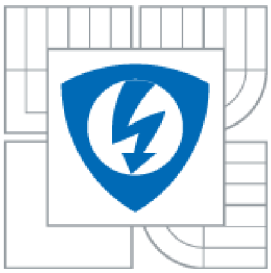
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MATLAB/SIMULINK, REAL-TIME SIMULACE A ŘÍZENÍ

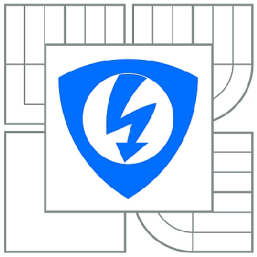
MATLAB/SIMULINK, REAL-TIME SIMULATION AND CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — DALIBOR MALÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN DVOŘÁČEK
SUPERVISOR

BRNO 2010



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:	Dalibor Malík	***ID:***	106612
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

MATLAB/Simulink, Real-Time simulace a řízení

POKYNY PRO VYPRACOVÁNÍ:

Detailně se seznamte s možnostmi a vlastnostmi použití reálného času v prostředí MATLAB/Simulink. V simulinku vytvořte vlastní Blockset, který bude obsahovat blok pro podporu reálného času a bloky pro komunikaci s řídicími systémy firmy B&R. Vytvořený Blockset použijte k řízení vybraných fyzikálních modelů z prostředí MATLAB/Simulink a ověřte základní varianty diskrétního PID regulátoru.

DOPORUČENÁ LITERATURA:

[1] Ogata. K: Modern control engineering, fourth edition. Upper Saddle River, New Jersey: Prentice Hall, 1997. ISBN 0-13-314899-8
[2] Astrom K. J., Wittenmark B: Computer-controlled systeme, theory and design. New Jersey: Prentice Hall, 1997. ISBN 0-13-314899-8
[3] Dorf R. C., Bishop R. H.: Modern control systems, tenth edition. ISBN 0-13-127765-0

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

Vedoucí práce: Ing. Martin Dvořáček

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato práce seznamuje s možností implementace reálného času, sériové linky RS-232 a komunikace s programovatelným automatem (PLC) firmy Bernecker & Rainer (B&R) v prostředí MATLAB/simulink. Podává informace o možnosti vytváření bloku v MATLAB/simulink, funkčním principu simulinku a implementaci bloků do blocksetu.
U vlastní komunikace s PLC je popsán princip PVI, vysvětleny spjaté terminologie a popis nastavení a udržení komunikace mezi zdrojovou stanicí a PLC.

Klíčová slova: s-funkce, reálný čas, rozhraní RS-232, PLC, PVI, blockset, řízení

Abstract

This bachelor thesis provides information about a possibility of implementing the real-time, a serial line RS-232 and communication with a programmable controller (PLC) by Bernecker &Rainer (B & R) in MATLAB / Simulink evironment. Further, it provides information about options of creating a block in MATLAB / Simulink, modes of operation of the simulink and implementation of blocks to the blockset.
In the proper communication with PLC the principle of PVI is described, the terminology connected with PVI is explained and setting and maintaining the communication between a source station and PLC is specified.

Keywords: s-function, real time, RS-232 interface, PLC, PVI, blockset, controls

Bibliografická citace

MALÍK, D. *MATLAB/Simulink, Real-Time simulace a řízení.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 52 s., 10 příloh. Vedoucí bakalářské práce Ing. Martin Dvořáček.

P r o h l á š e n í

„Prohlašuji, že svou bakalářskou práci na téma MATLAB/Simulink real-time simulace a řízení 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 § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: ……

podpis autora

P o ď k o v á n í

Děkuji vedoucímu bakalářské práce Ing. Martinu Dvořáčkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: ……

podpis autora

OBSAH

Seznam obrázků:

Seznam tabulek:

Seznam zkratek:

R-T	reálný čas
RS-232	sériová linka
T_S	vzorkovací perioda
T_U	doba průtahu
T_N	doba náběhu
PLC	programovatelný logický automat
PVI	hlavní nástroj pro přístup k PLC firmy B&R
PVICOM	rozhraní pro komunikaci mezi klientem a PVI manažerem
IP	internetový protokol
TCP/IP	komunikační protokol
INA2000	průmyslové síťové rozhraní firmy B&R
CPU	procesor
PC	stolní počítač
SA	identifikační číslo zdrojové stanice (PC)
DA	identifikační číslo cílové stanice (PLC)
REPO	port cílové stanice
ID	identifikace

1. ÚVOD

MATLAB/Simulink je integrované prostředí pro vědeckotechnické výpočty, modelování, simulace, měření a zpracování signálů, návrhy řídicích a komunikačních systémů, které bez doplnění patřičným algoritmem nepracuje v reálném čase. Reálný čas v prostředí MATLAB/simulink nám dovoluje jednoduchý návrh (ladění) řídicích aplikací v blokovém prostředí a posléze odzkoušení na reálném procesu. Vyhýba se tímto programovaní v jazyce C, které u složitějších aplikací může vést k obtížnějšímu ladění.

Tématem implementací reálného času se zabývá i samotný výrobce programu MATLAB the MathWorks a například firma Humusoft zastupující MathWorks pro Českou republiku a Slovensko. MathWorks vyvinul, mimo jiné na podporu reálného času, toolbox Real-time windows target a xPC target. Real-time windows target využívá přeložení navrženého modelu v simulinku z jazyka embedded matlab do zdrojového kódu v jazyce C, který doplní potřebnými algoritmy pro reálný čas. XPC target je prostředí využívající target PC (stolní nebo průmyslový počítač) odděleně od hostitelského PC (stolní počítač s MATLAB/simulink), pro běh aplikací v reálném čase.

Výše zmíněné real time toolboxy dosahují velmi nízkých vzorkovacích period, ale nevýhodou je složitost implementace, kdy zavedení reálného času není ani obsáhle v simulačním modelu. Dochází k absenci možnosti vykreslování grafů a podobně (ztrácí se vlastnosti simulace). Další nevýhoda bývá poměrně značná cena.

V laboratoři inteligentních regulátorů byla vytvořena Vratislavem Kořínkem komunikace s programovatelným automatem B&R doplněná algoritmem pro zavedení běhu simulace v reálném čase. Jedná se o efektivní doplněk zavádějící reálný čas přímo v simulačním modelu (neztrácí se vlastnosti simulace), využívající funkční princip simulinku. Dochází k pozastavení simulačního cyklu podle nastavené vzorkovací periody. Tento princip dovoluje minimální vzorkovací periodu v řádek desítek milisekund, ale pro potřeby laboratoře je tato vzorkovací perioda dostačující. S vývojem MATLAB/simulink a s rozšířenými možnostmi dochází u koncepce

Vratislava Kořínka k několika zásadním nevýhodám. Problémy s kompatibilitou, neefektivní implementace do simulačního modelu, neudržení přesné vzorkovací periody, složitá konfigurace části komunikace s PLC, implicitně nastavený datový typ a velikost pole a nemožnost komunikace s více proměnnými v PLC z jednoho simulačního modelu. Podstatnou nevýhodou je také nutná komunikace s PLC pro zavedení běhu simulace v reálném čase.

Prioritou nově vytvářené koncepce je odstranit veškeré výše zmíněné nedostatky a vytvořit rozšířenou blokovou podporu pro zavedení soft reálného času, komunikace přes sériové rozhraní RS232 a komunikace s PLC do prostředí MATLAB/simulink. Výsledný blockset obsahující bloky Real-time, Watch TS, serial send, serial receive, serial send-receive, B&R send a B&R receive je určen primárně do laboratoře inteligentních regulátorů pro ověření průběhů navržených řídicích systémů na fyzikálních modelech.

2. ÚVOD DO PROSTŘEDÍ MATLAB/SIMULINK

MATLAB je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.

Simulink je nadstavba MATLABu pro simulaci a modelování dynamických systémů, který využívá algoritmy MATLABu pro numerické řešení nelineárních diferenciálních rovnic. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic.[10]

2.1 MOŽNOSTI VYTVOŘENÍ BLOKU V SIMULINKU

V Matlab/simulink se realizují bloky na základě s-funkce. Možností vytvoření je hned několi. Jedny z nich jsou za pomocí level-2 m-file S-function nebo C MEX S-function, které také byly použity pro vytvoření všech bloků obsažených v této práci. Další možností jsou například level-1 M-file S-function, S-function builder a The Legacy Code Tool.

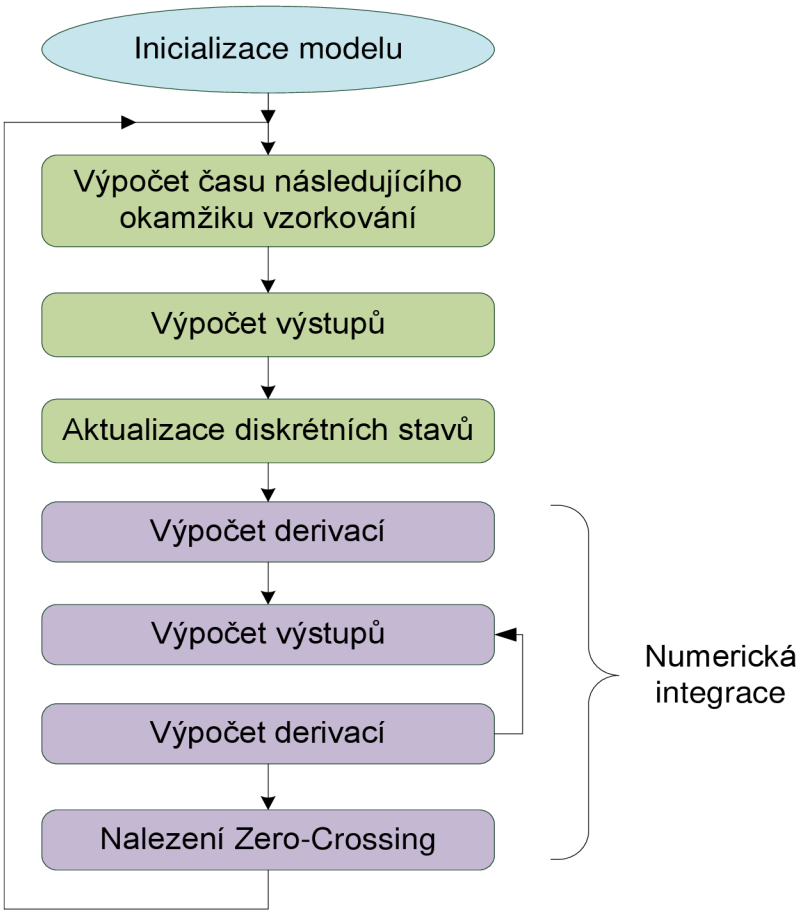
V level-2 M-file s-funkci se používají skripty psané jako textové soubory s příponou .m. Výhodou psaní kódu v M jazyku je využitelnost funkcí, které prostředí MATLAB nabízí. Například na úpravu vnějších parametrů bloku, masky s-funkce a celkově funkce pro práci s prostředím MATLAB.

Pokud je potřebná implementace knihovny nebo funkcí v jazyce C, lze využít C MEX s-funkci. Jde opět o vytvoření textového souboru nyní s příponou .c, se kterým ale matlab/simulink přímo nepracuje, ale je potřebné přeložení na soubor typu .mex. Pro kompilaci je možné využít překladače implicitně nastaveného v MATLABu, nebo nadefinovat vlastní překladač používaný v počítači.

Oběma typy s-funkcí je možné vytvoření uživatelských bloků s vícenásobnými vstupy/výstupy a zacházet s každým typem signálu, který se v simulačním modelu může vyskytovat. Přijaté data mohou být v bloku libovolně zpracovány a poskytnuty dalším blokům. [1],[11]

2.2 STRUKTURA S-FUNKCE

Struktura s-funkce je dána bez ohledu na použitém typu programovacího jazyka. Obsahuje funkce, které jsou volány a prováděny v každém simulačním cyklu a funkce inicializační, které se provedou pouze při vytvoření bloku, nebo při změně vstupních parametrů. Simulační cyklus je přehledně zobrazen a popsán na vývojovém diagramu viz obrázek 1 (obsahuje pouze základní funkce). Inicializace modelu odpovídá inicializačním funkcím, kde dochází k nastavení vzorkovací periody, vstupů/výstupů bloku a podobně. Následují funkce, které jsou volány v každém simulačním cyklu neboli při diskrétních simulacích každou vzorkovací periodu. Poslední tři bloky ve vývojovém algoritmu simulačního cyklu jsou volány pouze, pokud je v simulačním modelu nastaven běh simulace kontinuálně. Dochází v nich k použití numerické integrace výstupu pro získání pseudospojitého průběhu. [11]



Obrázek 1: Algoritmus simulačního cyklu [11]

2.3 MATLAB/SIMULINK MOŽNOSTI REÁLNÉHO ČASU

Výrobce matlab/simulink (the mathworks) nabízí několik možností jak docílit zavedení běhu simulovaného modelu v reálném čase. Jedny z možností jsou toolbox Real-time windows target a xPC target. S Real-time windows target úzce souvisí toolboxy Real-time workshop a Real-time workshop Embedded Coder, které rychle a efektivně přeloží navržený model v simulinku z jazyka embedded matlab do zdrojového kódu v jazyce C. Následně pomocí Real-time windows target může být C kód doplněn vhodnými algoritmy pro běh simulace v reálném čase. Značnou nevýhodou se jeví složitost implementace a flexibilita použití. XPC target je prostředí využívající target PC odděleně od hostitelského PC, pro běh aplikací v reálném čase. Hostitelské PC představuje stolní počítač s MATLAB/simulink, na němž je možné vytvořit simulační model v nereálném čase. XPC target software umožňuje přidat vstupní/výstupní bloky do simulačního modelu a poté pomocí hostitelského PC s Real-Time Workshop, Real-Time Workshop Embedded Coder a C/C++ kompilátorem vytvořit spustitelný kód. Spustitelný kód je stažen z hostitelského PC do target PC běžícího na xPC target real time jádře. Target PC může být zprostředkovaný i obyčejným stolním počítačem nebo řadou průmyslových počítačů. Nevýhodou je oddělení aplikace na jiné PC, čímž dochází ke ztrátě vlastností simulace, ale je podporována velmi nízká vzorkovací perioda. [11]

Další možností implementace reálného času do matlab/simulink je použití toolboxu navrženým společností Humusoft. Humusoft se zabývá hlavně matlab/simulink a věcmi k němu spjatých. Vyvinuli vlastní real-time toolbox, který je možno v demoverzi stáhnout z jejich stránek [10]. Přesněji demoverzi real time toolbox 4.0.1 pro Matlab R14 SP3 a vyšší. K otestování daného real time toolboxu nedošlo, ale z parametrů udaných výrobcem je možné vyčíst uváděnou minimální vzorkovací frekvenci v rozmezí 100Hz-25kHz, což odpovídá vzorkovací periodě 10ms-40us v závislosti na složitosti simulovaném modelu. K použití stačí pouze zavedení toolboxu do simulačního modelu, ale nevýhodou je cena. [10]

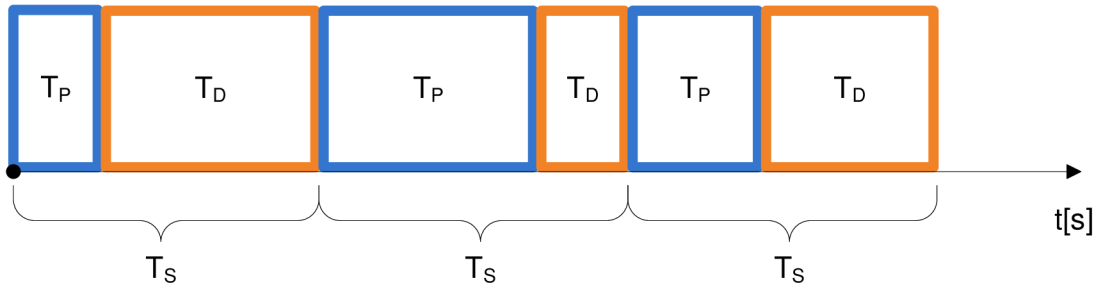
Další koncepcí je R-T blok používaný na VUT v laboratoři inteligentních regulátorů vytvořený Vratislavem Kořínkem. Jedná se o blok pro komunikaci s programovatelným logickým automatem firmy B&R, doplněným algoritmem pro

běh simulačního modelu v reálném čase. Algoritmus využívá funkční princip simulinku (cyklické opakování). Jak se ukázalo v průběhu testování, dochází k poměrně velkému rozptylu délky vzorkovací periody (nevýhoda například při regulaci). Střední hodnota všech průchodů simulačním cyklem však odpovídá vzorkovací periodě, tzn. simulace běží v reálném čase. Značnou nevýhodou je použití při návrhu již staré verze MATLABu a tím spjaté problémy s kompatibilitou, složitou implementací a konfigurací. Dále nelze R-T zavést do simulačního modelu bez navázání komunikace s PLC.

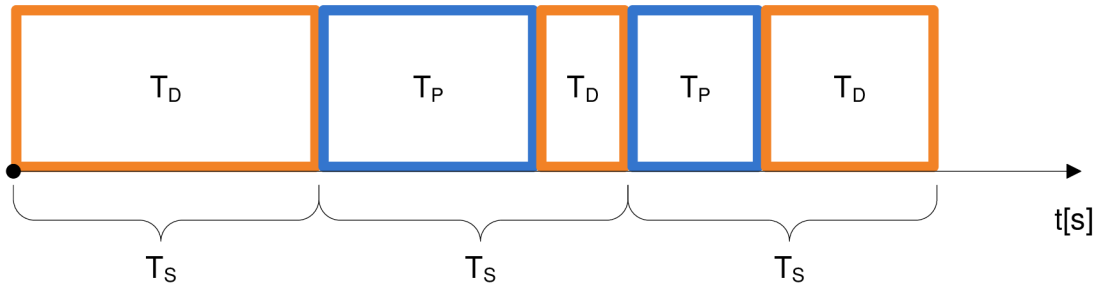
3. ZAJIŠTĚNÍ BĚHU SIMULACE V REÁLNÉM ČASE

3.1 REALIZACE REAL-TIME BLOKU

Běh simulinku v reálném čase byl navržen na principu dorovnávání simulačního cyklu na velikost vzorovací periody. Simulační cyklus na obrázek 1 je současně spouštěn ve všech s-funkcích obsažených v simulačním modelu a postupně se provádí jednotlivé funkce. Pokud dojde v určitém bloku ke zdržení, projeví se toto zdržení v celé simulaci. Dobu zpracování instrukcí procesorem označme T_P. Při nastavené vzorkovací periodě (T_S) je třeba, aby jeden simulační cyklus trval právě jednu zvolenou vzorkovací periodu. Této podmínky lze dosáhnout vytvořením algoritmu, který je schopný dorovnávat dobu T_P na velikost zadané vzorkovací periody, neboli zbrzdit simulační cyklus o dobu T_D. Tento princip je přehledně zobrazen na obrázeku 2.



Obrázek 2: Navození R-T běhu simulace



Obrázek 3: Nežádoucí navození R-T běhu simulace

K vytvoření programu byla použita C MEX s-funkce vzhledem k potřebné implementaci C instrukcí k vyčítání systémového času s rozlišitelností 1ms. Algoritmus zpoždění byl vytvořen ve funkci mdlupdate z důvodu spouštění funkce na konci simulačního cyklu a tedy získání nejpřesnější hodnoty s délkou simulačního cyklu Tp. Vývojový algoritmus viz příloha 1.

Odchylka mezi skutečnou délkou simulačního cyklu a zadanou vzorkovací periodou může za normálních podmínek nastat pouze v době ukončování a začátkem nového cyklu. Vykompenzování odchylky dochází vždy v následujícím cyklu tak, aby střední hodnota všech dob průchodů cyklem byla rovna vzorkovací periodě.

Pokud nedochází k vytěžování procesoru je odchylka zanedbatelná a na udržení R-T u vyšších vzorkovacích period (>20ms) nemá žádný vliv. Při spuštění více aplikací může docházet k odebírání procesoru simulinku a tím k nárůstu odchylky okolo 6 až 8ms, nárazově i více. Tato hodnota je uložena a v následujícím simulačním cyklu bude vyrovnána. Jestliže dojde k prodloužení cyklu na dvojnásobek (nebo vyšší) velikosti vzorkovací periody, k vykompenzování bude třeba několik následujících průchodů. Tento případ se může vyskytnout právě u nízkých vzorkovacích period v řádově jednotkách ms a při vytíženém procesoru. Z daného důvodu je doporučená minimální vzorkovací perioda okolo 50ms. Pro docílení nižší hodnoty je nutno použít blok na zvýšení priority programu MATLAB/Simulink pomocí dalšího vytvořeného bloku s názvem priority.

3.1.1 Zvýšení priority programu MATLAB/simulink

Pro docílení navýšení přesnosti R-T bloku je potřeba použít blok s názvem Priority, vývojový algoritmus viz příloha 2. Blok dává možnost změnit prioritu programu MATLAB/simulink v operačním systému windows buď na hodnotu Hight prioritu nebo Real-time prioritu. Nastavení Real-time priority sebou nese jisté omezení při simulaci, ale vzhledem k dosaženým výsledkům je vhodnější. Real-time priorita způsobuje zamrznutí systému windows, pokud se během simulace zasahuje do simulovaného modelu například otevřením bloku scope či jiné. Tento problém byl také jeden z důvodu rozdělení bloku priority a R-T do dvou oddělených bloků. S navýšenou prioritou na real-time je možné nastavit vzorkovací periodu na 6 až 10ms. Rozlišitelnosti C funkce clock je 1ms, ale při této vzorkovací periodě dochází již ke

zkreslení a není zaručena její přesná hodnota, proto není doporučeno nastavovat vzorkovací periodu pod doporučenou mez.

3.1.2 Možné problémy při použití R-T bloku

V simulačním modelu má každý blok určitou prioritu, která udává, kdy dojde k spuštění jednotlivých bloků. V případě bloku pro R-T je ideální případ, jestliže dochází k jeho spuštění až po všech blocích obsažených v simulačním modelu. Vyčtení v jakém pořadí dochází k spouštění bloků je možné v záložce *Format* → *Block displays* → *Sorted order*. Pokud by došlo k spouštění R-T bloku na začátku simulace, vypadal by graf navozující R-T běh simulace tak, jak je znázorněn na obrázku 3. První simulační cyklus je zcela vyplněn časovým čekáním a následující cykly jsou zaneseny odchylkou z vykonávaných funkcí po průběhu algoritmem k nastavení běhu simulace v reálném čase.

Blok R-T pracuje v případě znázorněném na obrázeku 2. Simulink si určuje pořadí spouštění bloků při simulaci tak, že vyjde od bloků, které mají počáteční podmínky a postupně prochází model tak, aby při volání následujícího bloku byly známé všechny jeho vstupy. Blok R-T nemá žádný vstup ani výstup stejně jako blok Priority a jsou tedy spouštěny jako poslední. [11]

Není reálné programově ošetřit pořadí spouštění bloků v simulačním modelu, protože simulink má jistou hierarchii pořadí. Je však možné blokům přiřadit vzájemnou prioritu, podle které je následně poupraveno pořadí. Priorita se nastavuje pomocí položky *Priority* v *Block propetries* (otevírá se z kontextového menu daného bloku). Platí, čím menší číslo, tím vyšší priorita. Tímto způsobem může být docíleno spouštění potřebného bloku (R-T blok) jako posledního.

3.2 WATCH TS, VISUÁLNÍ KONTROLA R-T

Jako aktivní kontrola délky simulačního cyklu je do blocksetu zaimplementován blok Watch TS. Kontrola je realizována na visuálním aspektu. Při udržení vzorkovací periody v reálném čase je barva bloku zelená, v opačném případě červená. Watch TS přijímá dva parametry od uživatele; vzorkovací periodu, která musí odpovídat nastavené vzorkovací periodě v R-T bloku a rozptyl (interval).

Interval vymezuje maximální dovelenou odchylku od zadané vzorkovací periody, která je vyhodnocená jako korektní. Vývojový algoritmus bloku viz příloha 4.

Blok neobsahuje vstupy ani výstupy a je tedy spouštěn implicitně jako jeden z posledních v simulačním modelu. Může dojít k problémům rozebraných v kapitole 3.1.2 Možné problémy při použití R-T bloku. V případě spouštění Watch TS za blokem pro reálný čas vzniká odchylka ±3-6ms od zvolené vzorkovací periody. Je tedy vhodné nastavení priority bloku, při jeho použití.

3.3 MĚŘENÍ VLASTNOSTÍ R-T BLOKU

K měření délky simulačního cyklu byl použit blok zapisující do MATLAB workspace časovou hodnotu na konci každého proběhlého cyklu (vývojový algoritmus bloku měření T_S viz příloha 3). Získané hodnoty byly zpracovány v m-file programu do histogramů a dále spočtená střední hodnota (v ideálním případě rovna vzorkovací periodě), rozptyl a směrodatná odchylka. Měření bylo provedeno s R-T blokem využívaným ve školních laboratořích inteligentních regulátorů, s nově navrženým R-T blokem bez navýšení priority a s novým R-T blokem s navýšenou prioritou na real-time prioritu, pro vzorkovací periody 50ms, 6ms a 0.5ms. Vzorkovací perioda 50ms představuje běžnou vzorkovací periodu, 6ms její doporučenou maximální nejnižší hodnotu a 0.5ms hodnotu, která již nelze v praxi použít.

Výpočet střední hodnoty:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (1)$$

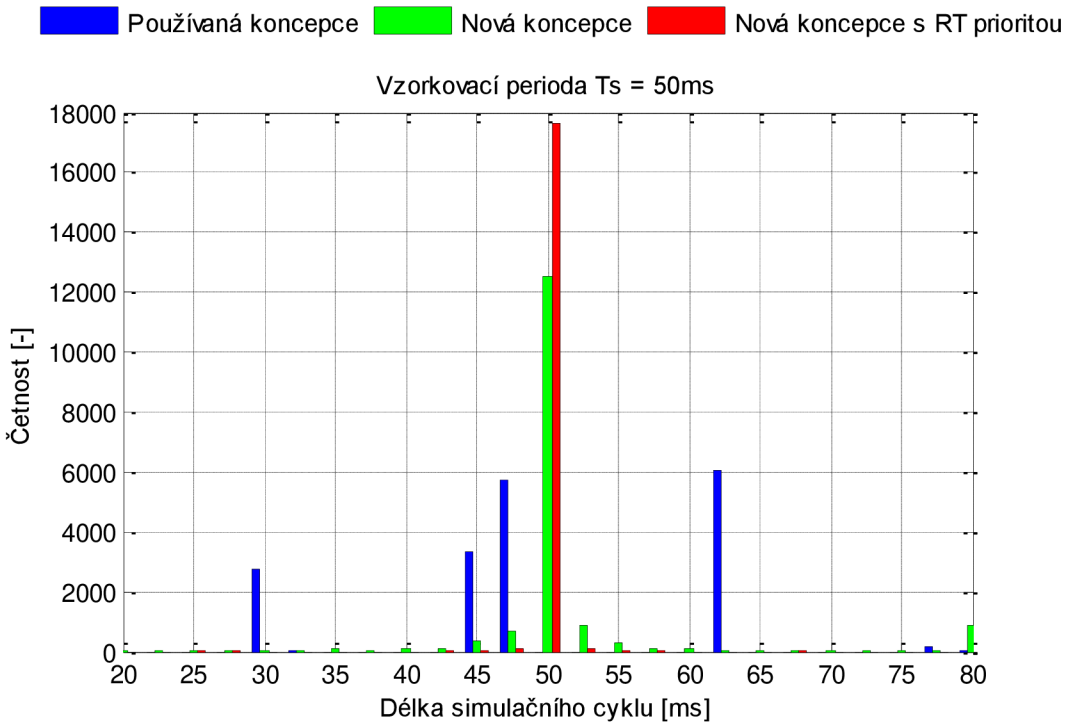
Výpočet rozptylu:

$$D = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \overline{x_i}\right)^2 \qquad (2)$$

Výpočet směrodatné odchylky:

$$s = \left(\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \overline{x_i}\right)^2\right)^{\frac{1}{2}} \qquad (3)$$

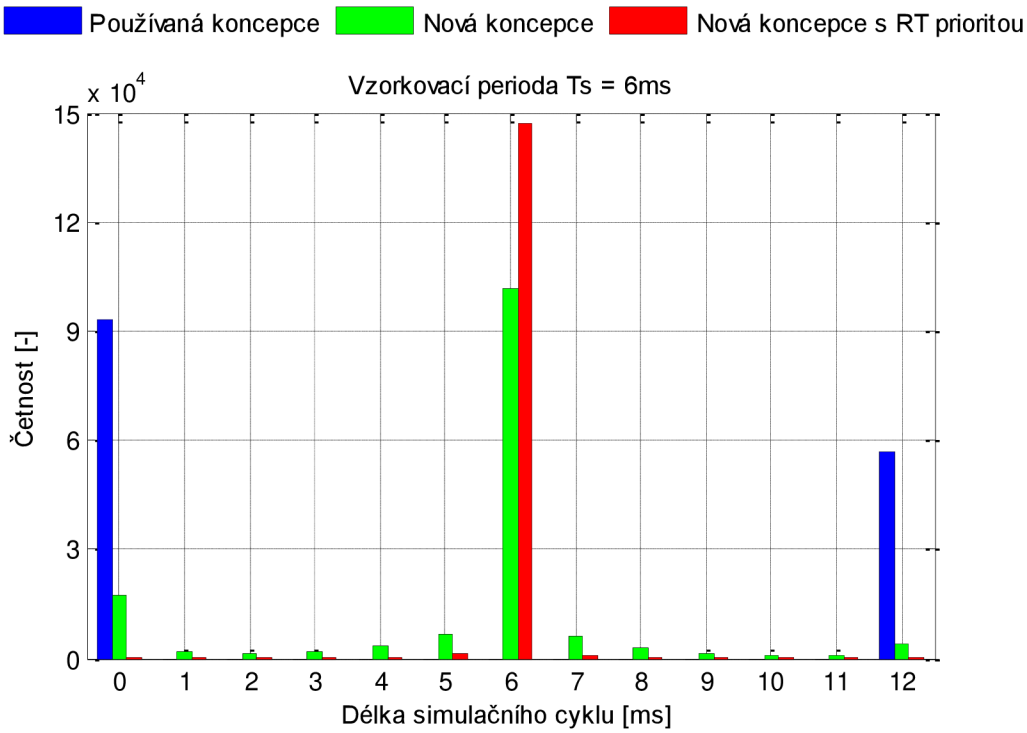
Kde N značí počet prvků a X jednotlivé prvky.



Obrázek 4: Délka simulačního cyklu pro Ts=50ms

Koncepce	Používaná	Nová	Nová s R-T prioritou
Střední hodnota [ms] (1)	50.0002	49.9972	49.998
Rozptyl [ms] (2)	133.6943	609.3337	1.8968
Směrodatná odchylka [ms] (3)	11.5626	24.6847	1.3773

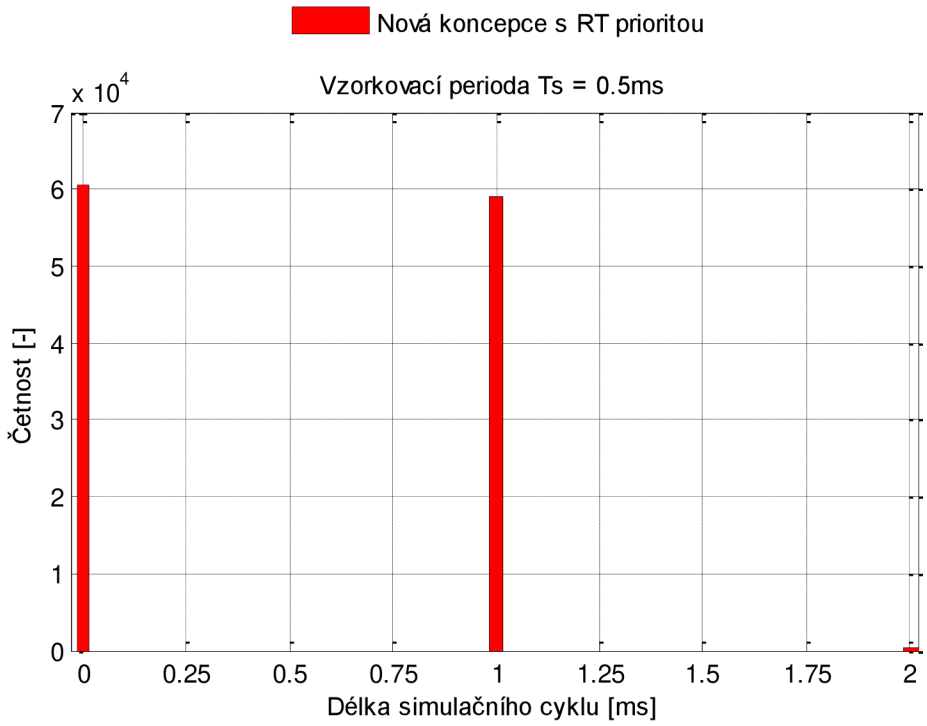
Tabulka 1: Vypočtené údaje pro T_S=50ms v R-T



Obrázek 5: Délka simulačního cyklu pro Ts=6ms

Koncepce	Používaná	Nová	Nová s R-T prioritou
Střední hodnota [ms] (1)	5.9999	5.9996	5.9999
Rozptyl [ms] (2)	77.8710	42.3917	0.3982
Směrodatná odchylka [ms] (3)	8.8245	6.5109	0.6310

Tabulka 2: Vypočtené údaje pro T_S=6ms v R-T



Obrázek 6: Délka simulačního cyklu pro Ts=0.5ms

Koncepce	Nová s R-T prioritou
Střední hodnota [ms] (1)	0.499
Rozptyl [ms] (2)	0.269
Směrodatná odchylka [ms] (3)	0.5186

Tabulka 3: Vypočtené údaje pro T_S=0.5ms v R-T

Koncepce Vratislava Kořínka nepodporuje užití samostatného R-T bloku. Blok je kombinovaný s komunikací pro PLC a je i za tímto účelem navrhnut a ovlivněn. Tento aspekt je viditelný v naměřených údajích, kdy dochází k značnému rozptylu kolem žádané vzorkovací periody, viz obrázek 4.

Vzhledem k udržení reálného času v simulaci odpovídá každé hodnotě nad žádanou vzorkovací periodou ekvivalentní hodnota pod žádanou vzorkovací periodou tak, aby byla udržena střední hodnota nastavené vzorkovací periody.

Měření nově navrhnuté koncepce bylo provedeno se zátěží jak v samotném simulovaném modelu (simulace s PSD regulátorem), tak i se spuštěnými aplikacemi pro zatížení procesoru a tím otestování maximálních mezí. R-T koncepce bez navýšené priority s nastavenou vzorkovací periodou Ts=50ms, viz Obrázek 4 a Tabulka1, vykazuje značný rozptyl, který pomocí bloku na zvýšení priority byl omezen na minimální hodnotu.

U nízkých vzorkovacích period v řádech jednotek a desítek milisekund (bez navýšené priority) dochází k situacím, kdy simulační cyklus projde bez vloženého zpoždění. Je to způsobeno více než dvojnásobnou délkou předešlého simulačního cyklu oproti zadané vzorkovací periodě. K vykompenzování takto vysoké odchylky je potřeba celý následující cyklus. Daný případ je viditelný na Obrázku 5 u nové koncepce bez navýšené priority a u staré koncepce.

Pro úplnost je zobrazena také vzorkovací perioda pod hranicí rozlišitelnosti funkce použité k získávání času viz Obrázek 6. Vzorkovací periody Ts=0.5ms není dosaženo, pouze okolních hodnot, ze kterých se následně skládá její střední hodnota. V praxi je daná vzorkovací perioda nepoužitelná. Doporučená minimální hodnota je 6-10ms z důvodů popsaných v kapitole 3.1.1 Blok zajišťující zvýšení priority simulace.

3.4 SHRNUTÍ

K simulování modelu v reálném čase byly vytvořeny celkem 3 bloky. Samotný real-time blok navozující běh simulace v reálném čase, blok priority pro zlepšení držení vzorkovací periody a blok Watch Ts hlídající přesnost vzorkovací periody s rozptylem, který si uživatel zvolí. Základní minimální vzorkovací perioda je doporučena na velikost Ts=50ms vzhledem k poměru velikosti rozptylu a velikosti nastavené vzorkovací periody. Pro dosažení nižší hodnoty je potřeba použít blok priority, kterým lze nastavit na hight prioritu nebo real time prioritu program MATLAB/simulink. S real time prioritou je dosaženo lepších výsledků a je možná minimální vzorkovací perioda až Ts=6ms. Real time priorita sebou nese ale nevýhodu v podobě omezené flexibility při simulaci. Během simulace není možné zasahovat do simulovaného modelu, například otevření scope a podobně. V opačném

případě dojde k zamrznutí sytému windows a je nutný restart (otestováno na jednojádrovém procesoru). Blok Watch Ts je vhodné používat ke kontrole vzorkovací periody, ale může docházet ke snížení její přesnosti o 3-6ms z důvodů popsaných v kapitole 3.2 Watch TS, kde je také rozebráno jak problém vyřešit.

Koncepce	Nová bez priority	Nová s prioritou
Maximální Ts	>50ms	≥6ms

Tabulka 4: Minimální vzorkovací perioda pro R-T blok

4. SÉRIOVÁ KOMUNIKACE RS232 V SIMULINKU

4.1 POPIS SÉRIOVÉ KOMUNIKACE RS-232

Standard RS-232 se používá jako komunikační rozhraní počítače a dalších přístrojů. Přenos dat je uskutečněn po jednotlivých bitech za sebou obdobně jako u rozhraní USB či ethernet. V současné době se již upustilo od používání dané komunikace a je nahrazována již zmíněnou komunikací pomocí USB, ale toto neplatí v průmyslu.

RS 232 používá dvě napěťové úrovně logickou 1 a logickou 0, kde logická 1 představuje nižší úroveň (záporné napětí) a logická 0 vyšší úroveň (kladné napětí). Přenos dat může probíhat buď synchronně, nebo asynchronně. Synchronní přenos dat probíhá nastavením na vodiči jednu ze dvou úrovní reprezentující přenášenou informaci a validita informace se potvrdí změnou úrovně signálu nebo impulsem na synchronizačním vodiči. U asynchronní komunikace synchronizační vodič odpadá. Synchronizace se provede při startu komunikace, kdy vysílací strana odešle určitá data, po kterých se přijímací strana zesynchronizuje. [8]

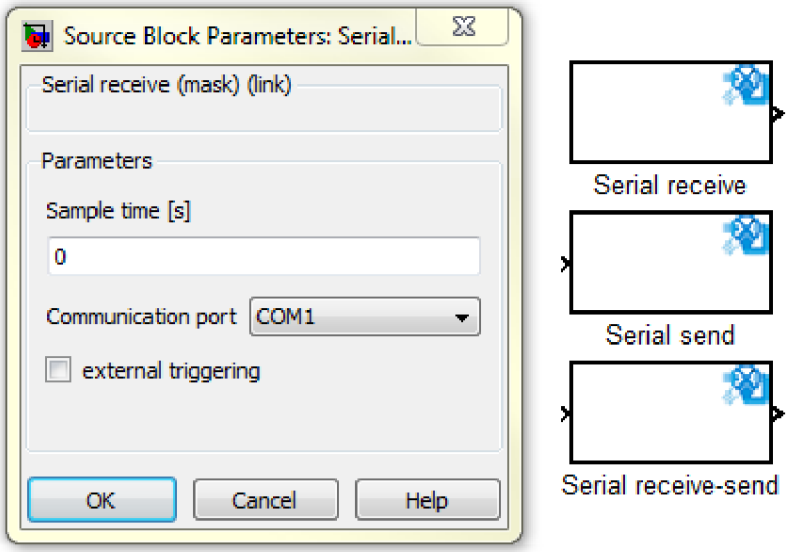
4.2 REALIZACE SÉRIOVÉ KOMUNIKACE

Pro sériovou komunikaci po RS232 byla v simulinku vytvořena jednoduchá podpora. MATLAB nabízí funkce na vytváření a obsluhu sériové komunikace. Pro jejich využití v simulinku bylo použito Level2 m-file s-funkci, která podporuje psaní matlab M kódu. Vytvoření objektu pro správu sériové linky je možné pomocí funkce serial(‘port’), která jako vstupní parametr přijímá port určený ke komunikaci. Objekt zpřístupní nastavení komunikace, rychlost přenosu, počet start/stop bitů, délky datového slova a velikost timeoutu v případě výpadku komunikace. Pro komunikaci se zařízením je nutné nejprve provázat spojení pomocí funkce fopen(‘objekt’). Jako indikace správného propojení je port zobrazen jako otevřený, v opačném případě je vrácena chyba, pokud je pokus o zápis nebo o čtení. K jednomu sériovému portu je možné připojit pouze jeden sériový objekt, při pokusu o propojení více objektů

s jedním sériovým portem je opět vrácena chyba. Na konci komunikace je nutné odpojení objektu od sériového portu, jinak není možné zahájit přes daný port nové spojení. Důležitou funkcí je instrfind, která vrací stav všech používaných sériových portů a umožňuje jejich uzavření.

Vytvořená podpora pro komunikaci po sériové lince RS 232 v simulinku obsahuje tři bloky (pro příjem, vysílání a blok obsahující spojení dvou předešlých). Algoritmus k vytvoření objektu sériové komunikace a jeho nastavení je implementován v s-funkci ve funkci mdlstart (provede se pouze na začátku simulace). Samotný algoritmus pro příjem a vysílání dat je ve funkci mdloutputs, pro docílení zpracování přijatých dat a jejich odeslání v jednom simulačním cyklu. Vývojový algoritmus viz příloha 5.

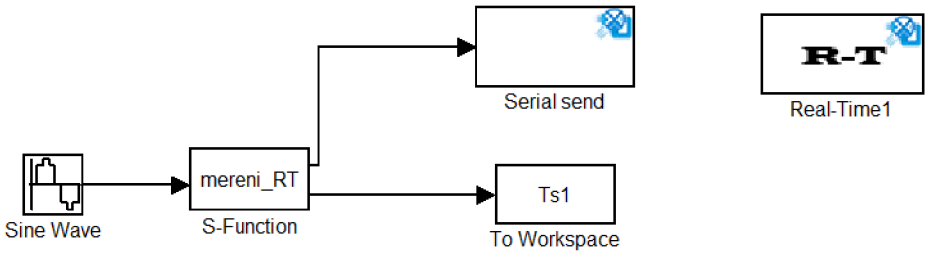
Nastavení sériové komunikace proběhlo podle standardu, přenosová rychlost 9600 baud což při hodnotě 1baud=1b se rovná rychlosti 1,2kb/s. Délka datového slova odpovídá 8bitům a je ukončen jedním stop bitem. Timeout neboli maximální doba čekání na čtení nebo zápis dat je nadefinovaná na hodnotu 20s nebo 0s (nastavení volí uživatel). Je-li zvolen timeout=20s (aktivní external triggering), dochází k navození vzorkovací periody v simulačním modelu s přijímacím blokem podle vysílací části za předpokladu $T_S < 20s$.



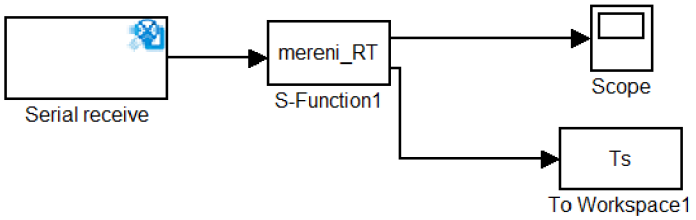
Obrázek 7: Nastavení bloku pro příjem dat ze sériové linky

4.3 MĚŘENÍ VZORKOVACÍ PERIODY S BLOKY PRO SÉRIOVOU KOMUNIKACI

Pro odzkoušení sériové komunikace, možnosti udržení reálného času a nastavení vzorkovací periody v simulačním modelu s přijímacím blokem přes external triggering bylo použito zapojení na Obrázku 8 a 9. Modely byly propojeny přes rozhraní RS232 pomocí USB konvertorů.

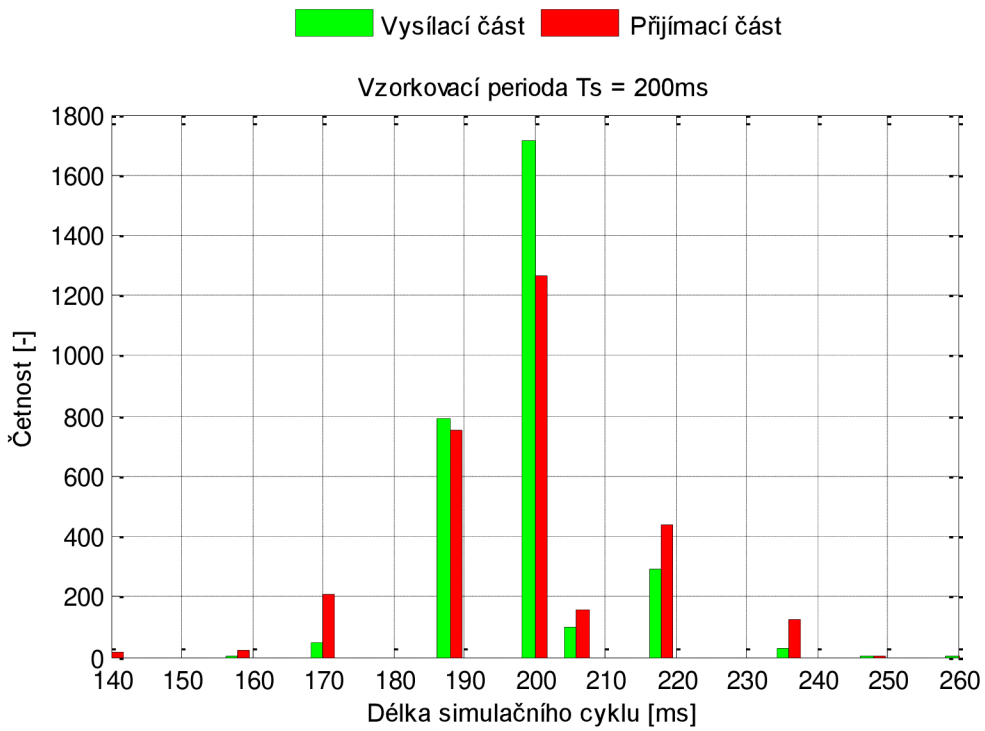


Obrázek 8: Zapojení s blokem pro sériové odesílání dat



Obrázek 9: Zapojení s blokem pro sériový příjem dat

Vzorkovací perioda Ts=200ms byla udržena v obou simulačních modelech s výraznějším rozptylem způsobeným nepoužitím bloku na zvýšení priority programu, viz Obrázek 10. Jistá odchylka je zanesena také samotným blokem na měření vzorkovací periody, protože není spouštěn na samotném konci simulačního cyklu, kde je hodnota vzorkovací periody nejpřesnější. Zcela přesné měření není možné docílit programově, ale jako ukázka možností bloků je dostačující.



Obrázek 10: Délka simulačního cyklu při použití sériové komunikace

	Vysílání s R-T	Příjem
Střední hodnota [ms] (1)	200.0093	200.0093
Rozptyl [ms] (2)	132.7951	258.7878
Směrodatná odchylka [ms] (3)	11.5237	16.0869

Tabulka 5: Vypočtené údaje pro T_S=200ms při použití sériové komunikace

4.4 SHRNUTÍ

Pro simulink je implementována jednoduchá podpora sériové komunikace přes RS 232 rozhraní v podobě tří bloků. Vnitřní strukturu bloku tvoří level 2 m-file s-funkce, která podporuje využití funkcí nabízené MATLABem k obsluze sériové linky. Podpora byla otestována ve dvou simulačních modelech propojených přes USB rozhraní s konvertorem na rozhraní RS 232.

Blok na příjem dat byl nastaven na timeout o velikosti 20s, neboli aktivní external triggering, pro odzkoušení přenosu R-T běhu simulace z vysílacího modelu (viz obrázek 10). Maximální vzorkovací perioda pro koordinaci modelů na jednotnou vzorkovací periodu je rovna hodnotě nastaveného timeoutu. Pokud bude external triggering vypnut, rychlost simulace nebude ovlivněna a přijímací blok odešle na výstup vždy naposledy přijatá data.

5. KOMUNIKACE S PLC FIRMY B&R

5.1 ZÁKLADNÍ INFORMACE O PVI

Zdroj [12]

PVI (Process Visualization Interface) je hlavním nástrojem pro přístup k programovatelnému automatu (PLC) od společnosti B&R prostřednictvím systému Windows 95 až Windows 7. PVI systém se skládá ze čtyř základních částí; PVI manažer, PVI monitor, PVICOM (klientské rozhraní) a PVI lines.

PVI manažer je hlavní složkou PVI, odpovídá za řízení všech typů procesů od nejjednodušších až k nejsložitějším. Je odpovědný za správný směr toku dat a jejich načasování.

PVI monitor je nástroj pro zobrazení provozního stavu komunikace. Lze s jeho pomocí sledovat provázání s jednotlivými programovatelnými automaty a dále vyčítat objekty použité k vytvoření komunikace.

PVICOM představuje rozhraní na nejnižší úrovni pro komunikaci mezi klientem a PVI manažerem. Jedná se o výkonnostně optimální rozhraní.

Poslední částí je PVI line (linkování), jehož úkolem je připojení PVI objektů na objekty, které jsou mimo PVI, neboli vytvořit vzájemné provázání.

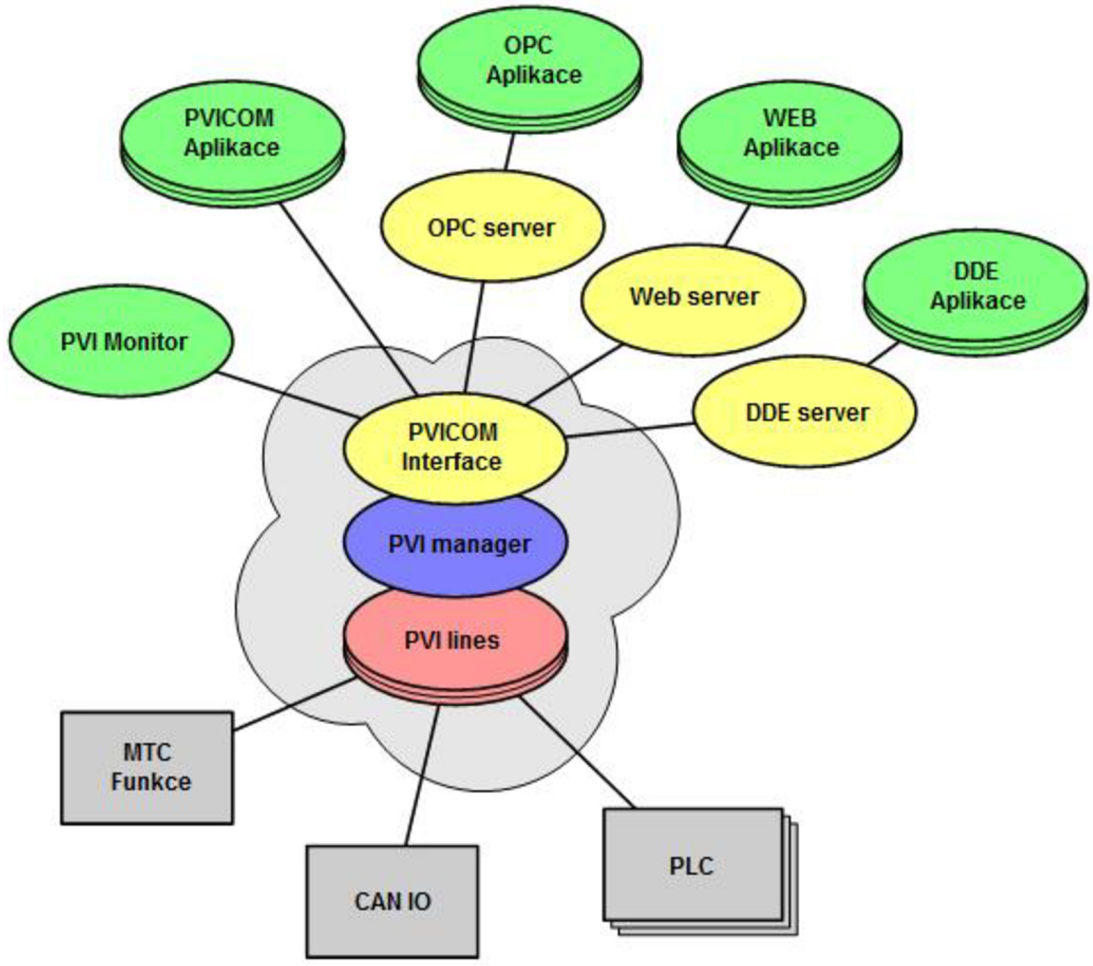
K vytvoření a hlídání komunikace pod PVI je možné použití dvou knihoven nabízených společností B&R. Jedná se o PVICOM.dll pro programovací jazyk C a BR.AN.PviServices.dll, což je nadstavba pro .NET programovací jazyky. Vybrána byla knihovna PVICOM.dll pro podporu programovacího jazyka na nižší úrovni (nižší úroveň přístupu k hardware) a podstatnou výhodou je podpora s-funkcí C/C++ jazyk, není tedy potřeba přídavných souborů. Dále v práci je tedy zabýváno pouze knihovnou PVICOM.dll

5.1.1 Přehled systému PVI

PVI struktura se skládá z řady variabilních objektů (proměnné v PLC) obsahujících údaje aktualizované buď cyklicky, nebo v určitých časových intervalech. Variabilní objekty jsou speciálními typy procesních objektů. Ostatní procesní objekty jsou používány pro stahování / nahrávání, čtení PLC informací

a podobně. Celkově jsou procesní objekty rozděleny do několika typů objektů. Každý typ objektu představuje určitou logickou nebo fyzickou část komunikačního spojení s PLC. Procesní objekty jsou spravovány ve vnitřní struktuře PVI manažeru. Propojení procesních objektů s externími objekty (objekty v PLC) se provede pomocí PVI line, tím dojde k vytvoření komunikace mezi stanicí (s PVI manažerem) a PLC.

Rozhraní PVICOM pracuje na principu klient/server. PVI manažer zde představuje server a PVICOM aplikace jsou klienti. Servery a klienti mohou být umístěny na jedné stanici, v tom případě se jedná o lokální komunikaci, nebo mohou být umístěny odděleně, v tom případě se jedná o vzdálenou komunikaci.



Obrázek 11: PVI základní uspořádání [12]

PVI Manažer:

PVI manažer, neboli správce, je odpovědný za správný směr toku dat a jejich načasování. To znamená, že stanoví, zda budou data odeslána z aplikace do stanoviště, či obráceně ze stanoviště do aplikace.

Správce musí být spuštěn při každém pokusu o navázání spojení s PLC. V knihovně PVICOM.dll je možné použití dvou funkcí k jeho spuštění, přičemž automaticky je spuštěn pouze při vytváření místní komunikace (PviInitialize nebo PviXInitialize). Pokud je třeba vytvoření vzdálené komunikace, musí dojít k jeho spuštění ručně.

PVI Monitor:

Slouží pro zobrazení provozních informací, konfiguraci diagnostických funkcí a globální nastavení PVI manažeru. Komunikace monitoru a manažeru probíhá na principu klient/server jako u běžné komunikace s PVICOM aplikací.

Pomocí PVI monitor je možné spustit funkci *PVI SnapShot viewer*, která zobrazí informace o objektech ve strukturovaném stromovém zobrazení nebo ve dvou seznamech. Kromě názvu objektu je možné ze seznamu vyčíst objektové ID (identifikace), stav objektu a popis objektu PVI. Tyto údaje jsou nezbytné při samotném vytváření komunikace mezi počítačovou stanicí a PLC. Stav objektu je možné rozlišit pomocí barevného zobrazení *Options → Display Colorad objects*. Zelená barva označuje aktivní objekt, modrá neaktivní objekt, červená aktivní objekt s error stavem a fialová neaktivní objekt s error stavem.

5.2 PVICOM ROZHRANÍ (UŽIVATELSKÉ ROZHRANÍ)

Zdroj [12]

Připojení k PVI manažeru:

Pro komunikaci s PVI manažerem je potřeba nejprve nastavit komunikační instanci. Tato instance vytváří spojení mezi aplikací a PVI manažerem. Pro vytvoření komunikace je možné užití dvou funkcí, jak již bylo zmíněno PviInitialize a PviXInitialize. Funkce PviXInitialize je možné na rozdíl od PviInitialize volat více než jednou. Každá výzva vytváří novou komunikační instanci a dovoluje nám

komunikovat s více PVI manažery na několika počítačích. Je-li komunikační instance nastavena jako PviXInitialize, pak všechny PVICOM funkce musí být volány s handle PviX. Při ukončení komunikační instance je potřeba její uvolnění pomocí funkce PviDeinitialize respektive PviXDeinitialize.

Monitorování komunikace:

Pro sledování komunikace se využívají funkce globální události. Lze je nastavit pro monitorování komunikačních spojení s PVI manažerem. Pokud dojde k ukončení PVI manažeru nebo k selhání komunikační sítě, dochází ke ztrátě všech provázaných objektů i dočasných procesních objektů v PVI manažeru. Po restartu PVI manažeru nebo po obnově komunikačního spojení musí aplikace nastavit dané objekty znovu. Globální události oznámí, kdy je třeba opětovné nastavení těchto objektů. Globální události se nastavují pomocí funkcí z knihovny PVICOM PviSetGlobalEventMsg nebo PviXSetGlobalEventMsg.

5.2.1 Nastavení objektové struktury

Přístup k údajům proměnné obsažené v PLC, nebo vykonávání speciálních služeb, jako je zjišťování stavu CPU a podobně, jsou prováděny prostřednictvím objektové struktury v PVI manažeru. Objektová struktura se skládá z jednotlivých procesních objektů uložených ve stromové struktuře. Při vytváření této struktury je možné libovolné pojmenování procesních objektů.

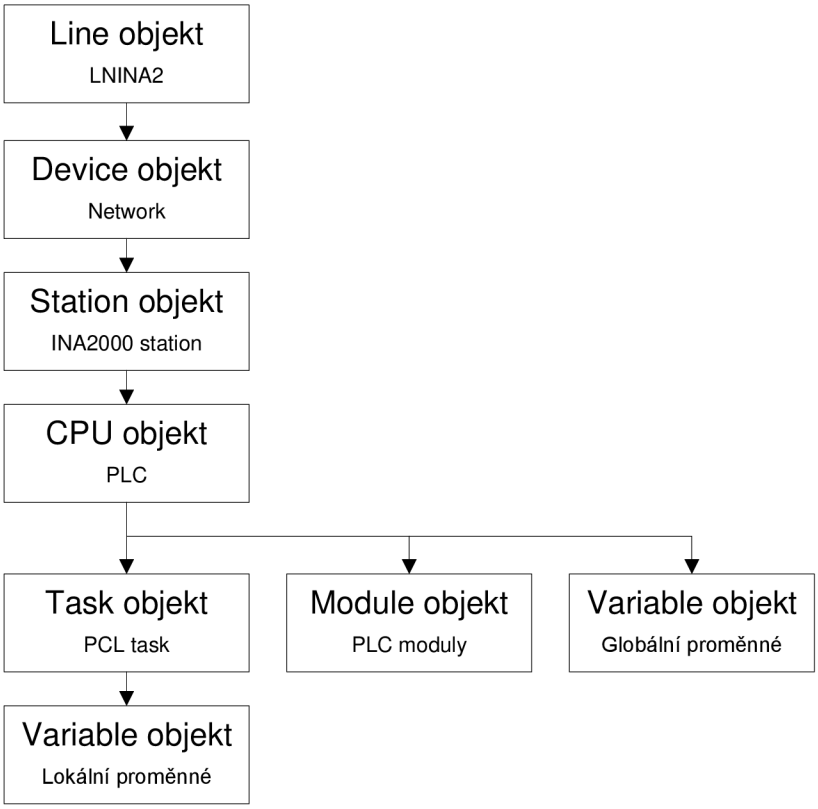
Existuje několik typů procesních objekt; base objekt, line objekt, device objekt, station objekt, CPU objekt, module objekt, task objekt a variable objekt. Objektová struktura se skládá z jednotlivých výše zmíněných objektů, jejich provázání zajišťuje vytvoření komunikace. Každý typ objektu má ve stromové struktuře určitou úlohu. Base objekt je na nejvyšší úrovni stromové struktury procesních objektů. Tento objekt je vždy implicitně vytvořen a nemusí být vytvářen uživatelem. Line objekt stanovuje, kdy bude PVI linka používána. Procesní objekty typu device, station a CPU se obvykle používají k vytvoření komunikace s PLC a k adresování PLC v síti. Objekty module, task a variable představují souhlasné objekty s objekty v PLC, přičemž task označuje úkol nebo proces a variable označuje proměnnou ve stanici.

Kromě jména procesních objektů se definuje také jejich popis (description), který udává vlastnosti objektů. Popis je definován jako řetězec ASCII znaků ukončený 0 (null). Procesní objekty se vytvářejí funkcí PviCreate/PviXCreate.

Výběr PVI linky pro komunikaci s PLC:

Základním úkolem PVI linky je propojit PVI objekty s objekty mimo PVI. Linka je také zodpovědná za komunikaci s PLC a určuje, jaký komunikační protokol bude použit. V laboratoři inteligentních regulátorů jsou PLC s PC propojeny pomocí ethernetu. Komunikační protokol TCP/IP podporuje protokol INA2000.

Protokol INA2000 je standardní PVI linka podporující služby stahování/nahrávání, vyčtení/zápis dat do respektive z proměnných v PLC a online obsluhu PLC. Komunikace může probíhat přes sériové rozhraní RS232 nebo RS422, modem, CAN, ethernet a profibus síť. Podle zvoleného komunikačního protokolu se dále nastavují popisy objektů tvořících objektovou strukturu pro navázání spojení s požadovaným PLC.



Obrázek 12: Uspořádání procesních objektů INA2000 [12]

Statické (Static) a dočasné (temporary) procesní objekty:

Procesní objekty můžou být vytvořeny dvojím způsobem, buď jako statické, nebo dočasné. Oba varianty pracují na stejném principu, ale rozdílné je jejich vytvoření a ukončení. Statické procesní objekty jsou vytvořeny a zůstávají aktivní po celou dobu běhu PVI manažeru, i při ukončení PVICOM aplikace. Při nové inicializaci objektu dojde k chybové hlášce 12002 (objekt se stejným jménem je již vytvořen). Při vytvoření statických objektů není možné komunikovat s jedním PLC objektem z více aplikací. Úplné odstranění statického objektu je možné pomocí funkce PviDelete/PviXDelete.

Na rozdíl od statických procesních objektů, u dočasných není hlášena chyba při vytvoření několika procesních objektů se stejným názvem, ale dochází k vytvoření odkazu. Procesní objekt je aktivní, pokud je na něj alespoň jeden odkaz. V opačném případě je odstraněn.

5.3 PVICOM FUNKCE

Zdroj [12]

Funkce nabízené knihovnou PVICOM.dll je možné použít s nebo bez instance handle PviX, ale nelze je v jedné vytvářené komunikaci kombinovat. Rozdíl mezi funkcemi je uveden v kapitole 5.2 PVICOM rozhraní, připojení k PVI manažeru. Při vytváření komunikace s více PVI manažery (použití PviX) dochází k nárůstu doby přenášení dat oproti komunikaci s jedním PVI manažerem.

Návratová hodnota funkcí je 0 v případě korektního provedení, v opačném případě je vrácena chybová hodnota. Pro vyčtení významu chyby je nutno použít help programu Automation studio. Z hlediska kvantity chybových hlášení je nelze zde uvést. Data vyčtená z PLC jsou získána přes odkaz v příslušném parametru funkce. Přijímané parametry jednotlivých funkcí jsou proměnné a je nutné důkladné nastudování.

Asynchronní a synchronní funkce:

Asynchronní funkce se vyznačují doplněním funkce o slovo *Response* (odpověď) nebo *Request* (dotaz) příklad: PviRequestWrite a PviResponseWrite. Během zpracovávání dotazu je možné odeslat další dotaz nebo vykonávat jiné úlohy.

Synchronní funkce zasílají dotaz a čekají na odpověď, nelze tedy zasílat více dotazů během jejich zpracování. Použití v PVICOM aplikaci je podstatně jednoduší než u asynchronních funkcí, protože program se nemusí starat o uživatelské zprávy a přiřazovat žádosti nebo odpovědi. Obecně je možné v aplikaci kombinovat synchronní i asynchronní funkce podle potřeby.

5.4 REALIZACE KOMUNIKACE S PLC

Jako podpora komunikace mezi simulinkem a PLC jsou k dispozici dva bloky; blok zapisující do PLC (B&R send) a blok pro čtení z PLC (B&R receive). K realizaci bloků byla použita C MEX s-funkce z důvodu potřeby implementace C knihovny PVICOM.dll. Zjednodušené vývojové diagramy bloku B&R receive a B&R send viz příloha 6 respektive 7.

Při vytváření bloků byl kladen důraz na snadnou manipulovatelnost a snížení počtů parametrů zadávaných uživatelem na minimum. Jsou k dispozici dvě možná řešení provázání bloků s PLC. Hlavní s možností nastavení PLC jména, IP adresu (DAIP), port cílové stanice (REPO), jméno úlohy nahrané v PLC a jméno proměnné, se kterou je třeba navázat spojení. V případě bloku B&R receive je třeba zadat i velikost pole vyčítané proměnné. Druhou možností je manuální nastavení podporující nastavení celé objektové struktury. Pro snadnější použití jsou parametry předdefinovány. Datový typ a velikost pole proměnné jsou vyčítány z PLC a nastavovány programově.

5.4.1 Nastavení procesních objektů, INA2000

Komunikace mezi PLC a PC probíhá na TCP/IP protokolu a je tedy zvoleno síťové rozhraní INA2000 podporující ethernet. Objekty potřebné k realizaci spojení jsou zobrazeny na obrázek 12. Line a device objekt definují typ síťového rozhraní, komunikační protokol a číslo zdrojové stanice (SA). SA je implicitně nastaveno na hodnotu 1. V případě komunikace z více PC s jedním PLC současně je potřeba každému PC přidělit jinou hodnotu SA (manuální nastavení), která nebude v rozporu s cílovým identifikačním číslem PLC (DA). Station a CPU objekt určují přístup k PLC v síti a nastavují parametry spojení. Přístup je možný dvěma způsoby, pomocí DA nebo DAIP. Je potřeba také nastavit REPO na hodnotu 11159 v případě reálného

PLC nebo 11160 pro virtuální PLC. Objekty Task a Variable definují přístup k proměnným. Tyto hodnoty jsou variabilní, záleží na nahraném programu v PLC.

5.4.2 Realizace bloku

Funkce používané z knihovny PVICOM.dll jsou nastaveny na lokální komunikaci s PVI manažerem, přičemž komunikace probíhá vždy synchronně. Asynchronní přenos dat není možný z důvodu absence systému předávání zpráv užívaný u windows aplikací. Objekty tvořící stromovou strukturu jsou vytvářeny jako dočasné procesní objekty, pomocí handle na s-funkci předávané jako jeden z parametrů funkci PviCreate. Handle každého bloku je ojedinělý, pokud proměnná vázající hodnotu handle není deklarována jako globální (static). Statická proměnná je vždy jedna a sdílená pro všechny instance s-funkce. Pro uchování hodnot v s-funkci se používají work vektory, které jsou pro každou instanci s-funkce jedinečné.

Vytvoření komunikace není jediným řešeným problémem. S vyčtením proměnné z PLC souvisí také správné určení datového typu a pokud se jedná o pole i velikost pole. Je žádané uživatele oprostit o co nejvíce zadávaných parametrů, a tím vzrůstají nároky na programové řešení.

Při implementaci bloku do simulačního modelu dochází k jeho inicializaci, ve které se mimo jiné nastaví počet a rozměr vstupních/výstupních portů. Během simulace, kdy lze programově vyčíst velikost pole proměnné v PLC, rozměr portu nelze měnit a je tedy potřeba, aby byl zadán uživatelem před spuštěním simulace. Tento problém se týká pouze bloku pro příjem dat z PLC. Blok pro zápis do PLC nastavuje velikost vstupního portu automaticky podle předchozího bloku v simulačním modelu pomocí funkce *DYNAMICALLY_SIZED*.

Po spuštění simulace dochází k načtení dat z masky bloku s-funkcí a jejich zpracování. Proběhne rozdělení, zda získaná data jsou z masky hlavní nebo z manuálního nastavení a podle toho jsou přiděleny názvy procesních objektů a jejich popisy proměnným. K vytvoření procesního objektu tvořícího objektovou strukturu se používá funkce PviCreate přijímající mimo jiné název objektu, typ objektu, popis objektu a uživatelskou zprávu pro propojení s dočasným objektem v našem případě handle na s-funkci. Pokud je uživatelská zpráva rovna nule, je vytvořen statický procesní objekt. V opačném případě je vytvořen dočasný procesní objekt. Každý

procesní objekt má přístupové ID (identifikační číslo) získané při jeho vytvoření. Přes parametr ID je možné vyčítat data o procesním objektu a tedy i datový typ proměnné, velikost pole, strukturu apod. Informace o procesním objektu je získávána jako řetězec. Po nastavení všech procesních objektů je potřeba jejich provázání s objekty mimo PVI (v PLC) pomocí funkce PviLink. Provázáním objektů dochází k vytvoření komunikačního spojení.

V simulinku komunikují bloky mezi sebou implicitně v datovém formátu double, pokud není přetypován. I když je double nejvyšším datovým typem nelze jej použít přímo pro zápis hodnoty vyčtené z PLC funkcí PviRead. PviRead očekává stejný datový typ pro zápis, jakého je typu vyčítaná proměnná z PLC. V případně zápisu ze simulinku do PLC je třeba funkci PviWrite přiřadit parametr v datovém typu odpovídajícím v PLC. Datový typ, jak již bylo zmíněno, je možné vyčíst přímo z PLC, ale jazyk C nepodporuje deklaraci proměnných v průběhu funkce, pouze na jejím začátku. Dopředu není známo, jaký datový typ bude třeba použít, proto byla vytvořena UNIE podporující základní datové typy v PLC (double, float, int, short, char, unsigned int, unsigned short, unsigned char, BOOL) a dle potřeby jsou přiřazovány.

Dalším dopředu neznámým parametrem je velikost pole. Velikost pole je nastavováno dynamickou alokací paměti pomocí funkce malloc, jakmile je nastaven datový typ. V bloku pro příjem dat z PLC je potřeba zadání velikosti pole uživatelem, ale pouze pro nastavení velikosti portu na výstupu bloku a to z důvodu nemožnosti změny tohoto údaje během simulace. Velikosti pole jsou vzájemně porovnávána, zda nedochází k pokusu zapsání větší proměnné na výstup bloku, než je velikost portu, nebo zapsání větší proměnné ze simulinku do menší proměnné v PLC. V případě nesrovnalosti je vypsána varující hláška v *command window* matlabu (pokud se nejedná o ztrátu dat) nebo zastavení simulace s chybovou hláškou (v případě možnosti ztráty dat).

Ukončení simulace provede odstranění dočasných procesních objektů a dojde k ukončení komunikace s PLC. Všechny funkce z PVICOM knihovny jsou hlídány a při výskytu chyby je simulace ukončena s chybovou hláškou ve které části programu

se chyba vyskytla. Chybové hlášení obsahuje číslo chyby, které je popsáno v help B&R automation studiu [12].

5.4.3 Maska bloku

Pro snadné nastavení bloku byla vytvořená maska s dvěma možnostmi nastavení. Zjednodušená verze typu „main“, kde je řada parametrů vnitřně přednastavených a uživatel musí zadat pouze vzorkovací periodu, název PLC, IP adresu, název úlohy v PLC, název proměnné, velikost pole a číslo komunikačního portu. Základní použití je také popsáno v helpu, který je dostupný z masky bloku.

Druhá varianta nastavení je typu „manual set“, kde uživatel dostává možnost nakonfigurovat nastavení všech procesních objektů dle vlastních požadavků.

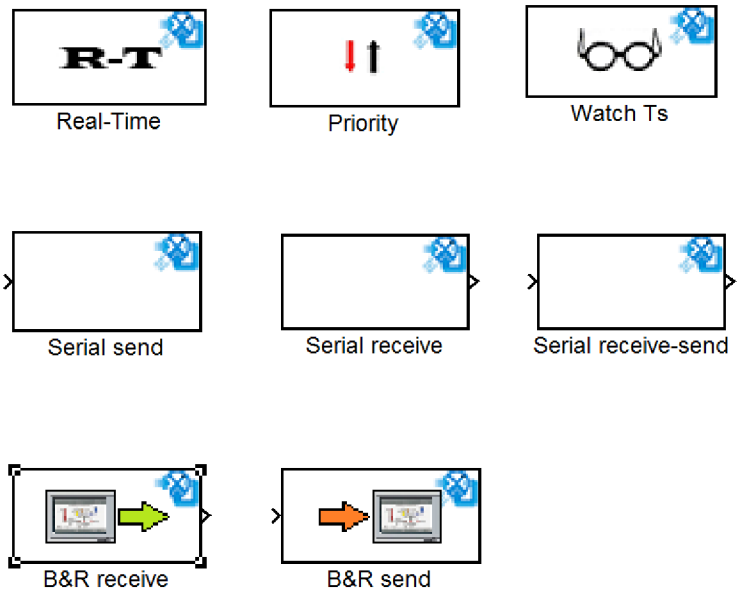
6. VYTVOŘENÍ BLOCKSETU

Vytvořené bloky jsou implementovány pro snadnou manipulaci v blocksetu nazvaným VUT toolbox, který je možné najít v simulink library browser (simulink knihovna) pod stejnojmenným názvem.

Pro implementování blocksetu do knihovny byl vytvořen jednoduchý m-file s názvem *install.m* a pro odstranění knihovny *uninstall.m*. Spuštěním souboru install dojde k vyhledání kořenového adresáře MATLABu a v něm složky toolbox, která obsahuje veškeré knihovní prvky simulinku. V adresáři toolbox je vytvořena složka se jménem VUT_library, pro přesunutí souborů obsažených ve složce s *install.m*. Dále je nadefinována a uložena cesta k nově vytvořené knihovně v programu MATLAB.

Název blocksetu v simulink knihovně a další parametry jsou nastaveny v konfiguračním souboru *slblock*, který je nedílnou součástí každého adresáře obsahující knihovní prvky. Název *slblock* je neměnný a musí být dodržen, tímto je zamezeno vytvoření více knihoven v jednom adresáři. [11]

Pokud implementace toolboxu proběhne v pořádku je vypsána v MATLAB okně hláška s úspěšným provedením operace. Pokud je již otevřen simulink library browser je nutné jeho zavření a opětovné otevření k načtení knihovny.



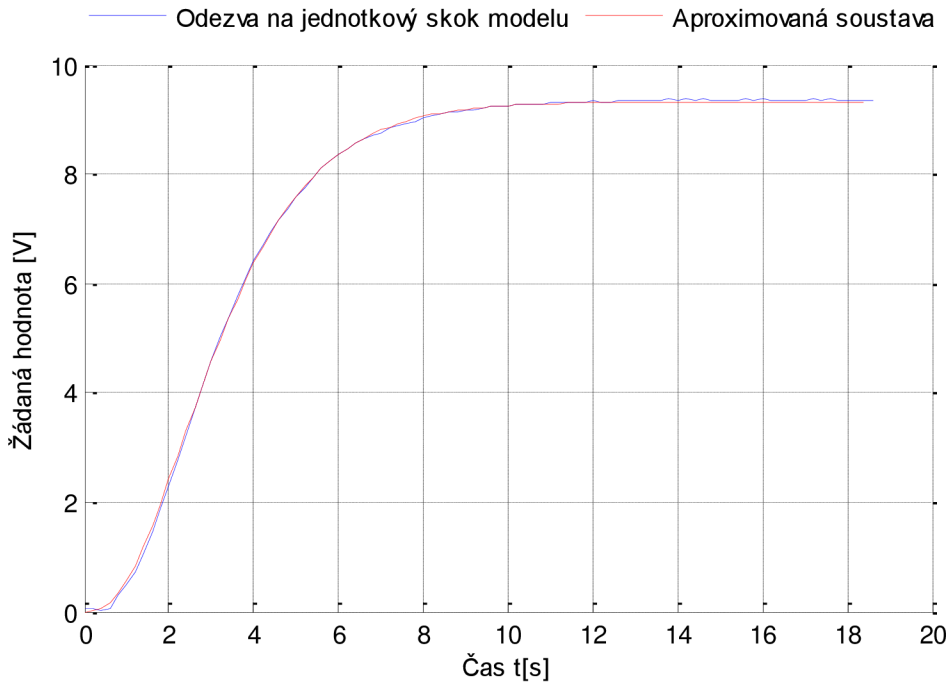
Obrázek 13: Knihovní prvky VUT toolbox

7. VYUŽITÍ BLOCKSETU K ŘÍZENÍ FYZIKÁLNÍCH MODELŮ

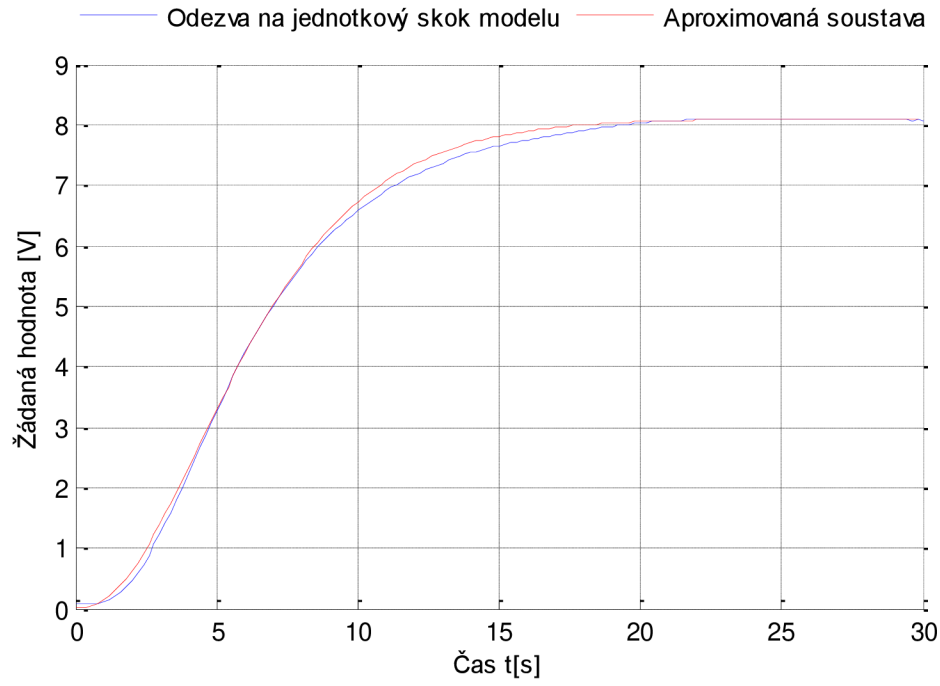
Odzkoušení funkčnosti komunikace s více PLC současně byla provedena na regulaci tří modelů; měření otáček ventilátoru s ventilátorem jako akčním členem, vyregulování aproximované soustavy ventilátoru na žádanou hodnotu a regulace setrvačného článku třetího řádu na žádané napětí. Zapojení bylo zvoleno ve třech koncepcích a s třemi různými typy regulace. Pro regulaci modelu s ventilátorem bylo použito PSD regulátoru, regulace aproximovaného modelu ventilátoru S-PD regulátor a regulace setrvačného článku FEEDFORWARD regulátor.

7.1 APROXIMACE SOUSTAV A NÁVRH REGULÁTORŮ PRO SIMULACI

Aproximací rozumíme nahrazení přesné charakteristiky charakteristikou matematicky popsatelnou. K získání aproximované soustavy bylo třeba použití přechodové charakteristiky, viz obrázek 13 a obrázek 14. Určení přesného typu a řádu aproximované soustavy je nutno znát velikost inflexního bodu, dobu průtahu T_U a dobu náběhu T_N. Soustavu druhého řádu je vhodné použít pro inflexní bod do velikosti 0.26. Jestliže nabývá vyšší velikosti, je třeba použít soustavu třetího nebo vyššího řádu. [3], [5]



Obrázek 14: Přechodová charakteristika modelu k měření otáček ventilátoru



Obrázek 15: Přechodová charakteristika setrvačného článku třetího řádu

7.1.1 Návrh PSD a S-PD regulátoru

Z průběhu přechodové charakteristiky ventilátoru, viz obrázek 13, byla odečtena hodnota pro inflexní bod i=3.5, doba průtahu T_U=0.9 a doba náběhu T_N=4.2. Z hodnoty získané jako poměr doby průtahu a doby náběhu získáme pomocí zdroje [3] řád soustavy.

$$n = \frac{T_U}{T_N} = \frac{0.9}{4.2} = 0.2143 \tag{4}$$

Pro vypočtené n = 0.3556 odpovídá soustava třetího řádu.

$$Fs(p) = \frac{k}{(Tp+1)^3} \tag{5}$$

Zesílení a velikost časové konstanty určené podle tabulky 3.1 ze zdroje [3].

$$Fs(p) = \frac{1}{(2.25p+1)^3}$$

Pro návrh diskrétního PSD regulátoru byla použita metoda Ziegler-Nichols (Z-N). Kritické zesílení a kritická perioda byla určena ze simulace v simulinku na hodnotu K_{KRIT} = 8 a T_{KRIT} = 4.15.

Dopočet jednotlivých konstant regulátoru:

Zesílení:

$$K = 0.6 \cdot K_{KRIT} = 0.6 \cdot 8 = 4.8 \tag{6}$$

Integrační konstanta:

$$T_I = 0.5 \cdot T_{KRIT} = 0.5 \cdot 4.15 = 2.075 \tag{7}$$

Derivační konstanta:

$$T_D = 0.125 \cdot T_{KRIT} = 0.125 \cdot 4.15 = 0.51875 \tag{8}$$

Vypočtené hodnoty bylo nutné dále doladit pro zmenšení prvního překmitu a urychlení regulačního děje. Pro zmenšení překmitu bylo zapotřebí snížení zesílení a mírné zvětšení integrační konstanty. Z hlediska udržení relativně rychlého ustálení celého systému s malým překmitem je dobré držet derivační konstantu rovnu 0.25*T_I. Výsledné hodnoty diskrétního PSD regulátoru tedy jsou zesílení K=1.3, integrační konstanta T_I=2 a derivační konstanta T_D=0.5. PSD regulátor je doplněn filtrací derivační složky pro snížení vlivu zesílení rušivých složek a dále omezenou

sumační složkou a omezeným akčním zásahem. K přebuzení sumační složky dochází, pokud napětí na sumátoru dosáhne vyšší hodnoty, než kterou je akční člen schopný zpracovat. PSD regulátor byl implementován do PLC. Návrh PSD regulátoru, viz příloha 8.

Shodné parametry byly použity pro S-PD regulátor, pro regulaci aproximované soustavy ventilátoru. Změna struktury regulátoru z PSD na S-PD se využívá k získání druhého stupně volnosti, neboli je možnost nastavit regulátor na vhodné vyregulování poruchy a pomocí proměnné beta (b) dále doladit výstupní přechodovou charakteristiku při odezvě na skok řízení. Je-li proměnná b nastavená na hodnotu nula, jedná se o S-PD regulátor v opačném případě SP-D regulátor. Další výhodou daného typu regulátoru je možnost docílení aperiodického přechodového děje v případech, kdy u běžného typu PSD regulátoru by tento požadavek znamenal neúměrné prodloužení přechodového děje. Také zde byl regulátor doplněn filtrací derivační složky, ale místo omezení sumační složkou bylo použito dynamické omezení sumační složky. Dynamické omezení sumační složky vyhodnocuje rozdíl mezi velikostí akčního zásahu a skutečnou velikostí akčního zásahu. Pokud je rozdíl nenulový, dochází tedy k přebuzení a zápornou zpětnou vazbou je omezována hodnota na sumátoru. S-PD regulátor byl implementován v PLC. Návrh S-PD regulátoru viz příloha 9. [1], [5], [6]

7.1.2 Návrh feedforward regulátoru

Návrh regulátoru proběhl obdobně jako v předešlém případě. Z průběhu přechodové charakteristiky setrvačného článku třetího řádu (obrázek 13) byla odečtena hodnota inflexního bodu i=2.87, doba průtahu T_U=1.8 a doba náběhu T_N=7.8. Podle vzorce (4) byla určena soustava třetího řádu (5). Dosazením zesílení a velikost časové konstanty určené podle tabulky 3.1 ze zdroje [3], dostáváme aproximovanou soustavu.

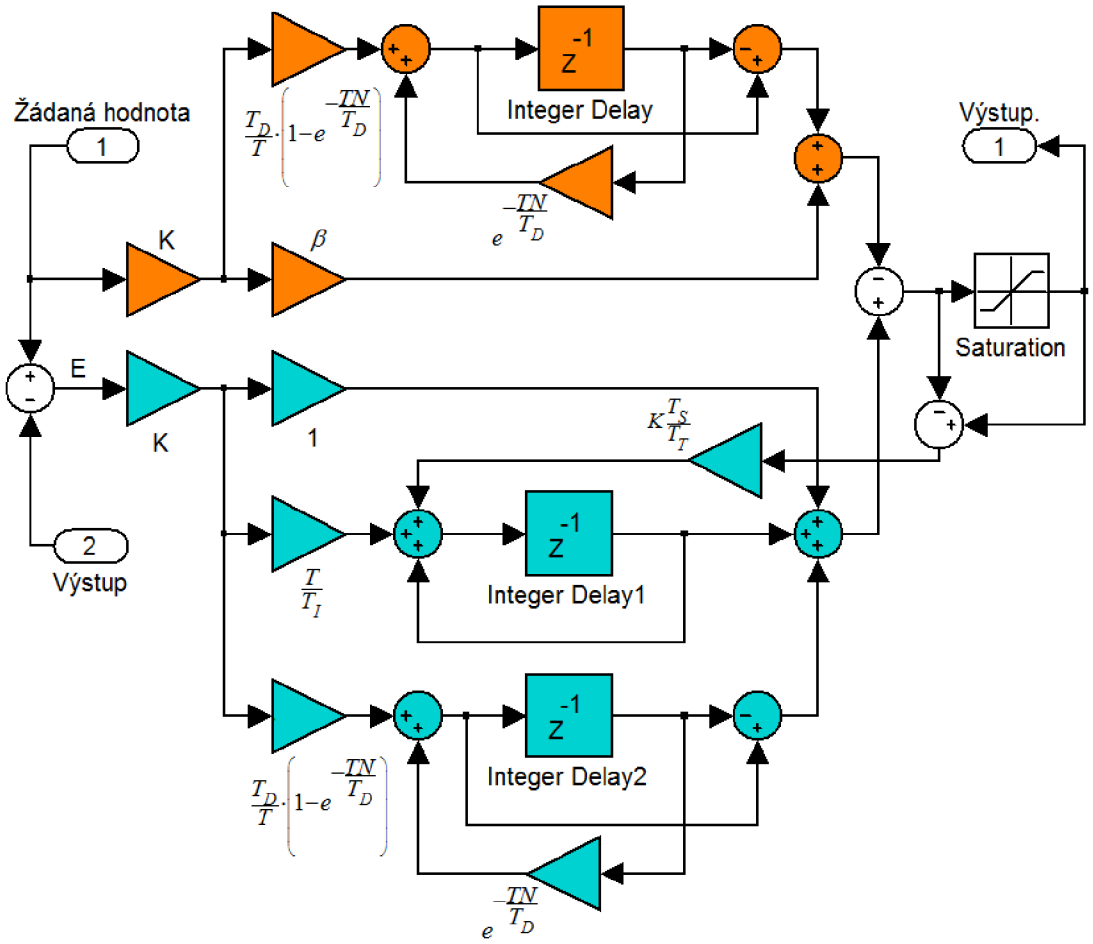
$$Fs(p) = \frac{1}{(2.25p+1)^3}$$

K návrhu regulátoru byla opět použita metoda Z-N. Kritické zesílení K_{KRIT} =8 a kritická perioda T_{KRIT} =8.2. Vypočtené zesílení (6) K=4.8, integrační konstanta (7) T_I=4.25 a derivační konstanta (8) T_D=1.025. Vypočtené hodnoty byly opět doladěny

pro docílení vhodného přechodového děje. Výsledné zesílení K=2.5, integrační konstanta T_I=4.7 a derivační konstanta T_D=1.225. Stejně jako u předešlých regulátorů, také zde bylo použito filtrace derivační složky u základního PSD regulátoru i v dopředné vazbě u PD regulátoru a dynamické omezení sumační složky.

Feedforward je ekvivalentní regulátorům typu S-PD, využívá se k získání druhého stupně volnosti, neboli je možnost nastavit regulátor na vhodné vyregulování poruchy a pomocí dopředné vazby doladit výstupní přechodovou charakteristiku při odezvě na skok řízení. Další výhodou je možnost docílení rychlého aperiodického přechodového děje na žádanou hodnotu. [7]

Na obrázku 15 je znázorněn feedforward regulátor použitý v simulačním modelu, viz obrázek 16, jako subsystém feedforward regulátor. Vstup ln1 odpovídá žádané hodnotě, vstup ln2 regulační odchylce a výstup out1 výstupu regulátoru.



Obrázek 16: Feedforward regulátor [7]

Obecné parametry:

Regulátory mají nastavenou vzorkovací periodu (T) T_S = 0.2, filtrační koeficient derivační složky N = 3, omezení akčního zásahu maximální hodnotou MaxAz = 10 a minimální MinAz = 0. Parametr T_T je potřeba odladit, při vysoké hodnotě vyřazuje působení zpětné vazby a malá hodnota vede k negativnímu ovlivnění rychlosti regulace. Parametr byl zvolen na hodnotu $T_T = 3$.

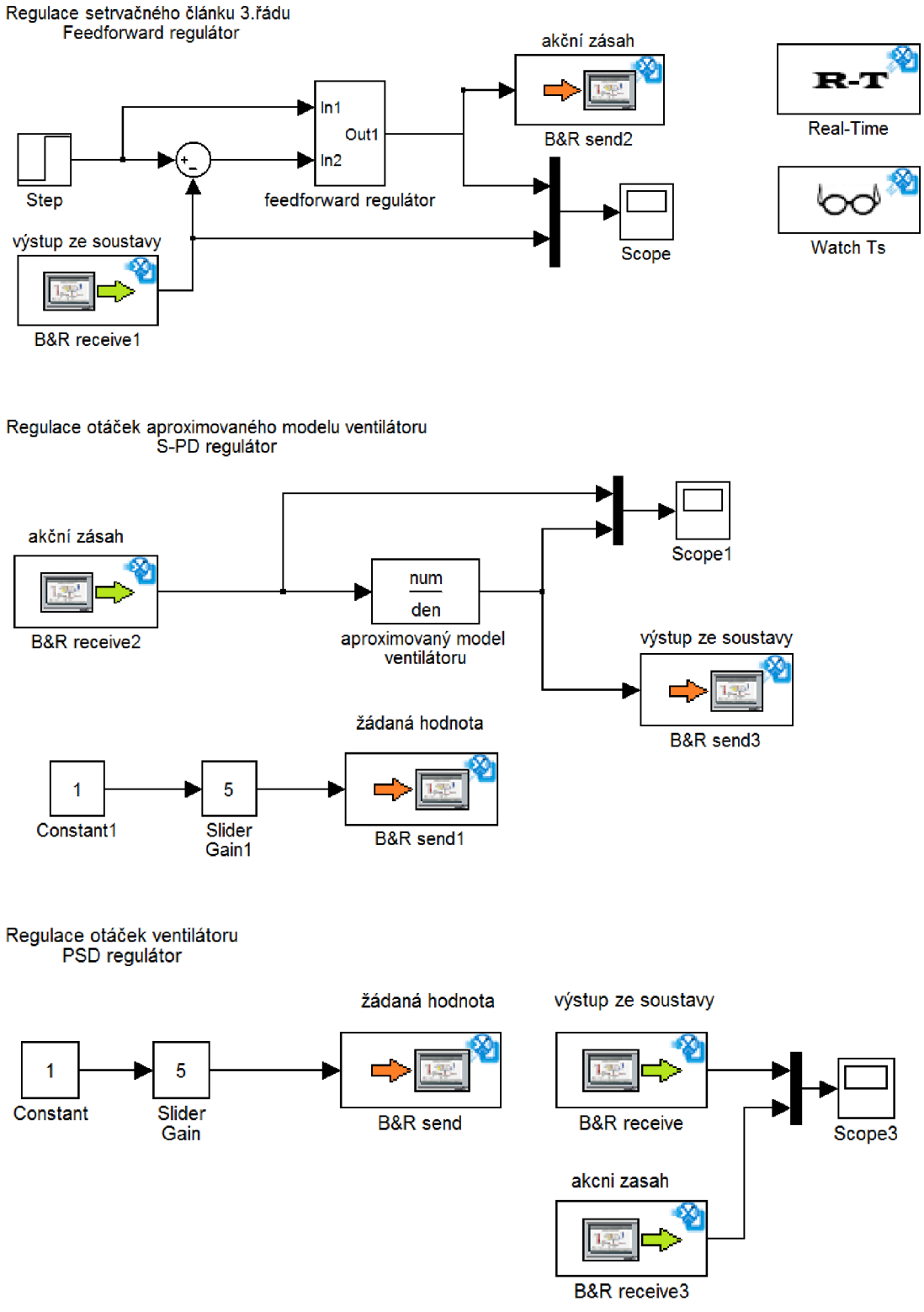
7.2 OVĚŘENÍ FUNKČNOSTI KOMUNIKACE

Pro ověření komunikace byl simulační model provázán se třemi PLC viz obrázek 16. Každá komunikace představovala jiný typ regulace, která může být požadována v praxi. Jedná se o fyzikální model řízený implementovaným regulátorem v PLC, kdy uživatel v simulinku pouze sleduje žádané veličiny a provádí úpravy parametrů regulátoru či žádané hodnoty. Další variantou je regulátor vytvořený v simulačním modelu regulující fyzikální model prostřednictvím PLC. Uživatel má možnost jednoduchého a přehledného návrhu regulátoru bez potřeby převodu do programovacího jazyka C. Regulátor může být odladěn a v konečné fázi naprogramován v jazyce C, potřebného pro implementaci do PLC. Třetí variantou je řízení regulátorem v PLC soustavu v simulačním modelu. Tímto způsobem je možné odladit nestabilní soustavy s regulátorem, který bude ovlivňován systémy jako za běžného provozu.

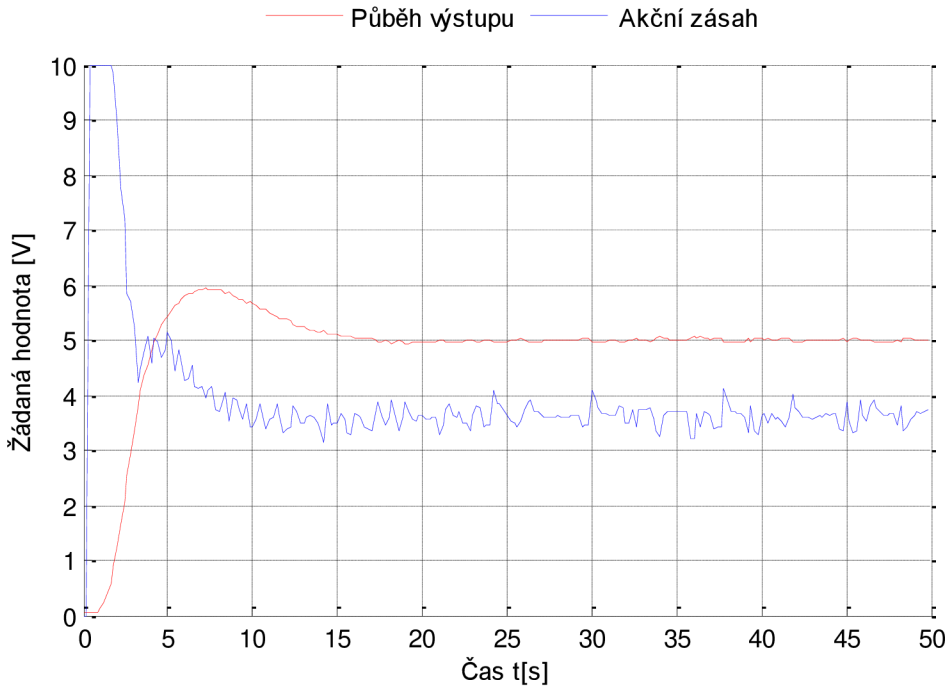
V simulačním modelu, viz obrázek 16, proběhla regulace fyzikálního modelu otáček ventilátoru implementovaným regulátorem PSD v PLC. Bloky byly užity pouze pro nastavení žádané hodnoty, vyčtení velikosti akčního zásahu a výstupu systému.

Regulace aproximovaného modelu ventilátoru obsahovala regulátor S-PD implementovaný v dalším PLC, který reguloval aproximovanou soustavu v simulinku. Bloky pro komunikaci s PLC přenášely akční zásah, výstup sytému a žádanou hodnotu od uživatele.

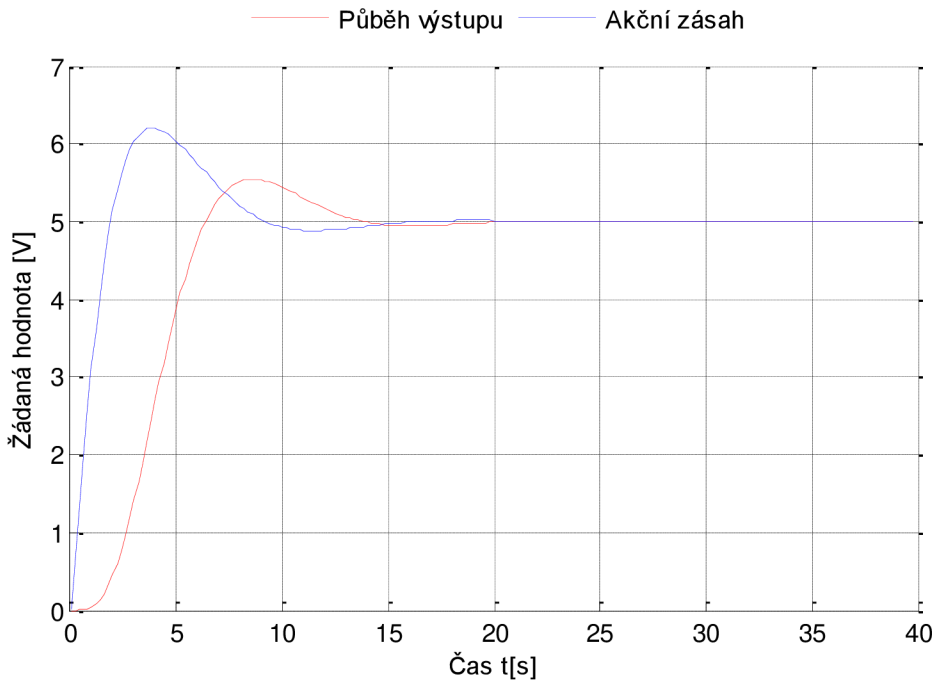
Poslední varianta obsahovala feedforward regulátor v simulačním modelu regulující přes PLC fyzikální model se setrvačným článkem třetího řádu na žádané napětí. Opět bloky přenášely výstup sytému z PLC a akční zásah do PLC.



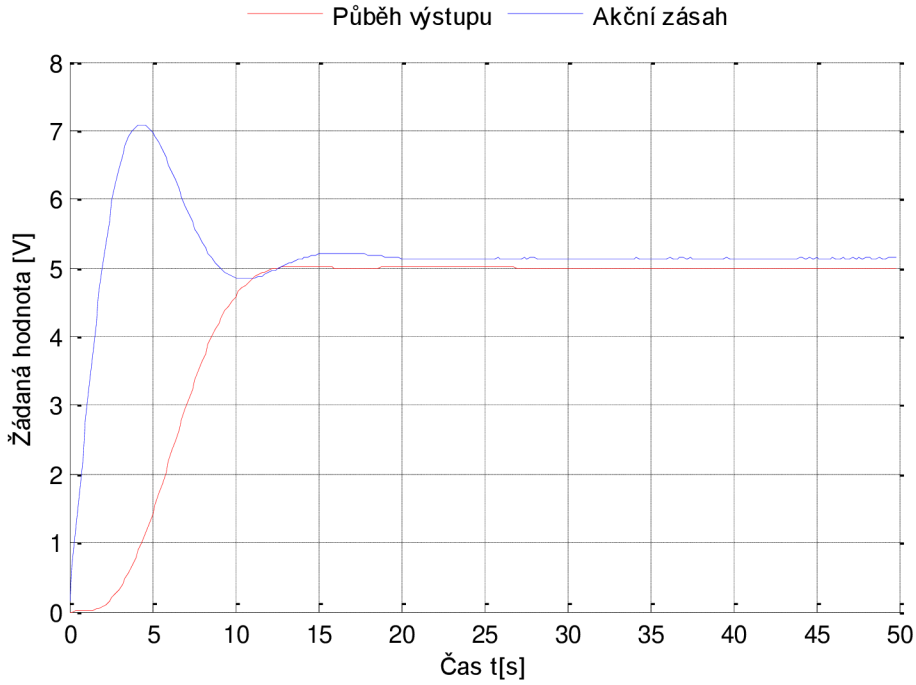
Obrázek 17: Komunikace s třemi PLC



Obrázek 18: Vyregulování otáček ventilátoru na žádanou hodnotu



Obrázek 19: Vyregulování aproximované soustavy ventilátoru



Obrázek 20: Vyregulování setrvačného článku třetího řádu na žádané napětí

8. ZÁVĚR

Bylo seznámeno s možnostmi a vlastnostmi použití reálného času v prostředí MATLAB/simulink. V simulinku došlo k vytvoření blocksetu obsahujícího podporu reálného času, sériové linky RS-232 a bloky pro komunikaci s řídicími systémy firmy B&R. Ověření funkčnosti blocksetu proběhlo na řízení vybraných fyzikálních modelech pomocí základních variant diskrétního PID regulátoru v laboratoři inteligentních regulátorů. Otestování blocksetu proběhlo na verzi MATLAB; R2008b, R2009b a R2010a.

Základem vývoje byla práce Vratislava Kořínka, která již nedostačovala pro potřeby laboratoře inteligentních regulátorů. Při vývoji nového real-time bloku se hledalo optimální řešení s co nejlepšími vlastnostmi udržení vzorkovací periody v reálném čase. Výsledné soft real-time řešení je porovnáno s dřívější koncepcí v histogramech, viz obrázek 4 a 5, doporučená minimální vzorkovací perioda se pohybuje okolo T_S=50ms. Jako doplněk vylepšení reálného času byl vyvinut blok Priority. Priority umožňuje zvýšení priority programu MATLAB/simulink v prostředí windows na Hight prioritu nebo Real-time prioritu. Real-time priorita znemožňuje během simulace zásahy do simulačního modelu, ale umožňuje docílení nejnižší vzorkovací periody až T_S = 6ms, viz srovnání v histogramech s předešlými verzemi obrázek 4 a 5. Bloková podpora reálného času je dále doplněna blokem Watch TS pro visuální kontrolu reálného času. Uživatel má možnost nastavit povolený rozptyl vzorkovací periody a změnou barvy mezi zelenou a červenou je indikováno její udržení.

Podpora pro sériové rozhraní RS232 v simulinku je realizována v M kódu, pro využití funkcí nabízené MATLABem na obsluhu a kontrolu linky. Komunikace lze také použít pro zavedení vzorkovací periody do simulačního modelu podle zdrojové části. Navození vzorkovací periody a porovnání se zdrojovou částí je zobrazeno v histogramu, viz obrázek 10. Volba komunikačního portu a nastavení vzorkovací periody podle zdrojové části je možné ve vytvořené masce bloku.

Při vývoji bloků pro komunikaci s řídicím systémem firmy B&R bylo použito programovacího jazyka C. Programovací jazyk C má výhodu oproti jazykům .NET

v podpoře s-funkcí (není potřeba přídavných souborů) a dále nižší úrovní přístupu k hardware. Jako podpora v simulinku jsou realizovány dva bloky. B&R receive pro příjem dat z PLC a B&R send pro odeslání dat do PLC. Je umožněna komunikace s neomezeným počtem proměnných, přičemž pro každou proměnnou je třeba použít jeden blok. Podporovány jsou základní datové typy, které je možno nastavit v PLC (double, float, int, short, char, unsigned int, unsigned short, unsigned char a BOOL). Datové typy a velikosti polí v blocích jsou nastavovány programově, při výskytu nesrovnalostí je realizován systém chybových hlášení, který zastaví simulaci a vypíše typ chyby a místo výskytu v programu. Pro snadnou konfiguraci byla vytvořena maska na bloky s dvěma možnostmi nastavení komunikace. Hlavní, které vyžaduje od uživatele zadání pouze základních parametrů a rozšířené, kde uživatel má možnost nakonfigurovat celou komunikaci ručně.

Vytvořené bloky (Real-time, Watch TS, serial send, serial receive, serial send-receive, B&R send a B&R receive) byly implementovány do blocksetu. Pomocí vytvořeného m.file souboru s názvem Install je možné blockset implementovat do knihovny simulinku pod názvem VUT toolbox, což značně ulehčuje implementaci bloků do simulačního modelu. V případě potřeby odstranění blocksetu z knihovny je přiložen m.file (Uninstall) pro odstranění a zrušení matlabovských cest.

Blockset byl použit k řízení vybraných fyzikálních modelů prostřednictvím tří PLC. Použité typy regulátorů byly PSD, S-PD a feed-forvard. Návrh proběhl pomocí metody Ziegler–Nichols s následným intuitivním doladěním konstant na požadovaný průběh výstupu. Dva algoritmy regulátoru byly implementovány v PLC a jeden v simulinku. Došlo k regulování fyzikálních modelů, ale také opačné varianty, kdy regulátor v PLC reguloval soustavu v simulinku.

S danou problematikou úzce souviselo důkladné seznámení s funkčním algoritmem simulinku, možnostmi vytváření s-funkcí a jejich provázání s maskou bloku. Při realizaci byly využity dva typy s-funkcí, C MEX s-funkce a level 2 m-file s-funkce. Realizaci komunikace s PLC vyžadovala nastudování struktury PVI skládající se z PVICOM rozhraní, PVI manažeru a PVI linky a možnosti jejich vyžití. Během vývoje práce bylo využito komunikace s technickým oddělením společnosti HUMUSOFT a B&R automation.

9. LITERATURA

[1] VELEBA, Václav. Číslicová řídicí technika: Počitačové cvičení. Brno : [s.n.], 2005. 77 s.

[2] PIVOŇKA, Petr. Číslicová řídicí technika. Brno : [s.n.], 2003. 151 s.

[3] BLAHA, Petr, VAVŘÍN, Petr. Řízení a regulace I : Základy regulace lineárních systémů-spojité a diskrétní. [s.l.] : [s.n.], [2005]. 214 s.

[4] KOŘÍNEK, Vratislav. Communication MATLAB-Ethernet. [s.l.]:[s.n.], 2004. 3 s.

[5] Ogata. K: Modern control engineering, fourth edition. Upper Saddle River, New Jersey: Prentice Hall, 1997. ISBN 0-13-314899-8

[6] Astrom K. J., Wittenmark B: Computer-controlled systeme, theory and design. New Jersey: Prentice Hall, 1997. ISBN 0-13-314899-8

[7] PIVOŇKA, Petr. Comparative analysis in implementations discrete PID controllers. *Sdf*. 2008, s. 1-6

[8] hw.cz: [online]. c1997-2009. Dostupný z WWW: <http://www.hw.cz/>

[9] MSDN: Microsoft Development [online]. c2009 Dostupný z WWW: <http://www.msdn.microsoft.com/>.

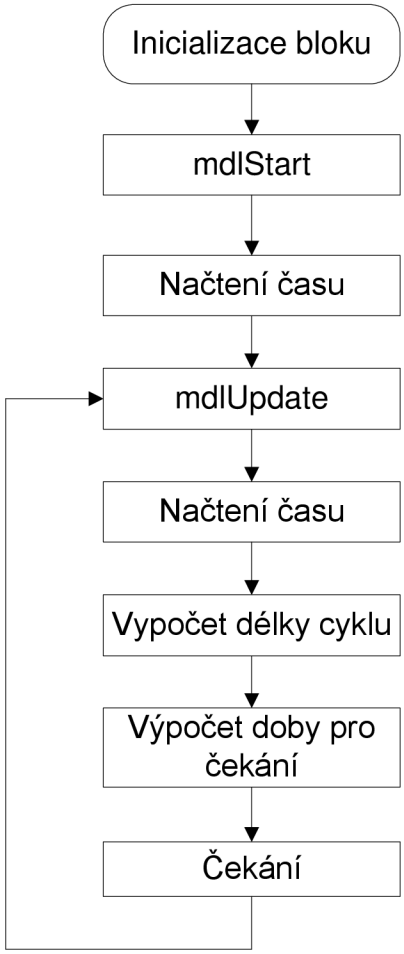
[10] Humusoft : Technical Computing, Control Engineering, Simulations [online]. c1991-2009 [cit. 2009-12-16]. Dostupný z WWW: <http://www.humusoft.com>.

[11] MathWorks : MATLAB and Simulink for Technical Computing [online]. c1994-2009. Dostupný z WWW: <http://www.mathworks.com/>.

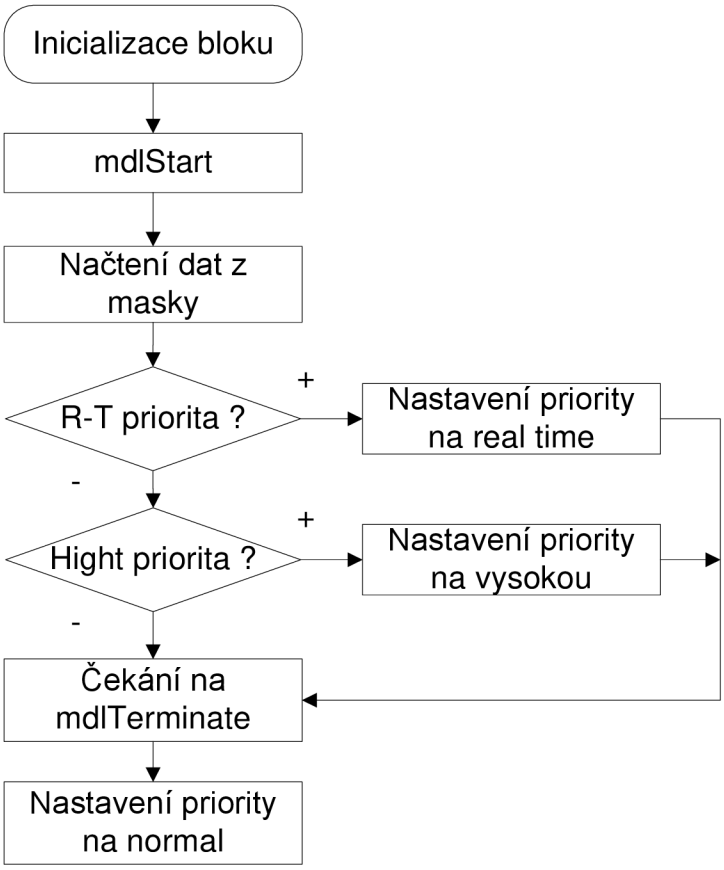
[12] Automation Studio Help, B&R Automation, 2008

Seznam příloh:

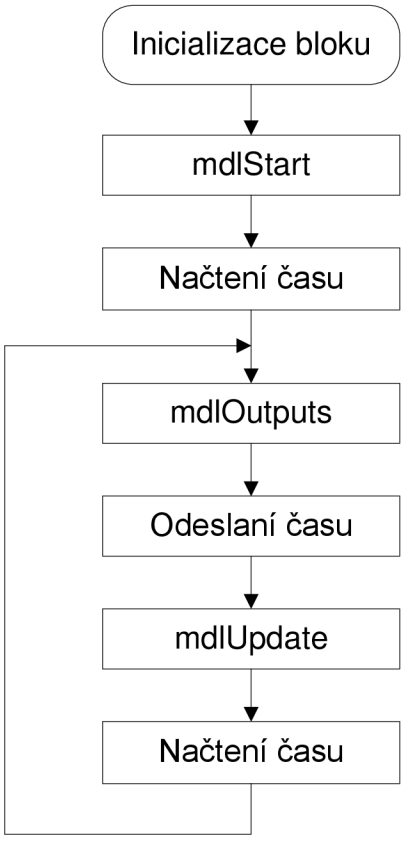
Příloha 1:


Inicializace bloku
mdlStart
Načtení času
mdlUpdate
Načtení času
Vypočet délky cyklu
Výpočet doby pro čekání
Čekání

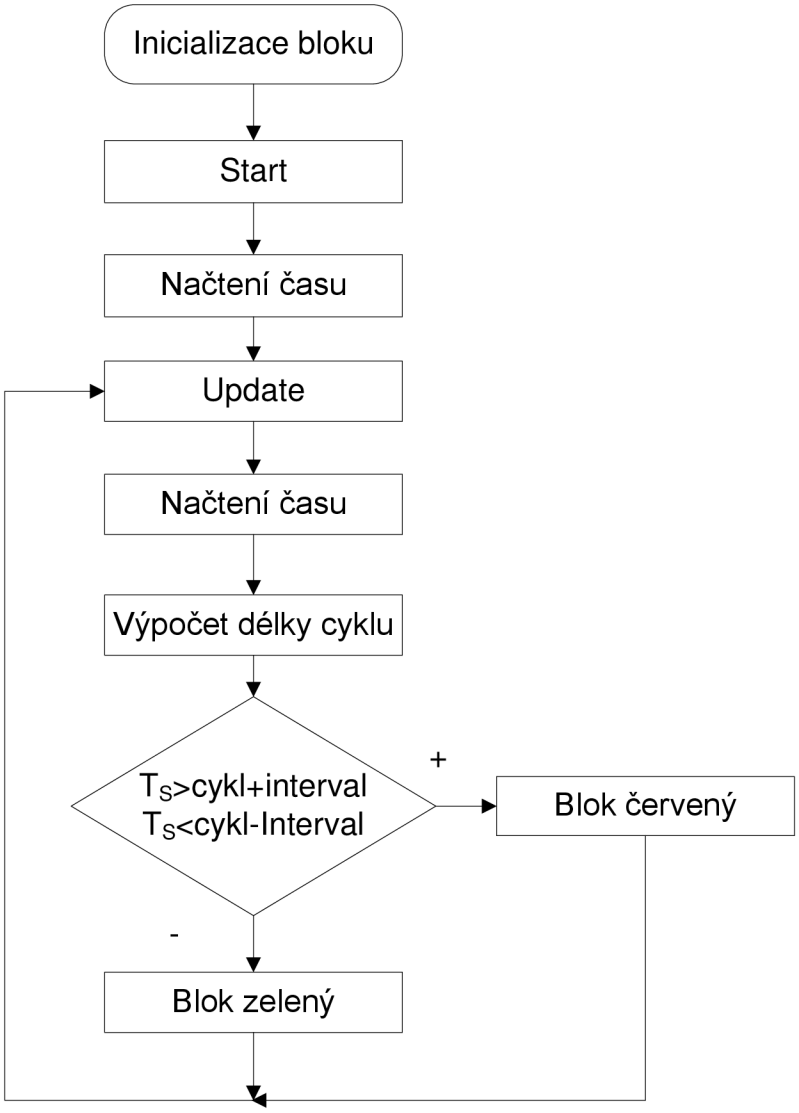
Příloha 2:



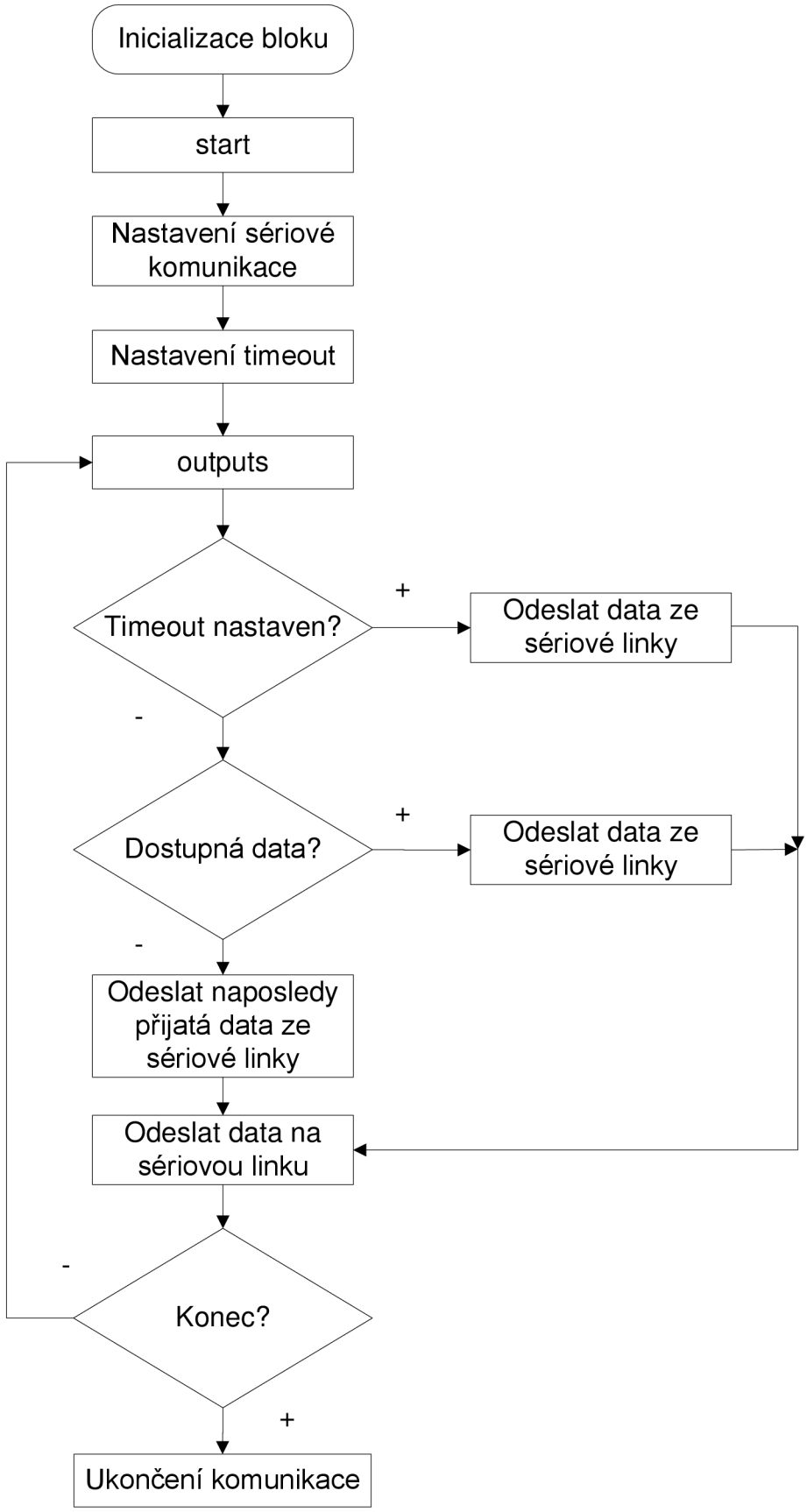
Příloha 3:



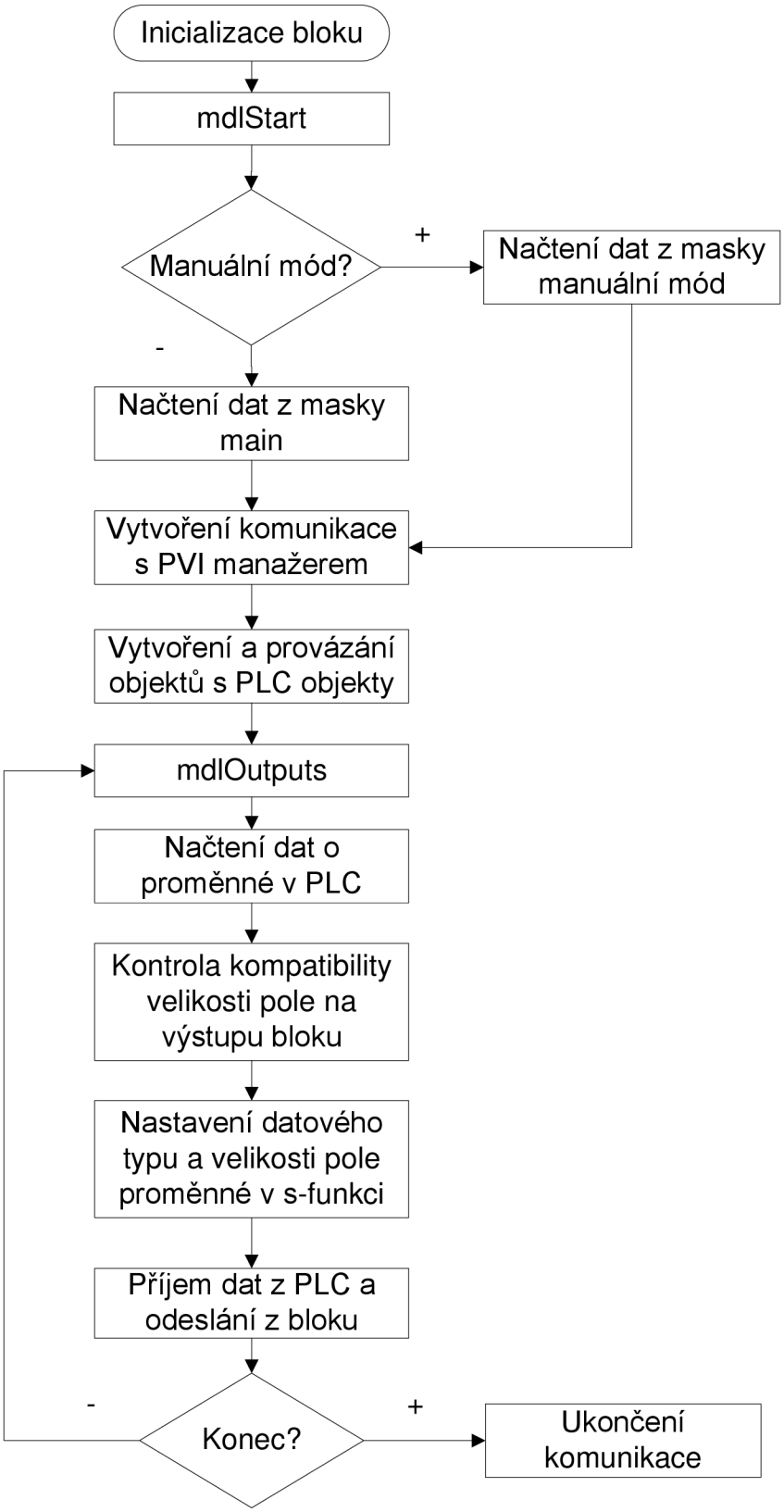
Příloha 4:


Inicializace bloku
Start
Načtení času
Update
Načtení času
Výpočet délky cyklu
TS>cykl+interval
TS<cykl-Interval
+
Blok červený
-
Blok zelený

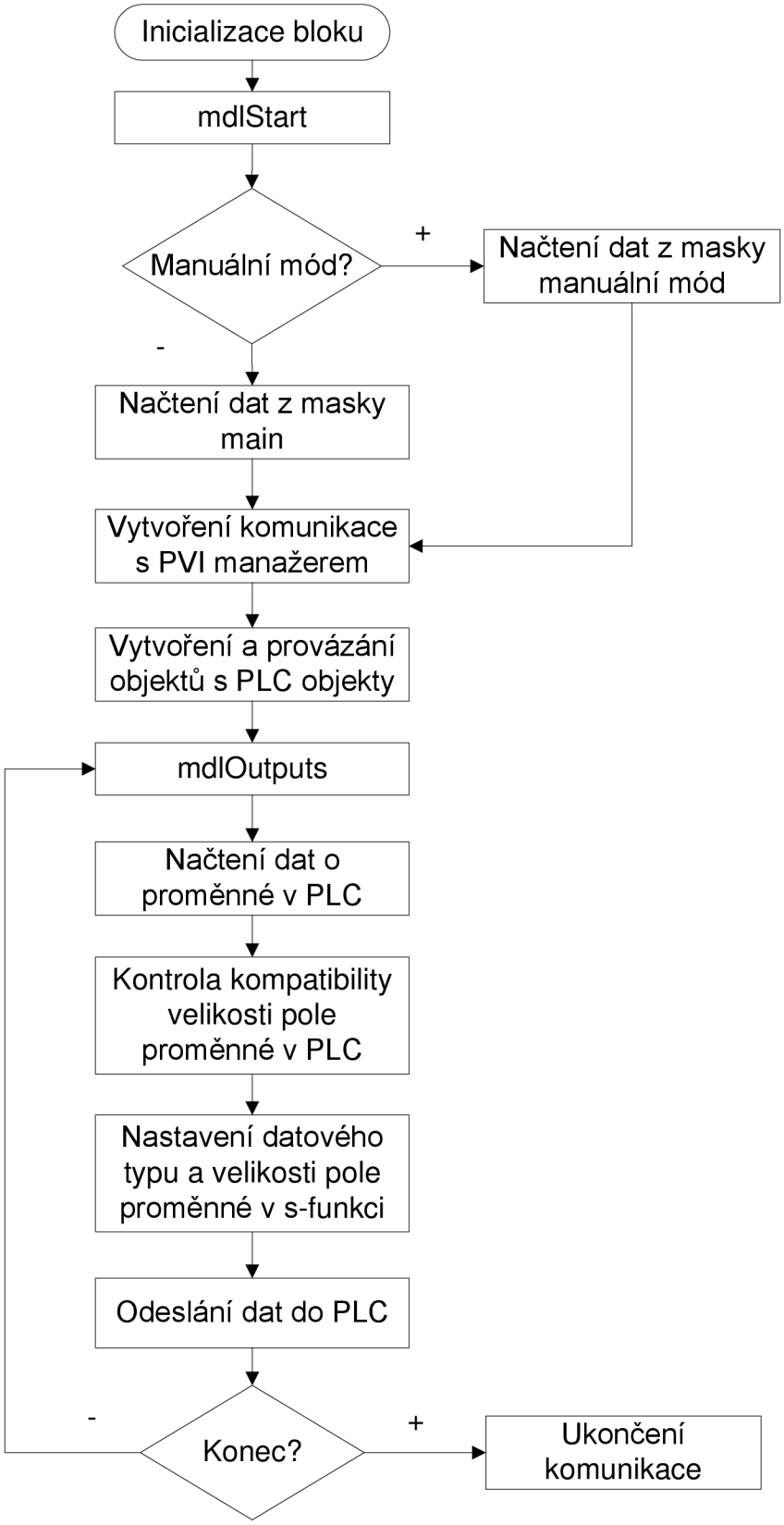
Příloha 5:


Inicializace bloku
start
Nastavení sériové komunikace
Nastavení timeout
outputs
Timeout nastaven?
+
Odeslat data ze sériové linky
-
Dostupná data?
+
Odeslat data ze sériové linky
-
Odeslat naposledy přijatá data ze sériové linky
Odeslat data na sériovou linku
Konec?
-
+
Ukončení komunikace

Příloha 6:


Inicializace bloku
mdlStart
Manuální mód?
+
Načtení dat z masky manuální mód
-
Načtení dat z masky main
Vytvoření komunikace s PVI manažerem
Vytvoření a provázání objektů s PLC objekty
mdlOutputs
Načtení dat o proměnné v PLC
Kontrola kompatibility velikosti pole na výstupu bloku
Nastavení datového typu a velikosti pole proměnné v s-funkci
Příjem dat z PLC a odeslání z bloku
-
Konec?
+
Ukončení komunikace

Příloha 7:


Inicializace bloku
mdlStart
Manuální mód?
+
Načtení dat z masky manuální mód
-
Načtení dat z masky main
Vytvoření komunikace s PVI manažerem
Vytvoření a provázání objektů s PLC objekty
mdlOutputs
Načtení dat o proměnné v PLC
Kontrola kompatibility velikosti pole proměnné v PLC
Nastavení datového typu a velikosti pole proměnné v s-funkci
Odeslání dat do PLC
Konec?
-
+
Ukončení komunikace

Příloha 8:

```
void _CYCLIC ZakladniCyclic( void ){
        Reg_Velicina=(REAL)(iAnalog01*(10.0/0x7fff));
        E = W - Reg_Velicina;
        U = K*E+sum+N*(K*E-ds+ds*exp(-Ts*N/Td));
        ds = K*E+ds*exp(-Ts*N/Td);
        sum = sum+K*Ts/Ti*E;
        if(sum>MaxAz)  sum = MaxAz;
        if(sum<MinAz)  sum = MinAz;
        if(U>MaxAz)  U = MaxAz;
        if(U<MinAz)  U = MinAz;
        Akcni_Zasah = U;
        oAnalog01 = (INT)(Akcni_Zasah*0x7fff/10);
}
```

Příloha 9:

```
void _CYCLIC ZakladniCyclic( void ){
        Y = vystup;
        U = K*(b*W-Y)+sum+N*K*(-Y-ds+ds*exp(-Ts*N/Td));
        ds = -Y+ds*exp(-Ts*N/Td);
        sum = sum+(W-Y)*Ts/Ti*K;
        if(sum>MaxAz) sum = sum+(MaxAz-U)*K*Ts/Tt;
        if(sum<MinAz) sum = sum+(MinAz-U)*K*Ts/Tt;
        if(U>MaxAz) U = MaxAz;
        if(U<MinAz) U = MinAz;
        akcni=U;
}
```