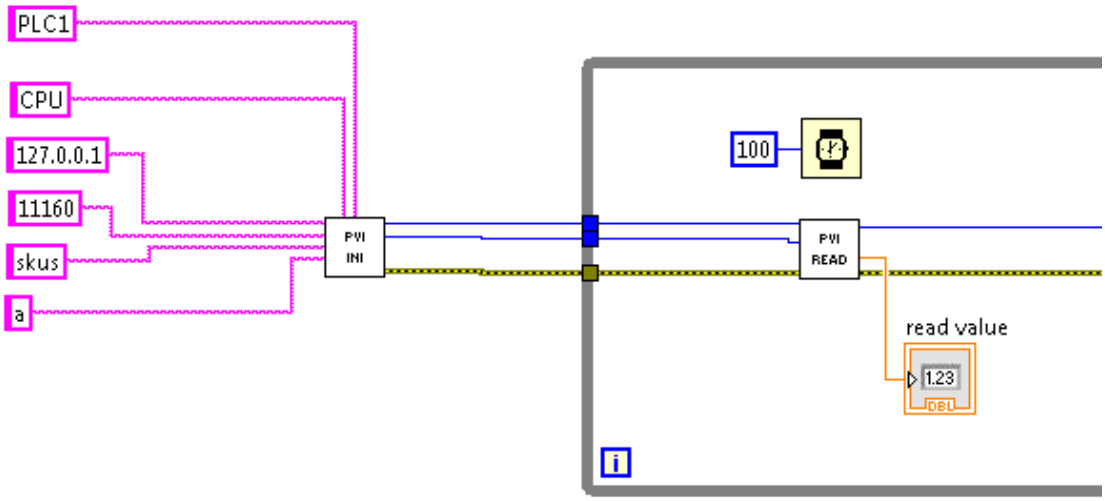
Obrázok 4.13: Ukážka periódy vzorkovania so zásahom v Simulinku

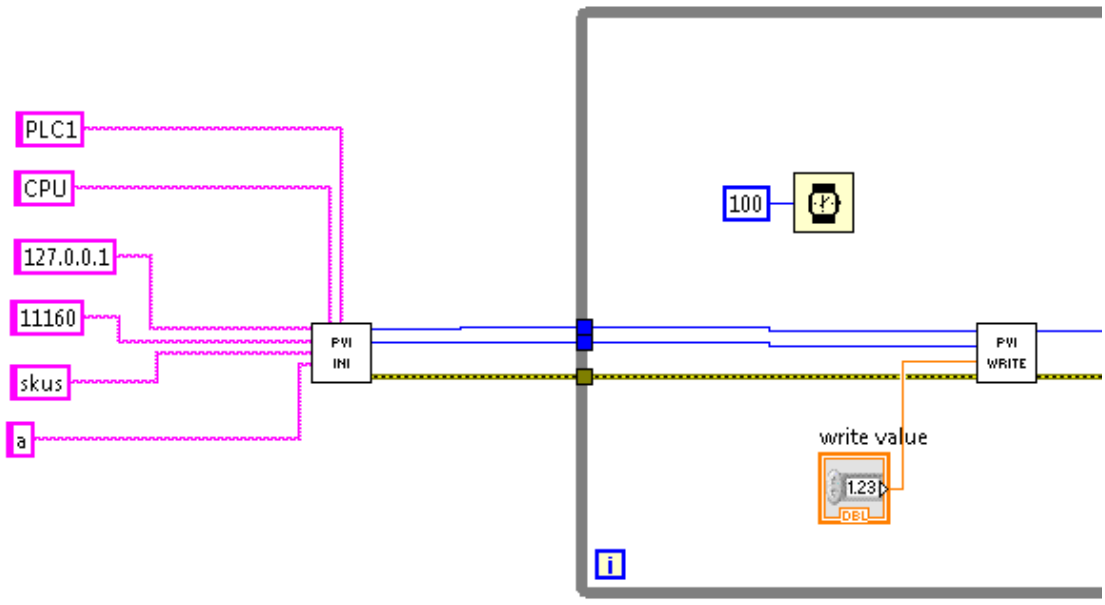
4.2 Použitie v LabVIEW

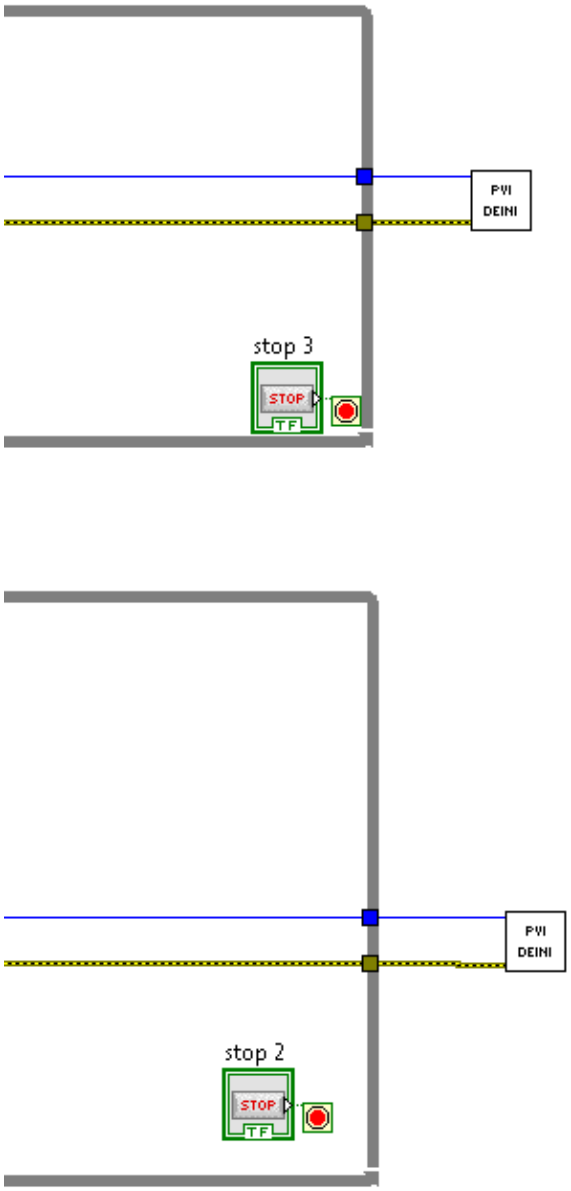
Pre LabVIEW som vytvoril dve knižnice. Knižnicu *MyPvi.dll* pre 32 bitovú verziu systému Windows a *MyPvix64.dll* pre 64 bitovú verziu systému Windows. V týchto dvoch knižniciach sa nachádzajú štyri funkcie, ktoré sú volané prostredníctvom *Call library function node*:

- `NaviazSpojenie` (inicializácia)
- `Citanie` (čítanie premennej)
- `Zapis` (zápis do premennej)
- `Deinicializacia`

Funkcia programu bola odskúšaná. Program funguje, ale bohužiaľ z nedostatku času nebolo možné otestovať ako sa simulácia presne správa v reálnom čase a pri použití riadenie vzorkovacej periódy z PLC. V skúšobnej verzii programu je perióda vzorkovania určená blokom *wait for* umiestnenom v cyklickej slučke *while*. Ako ukážku funkčného programu prikladám obrázok blokového diagramu funkcie *main.vi*.


PLC1
CPU
127.0.0.1
11160
skus
a
PVI
INI
100
PVI
READ
read value
1.23
DBL
i


PLC1
CPU
127.0.0.1
11160
skus
a
PVI
INI
100
PVI
WRITE
write value
1.23
DBL
i



Obrázok 4.14: Ukážka blokového diagramu funkcie *main.vi* v programe NI LabVIEW

4.3 regulácia fyzikálneho modelu

Aby som mohol dôveryhodne otestovať vytvorené funkcie, tak som sa ich rozhodol demonštrovať na riadení fyzikálnej sústavy. Pred tým, ako som začal navrhovať regulátor na danú sústavu, som najprv potreboval zistiť koľkého rádu táto sústava je a tiež zistiť jej prenos. Fyzikálny model bol model veterného tunelu.

Návrh:

Matematický model sústavy bol získaný minimalizáciou kvadratickej kriteriálnej funkcie (5)

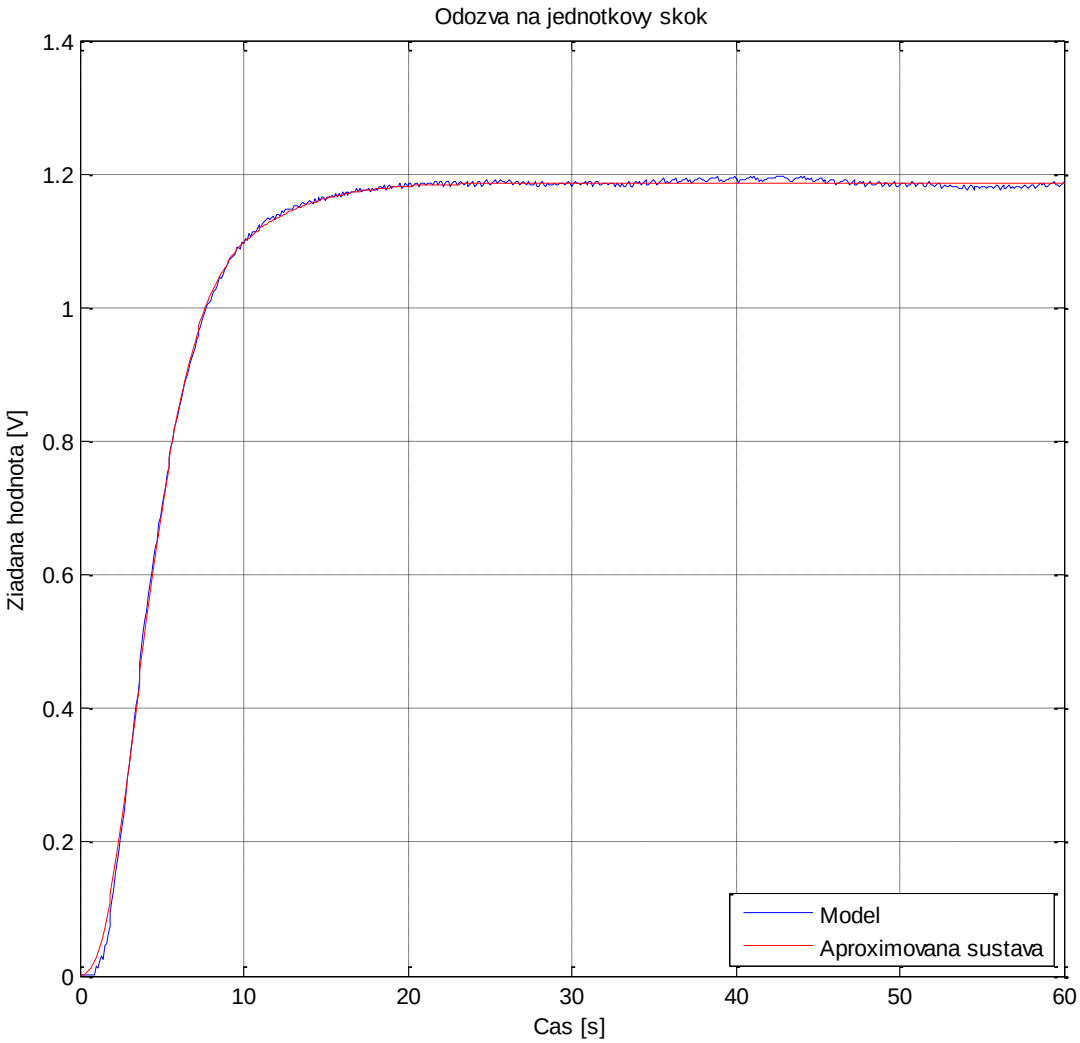
$$J = \frac{1}{N}\sum_{i=0}^{N-1}\left(h_M(i) - h_s(i)\right)^2 \tag{5}$$

kde $h_M(i)$ predstavuje hodnotu skokovej odozvy matematického modelu v čase i a $h_s(i)$ hodnotu skokovej odozvy fyzikálnej sústavy v odpovedajúcom čase. Štruktúra modelu bola zvolená v tvare (6)

$$F_M(p) = \frac{k}{a_3p^3 + a_2p^2 + a_1p + 1} \tag{6}$$

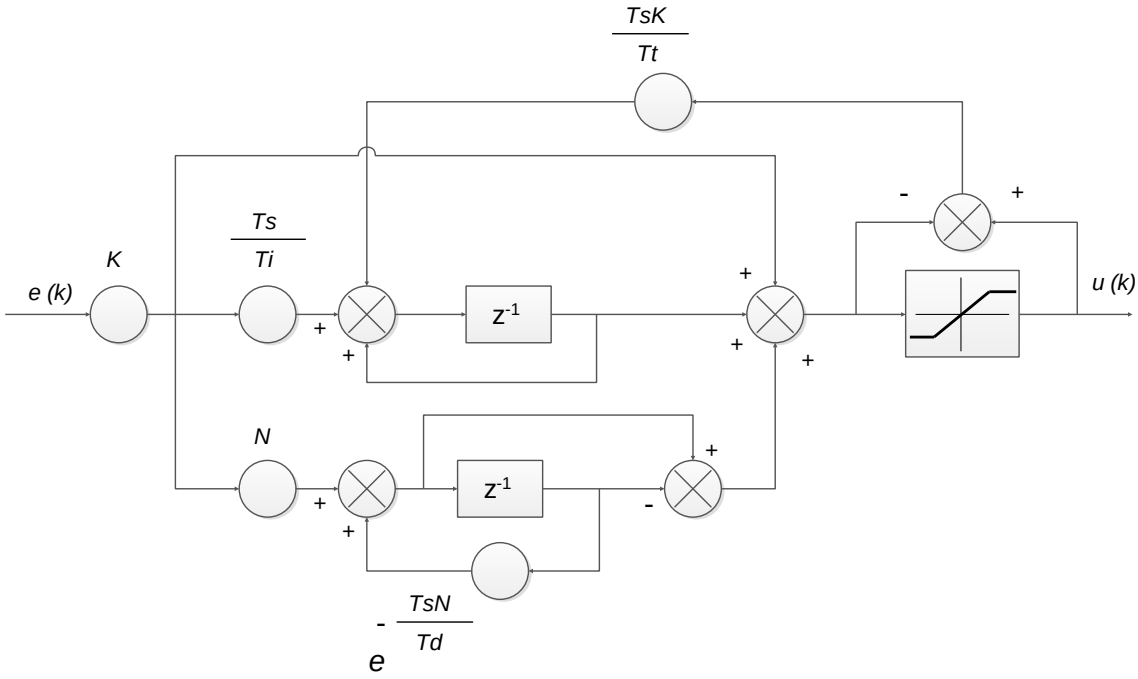
Hodnoty parametrov k, a_3, a_2, a_1 modelu (6) minimalizujú kriteriálnu funkciu (5) a boli nájdené simplexovou metódou realizovanou funkciou *fminsearch*. Výsledný operátorový prenos popisuje vzťah (7). Odozvy reálnej sústavy a nájdeného matematického modelu sú na obrázku číslo 4.15.

$$Fs(p) = \frac{1{,}187}{6{,}783p^3 + 8{,}075p^2 + 5{,}225p + 1} \qquad (7)$$

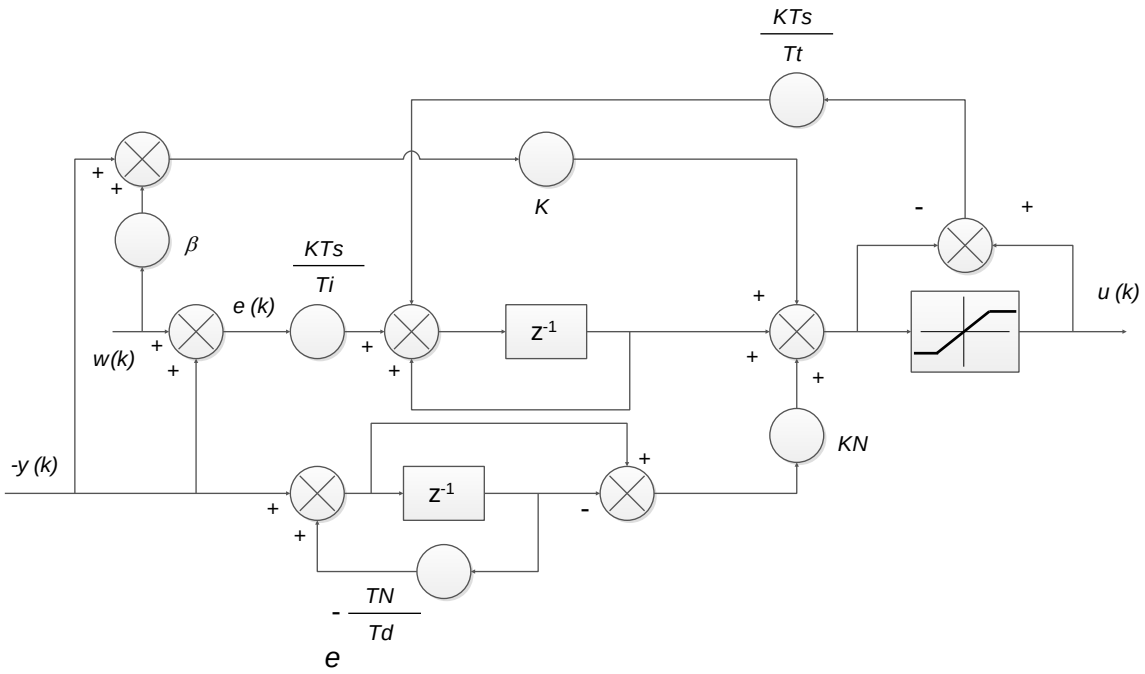


Obrázok 4.15: Odozva na jednotkový skok fyzikálneho modelu (aproximovanej sústavy)

Pre regulovanie sústavy som si vybral dva regulátory. Prvý regulátor je PSD regulátor s filtráciou diferenciálnej zložky a dynamickým obmedzením prebudenia sumačnej zložky (bloková schéma je zobrazená na obrázku číslo 4.16). Ako druhý regulátor som si vybral β-PSD regulátor (bloková schéma je zobrazená na obrázku číslo 4.17).



Obrázok 4.16: Bloková schéma PSD regulátora s filtráciou diferenciálnej zložky a dynamickým obmedzením prebudenia sumačnej zložky [6]



Obrázok 4.17: Bloková schéma $\boldsymbol{\beta}$-PSD regulátora [6]

Pre návrh regulátorov bola v obidvoch prípadoch použitá metóda Ziegler-Nichols (Z-N). Kritické zosilnenie a kritickú periódu som zistil pomocou funkcie *margin.* Parameter $T_t = 2$ som zvolil preto, aby pri vysokej hodnote nevyradil pôsobenie spätnej väzby a pri nízkej hodnote nespomalil rýchlosť regulácie.

$$K_{KRIT} = 4{,}4$$
$$T_{KRIT} = 7{,}16 \text{ p}$$

Výpočet jednotlivých konštánt regulátora:

Zosilnenie:

$$K = 0{,}6 \cdot K_{KRIT} = 2{,}6387$$

Integračná konštanta:

$$T_i = 0{,}5 \cdot T_{KRIT} = 3{,}5795 \text{ p}$$

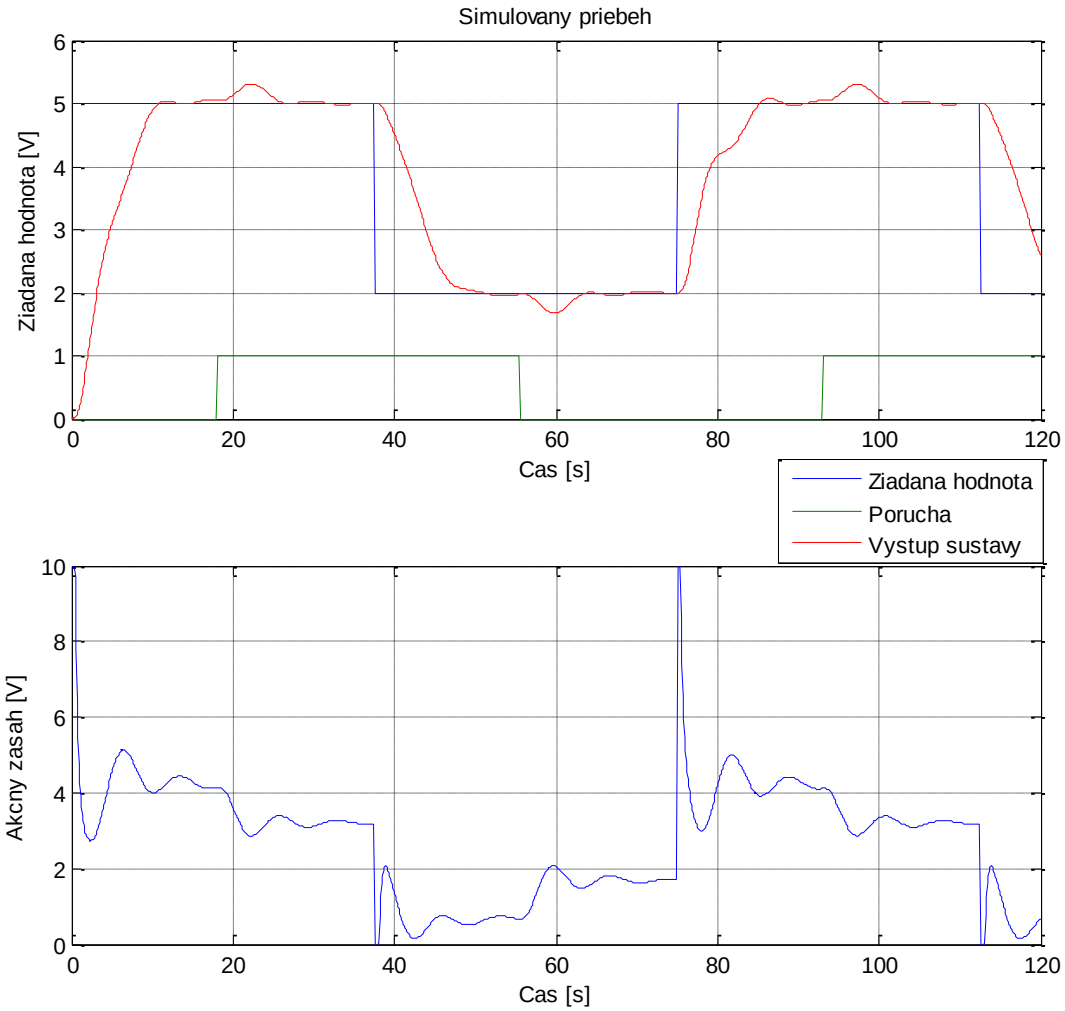
Derivačná konštanta:

$$T_d = 0{,}125 \cdot T_{KRIT} = 0{,}8949 \text{ p}$$

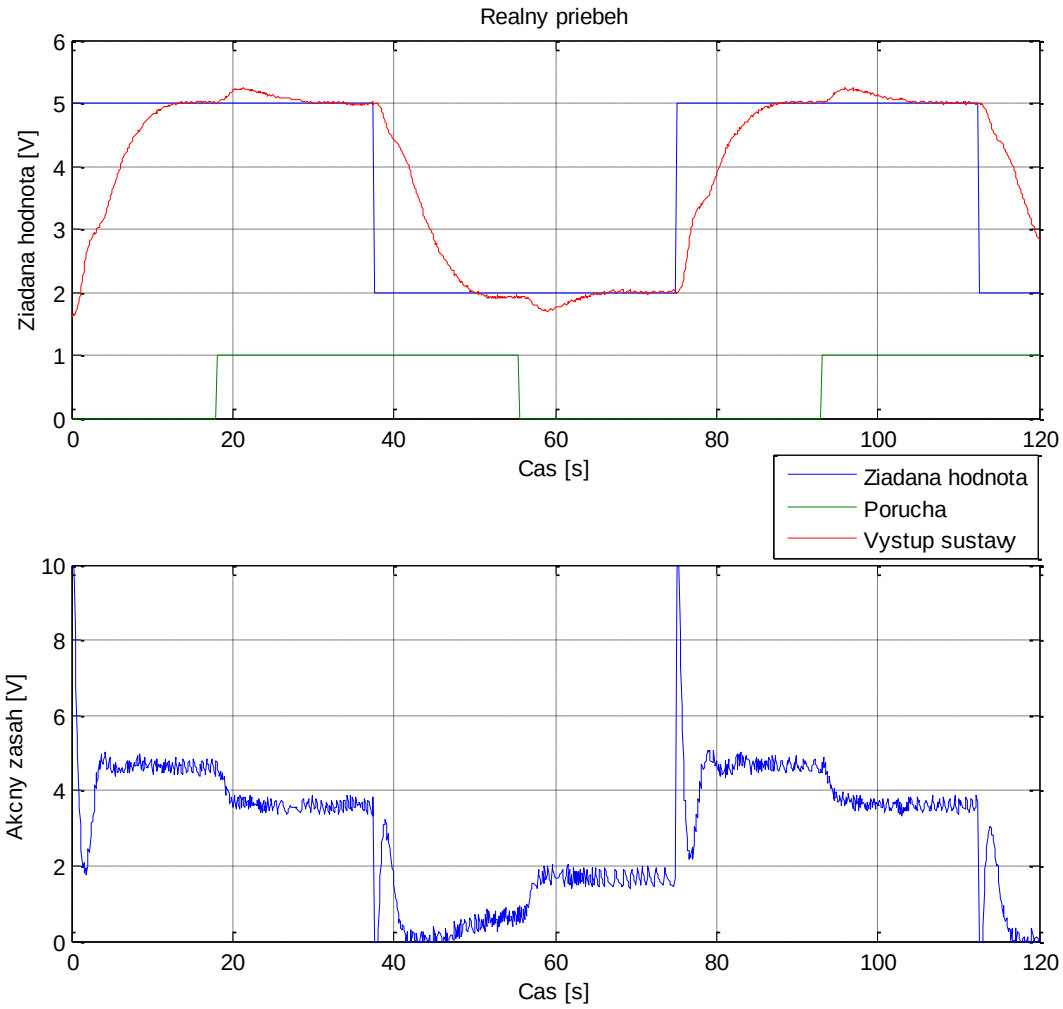
β-PSD:

$$\beta = 0{,}5$$

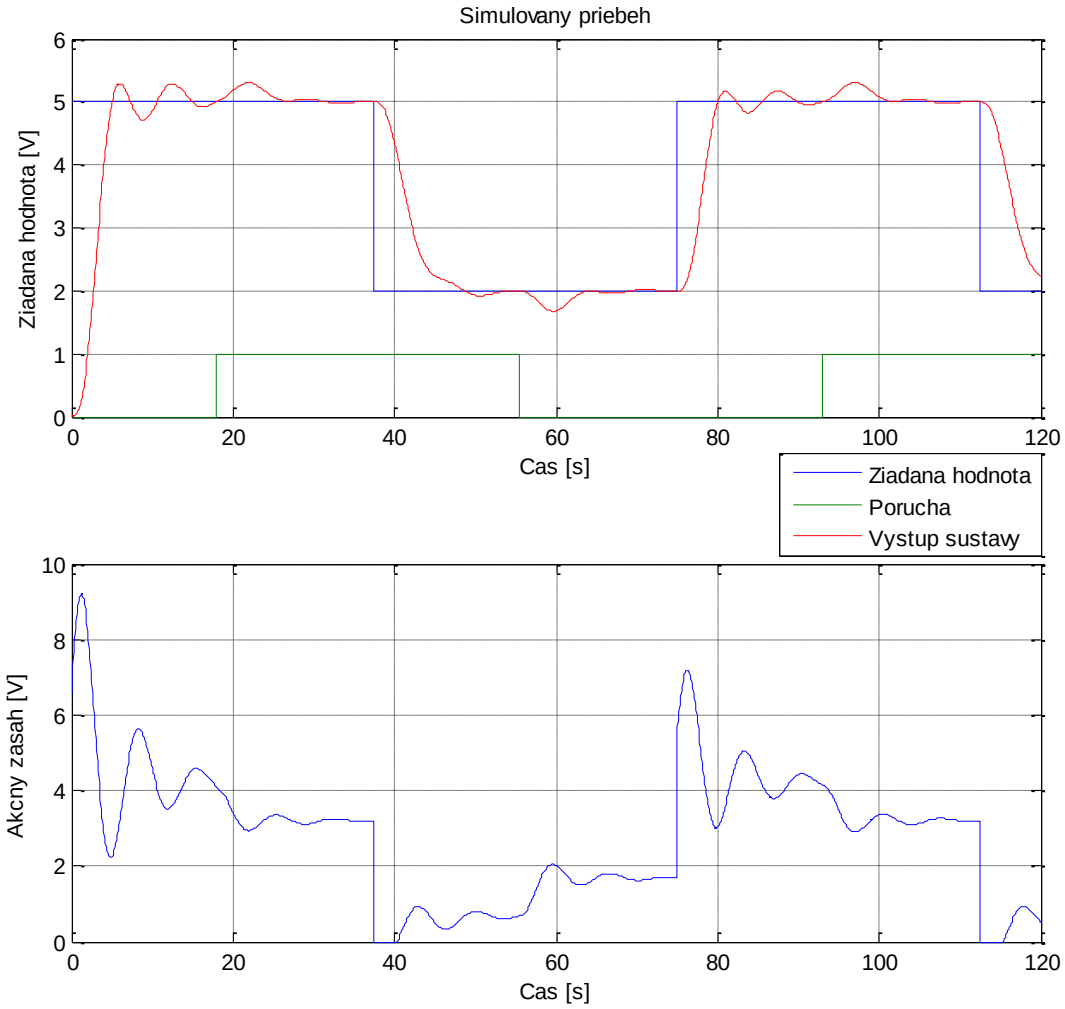
Meranie prebiehalo pri vzorkovacej perióde 100 ms. Akčný zásah mohol nadobudnúť hodnotu od 0 do 10 V. Výsledky meraní sú zobrazené v nasledujúcich obrázkoch, pričom porucha pôsobí na vstupe sústavy.



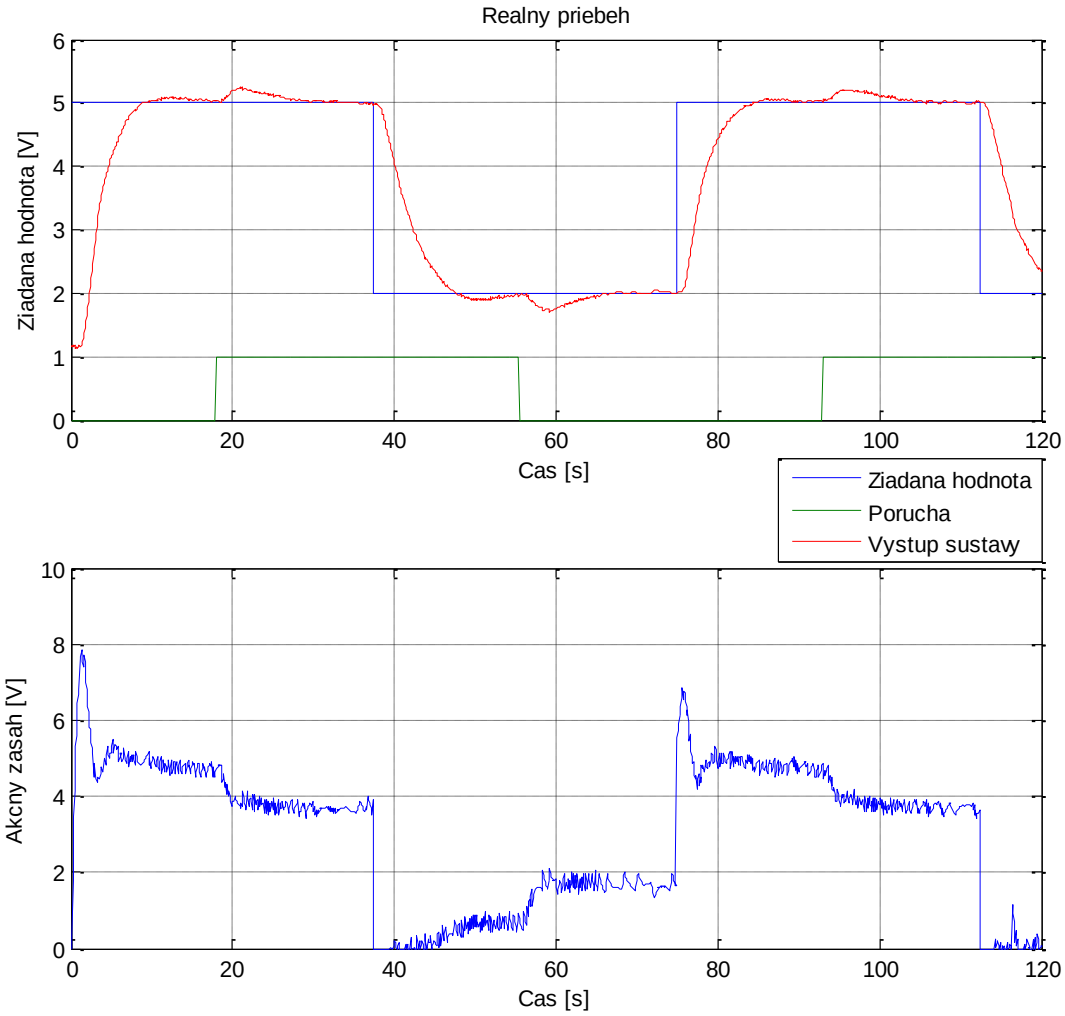
Obrázok 4.18: Priebehy signálov pri regulácii matematického modelu PSD regulátorom



Obrázok 4.19: Priebehy signálov pri regulácii fyzikálnej sústavy PSD regulátorom



Obrázok 4.20: Priebehy signálov pri regulácii matematického modelu $\boldsymbol{\beta}$-PSD regulátorom



Obrázok 4.21: Priebehy signálov pri regulácii fyzikálnej sústavy $\boldsymbol{\beta}$-PSD regulátorom

5 ZÁVER

Pre úspešné vytvorenie komunikácie medzi PC a PLC firmy B&R je potrebné sa podrobne zoznámiť s produktmi firmy B&R. Po preštudovaní Automation Net PVI som potreboval vybrať rozhranie pre komunikáciu s PLC. Z dôvodu rozšíreného využívania rozhrania ethernet som vybral pre komunikáciu toto rozhranie.

Ethernet podporovalo niekoľko komunikačných protokolov. Pri výbere z nich, som sa rozhodol pre komunikačný protokol INA2000, pretože je z môjho pohľadu najuniverzálnejším riešením. Dôležité v tomto projekte je pochopiť objektovú hierarchiu (viď obr. č. 2.2) kvôli prístupu k premenným v PLC.

Pred tým ako som začal vytvárať program som si musel vybrať v akom programovacom jazyku budem programovať, aký režim budem používať (synchrónny/asynchrónny) a aké funkcie budem používať. Programovací jazyk som si zvolil C++, režim som si zvolil synchrónny z dôvodu postupného volania blokov v Simulinku a funkcie sú popísané v kapitole 3.3.

V praktickej časti som vytvoril dva programy. Prvý program sa používa v Simulinku a druhý v LabVIEW.

V programe pre Simulink som vytvoril tri bloky, ktoré zabezpečujú čítanie premennej, zápis premennej a blok pre zaistenie behu simulácie v reálnom čase. Beh simulácie v reálnom čase je riadený čítaním premennej z PLC. Pre implementáciu vytvorených funkcií v C++ do Simulinku som využil S-funkciu level-2. Druh tejto S-funkcie podporuje C++ kód a jej štruktúra je vhodná pre navrhnutú koncepciu riešenia. V každom z týchto blokov je vytvorená maska ktorú užívateľ musí vyplniť aby sa mohol pripojiť k premennej. Kvôli ľahšiemu vloženiu funkcií do Simulinku som vytvoril v MATLABE *m-file*. Tento *m-file* automaticky vloží funkcie do Simulinku. Pri testovaní sa vyskytol v programe problém s deinicializáciou blokov, ktorý bolo nutné odstrániť. Funkčnosť všetkých funkcií bola podrobne otestovaná. Testovanie prebehlo aj na regulácii fyzikálnej sústavy (veterného tunela), pričom boli použité dva regulátory PSD a β-PSD.

V programe pre LabVIEW som vytvoril dve knižnice. Prvá je knižnica s názvom *MyPvi.dll* pre 32 bitovú verziu systému a druhá je *MyPvix64.dll* pre 64 bitovú verziu systému. Pre otestovanie týchto funkcií som vytvoril jednoduchý program ktorého blokový diagram je zobrazený na obrázku číslo 4.14. Beh simulácie v reálnom čase je riadený v PC. Funkčnosť všetkých funkcií bola otestovaná. Bohužiaľ z nedostatku času nebola vyhodnotená presnosť vzorkovacej periódy.

LITERATÚRA

[1] B&R. *Help: Automation Studio*. Rakúsko, 2013. V4.0.15.019. Dostupné z: http://www.br-automation.com/en/

[2] B&R. TM700TRE.00-ENG. Rakúsko, 2007, 46 s.

[3] B&R. TM710TRE.00-ENG. Rakúsko, 2007, 42 s.

[4] Documentation Center. MATHWORKS. *Simulink Documentation* [online]. 1994-2014 [cit. 2014-05-17].
Dostupné z: http://www.mathworks.com/help/simulink/index.html

[5] Windows Dev Center. MICROSOFT. *TimeGetTime function* [online]. 2014 [cit. 2014-05-18]. Dostupné z:
http://msdn.microsoft.com/en-us/library/windows/desktop/dd757629(v=vs.85).aspx

[6] PIVOŇKA, Petr. *Číslicová řídící technika* [pdf]. FEKT VUT v Brně. 2012, 112 s.

Zoznam skratiek

PLC	programovateľný logický automat
PVI	(Process Visualization Interface) hlavný nástroj pre prístup k PLC firmy B&R
PVICOM	rozhranie pre komunikáciu medzi klientom a PVI Managerom
TCP/IP	(Transmission Control Protocol / Internet Protocol) Sieťový komunikačný protokol
INA2000	priemyslové sieťové rozhranie firmy B&R
PC	(Personal com)
CPU	(Central Processing Unit) Procesor
DA	identifikačné číslo cieľovej stanice (PLC)
DLL	(Dynamic-link library) dynamicky linkovaná knižnica
IP	(Internet Protocol) Internetový protokol
Timeout	prerušenie

Zoznam príloh

Príloha 1.

1. V priečinku *LabVIEW* sa nachádzajú všetky súbory pre NI LabVIEW
2. V priečinku *Matlab* sa nachádzajú všetky súbory pre Simulink
3. Elektronická verzia BP