TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

VZDÁLENÝ PŘÍSTUP K REGULAČNÍM ÚLOHÁM NA CRIO/MYRIO

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Autor práce: **Bc. Ondřej Vinkler**
Vedoucí práce: Ing. Lukáš Hubka, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Vinkler**
Osobní číslo:	**M13000214**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Automatické řízení a inženýrská informatika**
Název tématu:	**Vzdálený přístup k regulačním úlohám na cRio/myRio**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Prozkoumejte možnosti vzdálené komunikace s jednotkou NI cRio nebo NI myRio a popište je.
2. Zrealizujte program pro vzdálené identifikační měření reálného procesu NI cRio či NI myRio.
3. Zrealizujte program pro vzdálené řízení procesu pomocí NI cRio či NI myRio.
4. Popište možnosti realizace podpůrného SW pro zpracování výsledků a návrh řízení a vybraný přístup zrealizujte.
5. Sestavte základní didaktický popis úlohy.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **40–50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **VLACH, J., HAVLÍČEK J. a VLACH J. Začínáme s LabVIEW. Praha: BEN - technická literatura, 2008.**

[2] **NI Developer Zone [online]. 2013 [cit. 2013-09-09]. Dostupné z: http://zone.ni.com.**

[3] **MODRLÁK, Osvald a Lukáš HUBKA. Automatické řízení: učební text. Liberec: Technická univerzita v Liberci, 2012.**

Vedoucí diplomové práce: **Ing. Lukáš Hubka, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant diplomové práce: **Ing. Jan Opálka**
Ústav mechatroniky a technické informatiky

Datum zadání diplomové práce: **10. října 2014**

Termín odevzdání diplomové práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci 14

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2015

Podpis: 

Poděkování

Rád bych poděkoval vedoucímu své diplomové práce Lukáši Hubkovi, za jeho pomoc při konzultacích. V neposlední řadě bych rád poděkoval všem, kteří mě podporovali při studiu a tvorbě této práce, zejména mým rodičům.

Abstrakt

V této práci se řeší problematika vzdáleného připojení k laboratorní úloze za pomoci hardwaru od National Instrument CompactRIO. Důležitou součástí je prozkoumání možností vzdáleného připojení, které teoreticky přicházejí v úvahu. Na základě zvážení výhod a nevýhod je vybrána vhodná metoda, která je tvořena dvěma částmi: programem na CompactRIO, který ovládá a čte data z připojené soustavy v reálné čase, a webovým rozhraním. Komunikace mezi těmito dvěma částmi je tvořena webovými službami (web services) na CompactRIO a asynchronními http požadavky tvořenými pomocí webového nástroje AJAX. Připojením na danou webovou stránku lze tedy měřit statické a dynamické vlastnosti Wattova regulátoru, numericky spočítat přenos modelu, navrhnout regulátor a vyzkoušet jeho vlastnosti v praxi.

Klíčová slova:

vzdálené připojení, CompactRIO, webové služby, PID regulátor, AJAX, Bootstrap, Google charts, LabView

Abstract

In this paper is dealt with issues of a remote access to a laboratory task with CompactRIO, a hardware designed by National Instrument. The important part is to analyze the possibilities of a remote connection which theoretically come into consideration. Based on advantages and disadvantages a suitable method is chosen. The method consists of two parts: a program for CompactRIO, which controls and reads data from a connected system in real time, and a web-based interface. The communication between these two parts is created by a web service at CompactRIO and asynchronous http requests using a web tool AJAX. The connection to a web page enables the user to measure static and dynamic properties of Watt regulator, calculate the transfer function of the model, design a controller and test its properties in practice.

Key words:

remote access, CompactRIO, web services, PID controller, AJAX, Bootstrap, Google charts, LabView

OBSAH

Seznam obrázků

Seznam tabulek

Seznam příloh

1 Úvod

V době rozvíjejících se internetových technologií se nabízí stále větší možnost zapojit internet do výuky. Internetem dnes disponuje téměř každý. Neomezené internetové připojení je v domácnosti stejně běžné jako teplá voda. Čím dál více lidí také vlastní chytré telefony nebo tablety, které poskytují mobilní internetové připojení. A když náhodou není nic podobného k dispozici, vždy se najde internetová kavárna nebo restaurace s možností bezplatného připojení k wifi internetu.

Ve výuce automatického řízení se často pracuje s reálnými modely. Dělají se identifikační měření soustav, identifikují se vlivy poruch, navrhuje se řídicí systém, u kterého se následně testuje funkčnost na reálném modelu, a porovnávají se naměřená data s vypočtenými. Tato měření se ve většině případu provádějí přímo v laboratoři s nutnou přítomností pedagoga, jelikož studenti by neměli být v laboratoři sami. Když poté student píše zprávu o provedeném měření, může zjistit, že data, která naměřil, jsou nedostatečná nebo špatná a že by je potřeboval naměřit znovu. Musel by se vydat do školy, domluvit se s učitelem a provést nové měření. Může to být velká ztráta času. To přivádí k myšlence vytvořit aplikaci, pomocí které by šlo provést měření z pohodlí domova. Student by se s pomocí webového prohlížeče nebo speciální aplikace připojil k modelu v laboratoři a mohl by bez problém naměřit potřebná data. Vše bez přítomnosti pedagoga, bez nutnosti absolvovat cestu do školy. Samozřejmě toto vzdálené měření nemůže nahradit přímou interakci s modelem v laboratoři, ale pro studenta, který již s modelem pracoval, to může být užitečné zjednodušení v případě komplikací s naměřenými daty přímo z laboratoře.

Základním cílem této práce je vytvoření programů, které umožní vzdálené měření na reálné soustavě v laboratoři. Základem je hardware, který je schopný ovládat řízenou soustavu v reálném čase. Na tomto kusu hardwaru, je spuštěný jeden program, který komunikuje se soustavou a následně data zpracovává a předává k uživateli. Druhý program se nachází na straně uživatele (v jeho počítači) a díky němu dokáže komunikovat a ovládat řízenou soustavu.

V práci bude představený hardware, který je použitý pro komunikaci přes internet a zároveň k ovládání reálné soustavy v laboratoři. Budou popsány parametry zařízení, jeho základní i přídavné části a také software, který je nezbytný a vhodný pro programování tohoto zařízení. Následovat bude jednoduchý popis soustavy, ke které je systém připojený a který se používá k testování funkčnosti celého systému. Důležitou částí celé práce jsou analýza možností vzdáleného připojení k reálné soustavě a tvorba grafického rozhraní pro uživatele. Některé možnosti využívají pro přístup pouze webový prohlížeč, jiné jsou realizované pomocí vlastní aplikace. Budou zmíněny výhody i nevýhody jed-

notlivých přístupů, např.: nutnosti instalace dalších programů, nebo náročnost realizace. V další části práce bude popsáno, které postupy a algoritmy byly použity při tvorbě programu pro řídící zařízení, včetně podprogramů pro obsluhu http požadavků. Součástí programu jsou i podpůrné funkce pro identifikaci soustavy. Další kapitola potom pojednává o tvorbě grafického rozhraní pro webový prohlížeč, které komunikuje se zařízením v laboratoři.

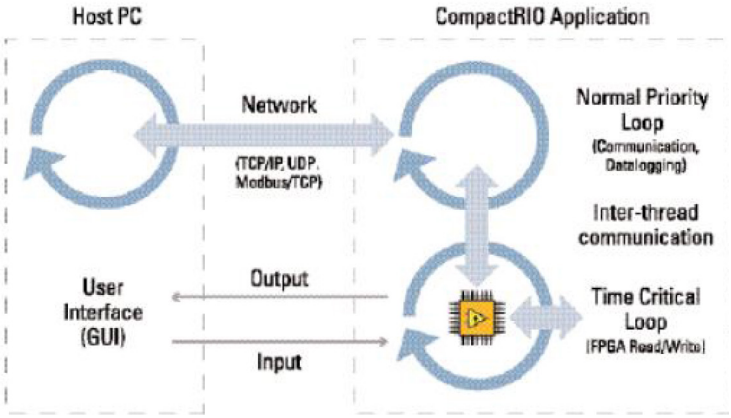
2 Použitý harware

2.1 CompactRIO

CompactRIO (obrázek 2.1) je zařízení, které se využívá pro řízení a sběr dat z reálných systémů[1]. Kombinuje real-time procesor a nastavitelné FPGA (field-programmable gate array), které poskytují vysokou výpočetní schopnost a rychlost. CompactRIO se jako kus hardwaru skládá ze dvou částí. V první části jsou vyvedené konektory pro napájení, ethernet, usb, RS-232, signalizační diody a podobně. Druhá část je nastavitelné šasí, do kterého se připojují přídavné C series moduly. CompactRIO je k dostání v několika verzích, které se liší výkonem procesoru, velikostí hradlového pole, velikostí dočasné a permanentní paměti a velikostí šasí. V této práci se pracuje s CompactRIO verze c9076. Procesor dosahuje rychlosti 400MHz. FPGA typu spartan-6 LX45 má 43661 logických buněk, 58 násobiček a 2088 kB paměti RAM. Šasí, do kterého lze připojit až 4 přídavné moduly, je napojené přímo na FPGA. Tudíž veškeré digitální nebo analogové vstupy a výstupy jsou obsluhovány právě přes FPGA. CompactRIO je možné jednoduše připojit k síti pomocí zabudovaného ethernetového konektoru. Další konektory (USB, RS-232) slouží pro připojení dalších zařízení nebo k rozšíření permanentní paměti. Velikost paměti RAM je 256 MB a velikost permanentní paměti je 512 MB.

Obrázek 2.1 CompactRIO c9076 [1]

CompactRIO obsahuje vlastní operační systém LabView real-time ETS OS, který umožňuje vývoj aplikací, obsahujících časově kritické smyčky a smyčky s nižší prioritou (obrázek 2.2). Časově kritické smyčky jsou spojeny s FPGA a většinou se týkají čtení a zápisu do vstupů a vý-



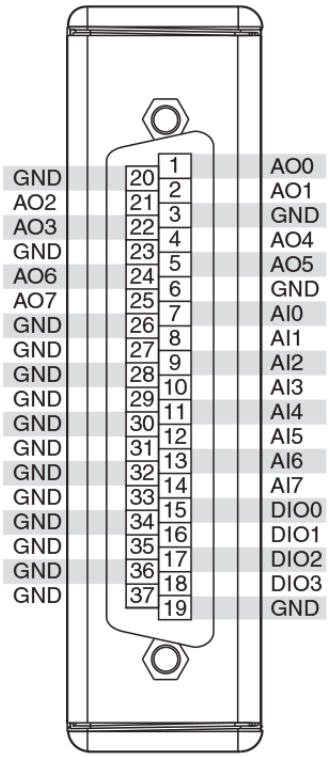
Obrázek 2.2 Real-time operační systém [1]

stupů. Základní funkce pro čtení a zápis jsou v systému již zabudovány a umožňují vysokou rychlost přístupu ke vstupům a výstupům. Smyčky s nižší prioritou jsou používány pro záznam a zpracování dat a ethernetovou nebo sériovou komunikaci. Do CompactRIO se následně může doinstalovat další software, který přidává další funkce.

CompactRIO dále obsahuje vestavěné servery pro Virtual instrument software architecture (VISA), HTTP, FTP. VISA server umožňuje vzdálenou komunikaci s FPGA vstupy a výstupy přes ethernet. HTTP server umožnuje přístup k souborům, HTML stránkám a uživatelským rozhraním LabView aplikací pomocí webového prohlížeče a příslušného plug-inu. FTP server umožňuje přístup k nahraným datům a konfiguračním souborům.

2.2 C SERIES MODULY

C series moduly jsou navrženy jako soběstačné měřící moduly. Jsou v nich obsaženy veškeré obvody, které jsou potřeba k měření, včetně D/A, A/D převodníků a proto je možné připojit senzory rovnou k těmto modulům, bez nutnosti přidávání další elektroniky. Je mnoho různých druhů modulů, protože každý je vybaven jiným druhem vstupu nebo výstupu, počínaje analogovými a digitálními vstupy a výstupy přes termoelektrický článek po diferenční a TTL vstupy. Každý modul je vybaven nějakým druhem konektoru (např.: šroubové svorky, BNC, D-Sub). Při zapojení modulu do šasí compactRIO, je tento modul přímo propojen s FPGA a je vytvořen vysoce výkonný měřící systém. Jsou poskytnuty téměř neomezené možnosti pro časování, spouštění, synchronizaci a rozhodování. V této práci jsou ke compactRIO připojeny dva moduly: NI 9381 a NI 9263.



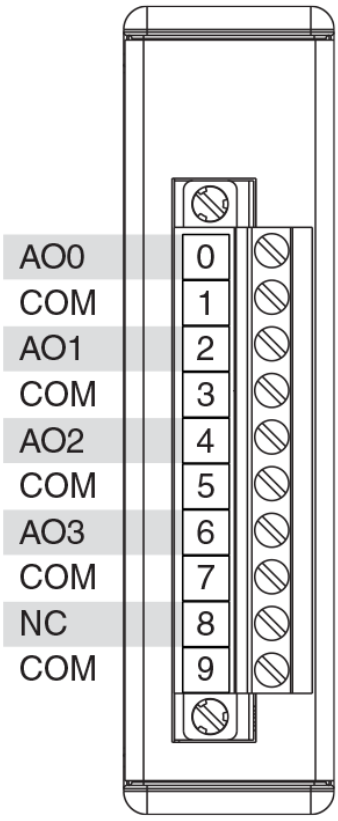
Obrázek 2.3 D-Sub konektor a rozložení pinů na NI 9381[2]

NI 9381 – multifunkční vstupně výstupní modul [2]

Obsahuje 8 analogových vstupů pro vstupní napětí v rozmezí 0-5V. O převod na digitální signál se stará 12 bitový převodník s postupnou aproximací. Dále obsahuje 8 analogových výstupů pro výstupní signál o síle 0-5V. O převod z digitálního signálu se stará 12 bitový flash převodník. Dále obsahuje 4 digitální vstupy/ výstupy. Pro zapojení vstupů je vybavený 37 pinovým D-Sub konektorem (obrázek 2.3). Jeden časovač dokáže udržet vzorkovací frekvenci na hodnotě 20 kS/s. Tato hodnota se poměrově dělí mezi všechny analogové vstupy a výstupy. To znamená, že je buď jeden vstup nebo výstup vzorkovaný na 20 kS/s, nebo při zapojení všech vstupů a výstupů je každý vzorkovaný frekvencí 1250 S/s.

NI 9263 – analogový výstupní modul [3]

Tento modul obsahuje 4 analogové výstupy. Je přesnější, protože o převod veličiny se stará 16 bitový flash převodník. Je vybavený 10 pinovými šroubovacími svorkami (obrázek 2.4). Kromě výstupů je na piny vyvedena i izolovaná napěťová zem modulu. Časovač v tomto případě poskytuje vzorkovací frekvenci 100 kS/s a aktualizuje všechny výstupy zároveň. Zároveň obsahuje ±30V přepěťovou ochranu, ochranu proti zkratu, malé přeslechy, vysokou rychlost přeběhu a vysokou relativní přesnost.



Obrázek 2.4 Šroubovací svorka a rozložení pinů na NI9263 [3]

2.3 Vývojový software

Doporučený software pro práci s compactRIO je v první řadě jednoznačně vývojové prostředí LabView. V LabView se vytváří veškeré programy, které potom mají běžet compactRIO a řídit nebo sbírat data z měřeného systému. Pro real-time procesor je možné programovat i v jazycích C/C++, ale FPGA je nutné programovat v LabView. Následně je také dobré mít nainstalovaný FPGA modul, který je nutností pro programování na compactRIO. Programování pro FPGA využívá nízko úrovňové softwarové nástroje a hardware popisující jazyky (HDL). Tyto nástroje právě poskytuje FPGA modul. Dalším potřebným modul je Real-time modul. Tento modul rozšiřuje grafické programování v LabView až do možnosti vytvoření samostatné aplikace. LabView je původně určeno pro operační systémy jako např. Windows, které nejsou optimalizovány pro běh časově kritických aplikací, které mají striktní časové požadavky. Real-time module umožňuje vyvíjet aplikace pro real-time operační systém a následně s ním komunikovat. Tyto moduly jsou potřeba pro správnou komunikaci s compactRIO, ale existuje spousta dalších modulů, které usnadňují práci v LabView. Přidávají nové funkce nebo již zpracované a vysoce optimalizované algoritmy. V této práci se ještě používá Control design and simulation modul. Jak už název napovídá, tento modul umožňuje řídit reálné systémy a simulovat modely systémů. Umožňuje efektivní využívání nástrojů spojených se řízením od přenosových funkcí, stavového popisu po přechodovou a bodeho charakteristiku. Není problém zapojení reálné měřené veličiny do simulačního schématu a tím vytvořit řídící model.

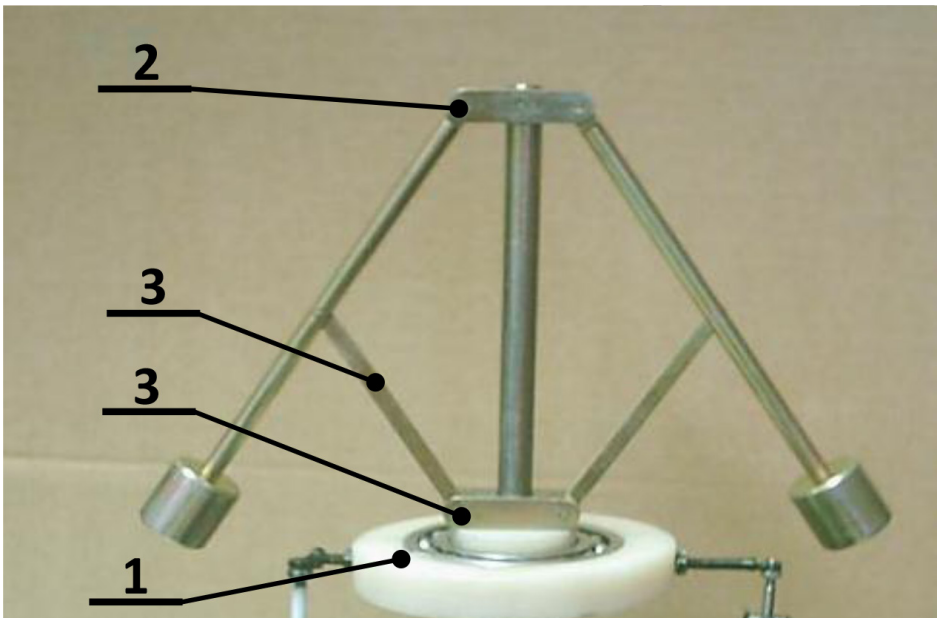
2.4 Základy programování v LabView

Uživatelské rozhraní programu v LabView má obvykle podobu čelního ovládacího panelu měřicího přístroje. Z tohoto důvodu se program nazývá Virtual Instrument, zkráceně VI. Každé VI se skládá z front panelu, který je obdobou grafického uživatelského rozhraní, a blokového diagramu, což je zdrojový kód programu [4]. Na front panel se vkládají ovládací a zobrazovací prvky, např. tlačítka, přepínače, světelné signalizace a grafy. U těchto prvků je možné nastavovat spoustu vlastností, které se podobně řeší i při aplikaci přímo v hardwaru. Třeba u tlačítka lze nastavit, jestli bude spínat na vzestupnou nebo sestupnou hranu při stisku nebo puštění. Instance grafického rozhraní se zobrazují i v blokovém schématu, kde po zapojení s dalšími funkce tvoří zdrojový kód programu. Vykonávání programu v LabView je řízeno datovým tokem a nikoli vykonáváním řádků kódu jdoucích po sobě. Datový tok jednoznačně určuje směr provádění programu. Funkce se provede právě tehdy, kdy na všech vstupech jsou potřebná data k jejímu provedení. Po vykonání má funkce na svých výstupech zpracovaná data, která předá dál. V textových programech je nutné vždy definovat proměnné. Určitou obdobou v LabView je datový tok reprezentovaný datový spojem. LabView automaticky alokuje paměť pro data a nejsou-li již používaná, automaticky je také smaže. Zvětšování a zmenšování polí nebo řetězců se také děje automaticky. Programování v LabView je velice intuitivní, ale u složitějších programů se může stávat, že na obrazovce je jen změť datový spojů a je obtížné se orientovat. Ale pokud programátor dokáže správně využívat možnosti LabView, např, subVI, orientace v programu se zjednodušuje a LabView se tak stává velice mocným nástrojem, pomocí kterého lze vytvořit téměř cokoliv.

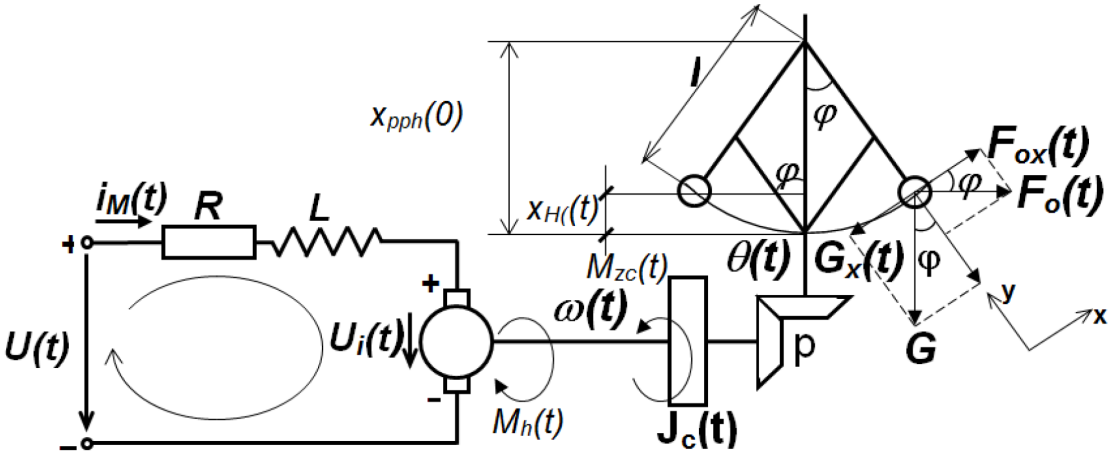
2.5 Řízený proces

Pro testování vyvíjeného systému je ke compactRIO připojena reálná soustava, v tomto případě Wattův regulátor [5], který se používá ke stabilizaci otáček parního stroje. Skládá se ze dvou závaží (koule), která rotují podél svislé osy a jsou poháněna strojem, jehož otáčky mají být regulovány. Čím větší jsou otáčky stroje, tím větší odstředivá síla působí na závaží, která se díky upevnění převádí na svislý pohyb, který je dále převáděn k ventilu přivádějícímu páru ke stroji. Je to vlastně mechanická zpětná vazba, která dovoluje působením poměrně malých sil regulovat velmi výkonný stroj.

V laboratoři není wattův regulátor primárně využíván k regulaci, ale jako systém, který má nelineární charakter a u kterého se měří aktuální poloha v závislosti na otáčkách motoru. Laboratorní soustava se skládá ze stejnosměrného motoru, který je přes převodovku s fixním převodním koeficientem spojený s wattovým regulátorem. Spolu s tachogenerátorem jsou tyto části uloženy v základně soustavy. Je měřena aktuální výška koulí. Otáčky stejnosměrného motoru jsou ovládány zdrojem, do které vstupuje napětí



Obrázek 2.5 Wattův regulátor (1 - ložisko, 2 - upevnění hlavních ramen, 3 - stabilizační ramena)



Obrázek 2.6 Mechanické schéma Wattova regulátoru [5]

v rozsahu 0-5V a vystupuje napětí pro motor v rozsahu 0-24V. Vstup do tohoto zdroje je výstup z compactRIO. Senzor pozice dává výstup v rozmezí 0-10V a je proporcionální s výškou koulí regulátoru. Tento výstup je připojen na vstup compactRIO.

Mechanické regulátory bývají většinou velice nelineární. Nejinak je tomu i u Wattova regulátoru, který je jejich klasickým zástupcem. Matematický popis je odvozený ze schématu na obrázku 2.6. V odvození se počítá s modelem stejnosměrného motoru, dynamickou rovnováhou sil působících na systém závaží a D'Alambertova principu. Jelikož je stejnosměrný motor spojený se systémem závaží přes pevné spřáhlo, je možné model popsat třemi diferenciálními rovnicemi [5].

První rovnice (1) představuje elektrickou část motoru. Druhá rovnice (2) představuje pohybovou rovnici systému závaží a třetí rovnice (3) představuje momentovou větu. Jednotlivé symboly jsou popsány v tabulce 2.1.

$$\frac{di_M(t)}{dt} + \frac{R}{L} i_M(t) + \frac{km}{L} \omega(t) = \frac{1}{L} U(t) \quad (1)$$

$$\frac{d^2\varphi(t)}{dt^2} + \frac{b}{m}\frac{d\varphi(t)}{dt} + g \sin\big(\varphi(t)\big) - \theta(t)^2 * l * \sin\big(\varphi(t)\big) * cos\big(\varphi(t)\big) = 0 \quad (2)$$

$$J_c(t)\frac{d\omega(t)}{dt} = [M_h(t) - M_{zc}(t)] \quad (3)$$

Je vidět, že už rovnice ideálního wattova regulátoru jsou nelineární. Je možné předpokládat, že původní neideální systém bude mít z hlediska nelinearity ještě horší vlastnosti. Bohužel nelinearita je z hlediska řízení komplikace. Regulace takového systému může být velice obtížná, a proto je třeba se na nelinearitu zaměřit následně volit vhodné způsoby řízení.

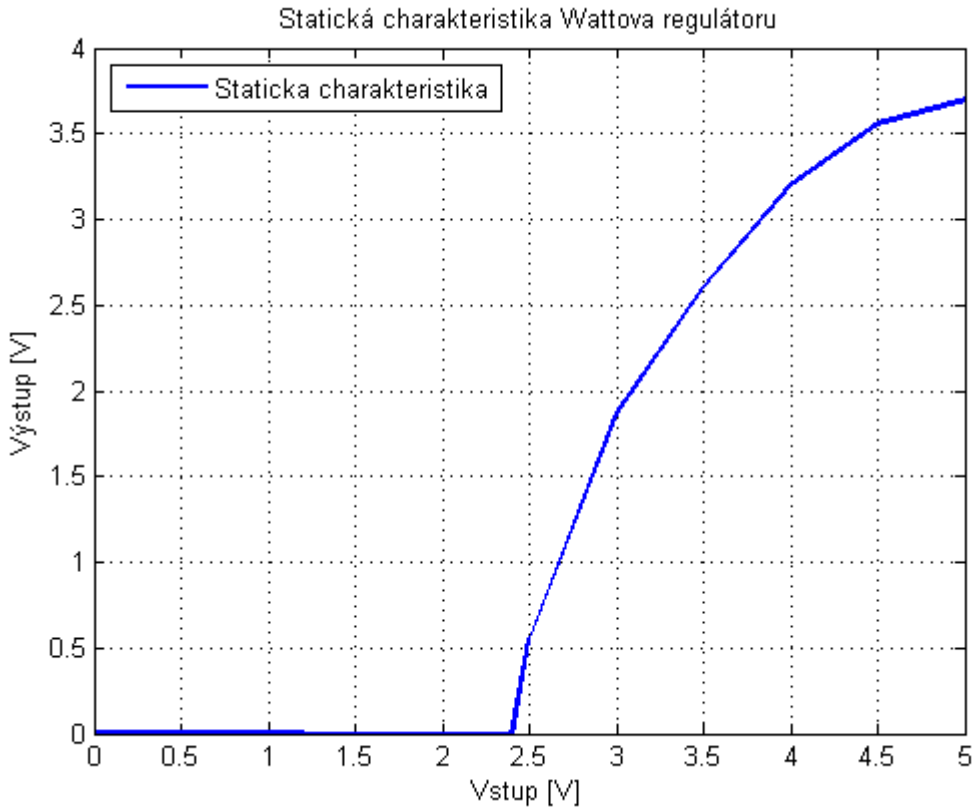
Nyní budou blíže prozkoumány vlastnosti Wattova regulátoru z hlediska řízení. Statická charakteristika byla měřena při otevřené smyčce v rozsahu 0-5V ve skocích po 0,5V. Výsledky je možné vidět v tabulce 2.2. Na obrázku 2.7 je grafické zobrazení statické charakteristiky. Je vidět, že statická charakteristika je podle předpokladů nelineární. V rozmezí 0-2,4V je pásmo necitlivosti.

Symbol	Jednotka	Popis
$J_c(t)$	$Kg*m^2$	Celkový moment setrvačnosti
$M_h(t) = km * i_M(t)$	Nm	Momentová rovnice stejnosměrného motoru
$M_{zc}(t) = M_{oc}(t) + M_{zext}(t)$	Nm	Celkový točivý moment zátěže
$M_{oc}(t) = M_r(t) + M_{zr}(t)$	Nm	Točivý moment zátěže způsobený třením ložiska a aerodynamickým třením
$M_{zext}(t) = 0$	Nm	Vnější točivý moment
$\theta(t) = \frac{1}{p}\omega(t)$	$rad*s^{-1}$	Úhlová rychlost systému se závažím
$\varphi(t)$	rad	Úhel vychýlení ramen závaží
$\omega(t)$	$rad*s^{-1}$	Úhlová rychlost motoru
b	$kg*m^2*s^{-1}$	Koeficient tření
g	$m*s^{-2}$	Gravitační konstanta
km	V*s	Konstanta motoru
m	kg	Hmotnost závaží
$i_M(t)$	A	Proud v cívce motoru
L	H	Induktance motoru
R	Ω	Odpor motoru
U(t)	V	Napětí zdroje motoru
l	m	Délka ramene záváží
p	-	Převodový poměr [1:8]

Tabulka 2.1 Symboly a základní rovnice pro výpočet ideálního wattova regulátoru

Vstup[V]	0	0,5	1	1,5	2	2,4	2,5	3	3,5	4	4,5	5
Výstup[V]	0	0	0	0	0	0	0,55	1,87	2,65	3,2	3,56	3,7

Tabulka 2.2 Statická charakteristika wattova regulátoru



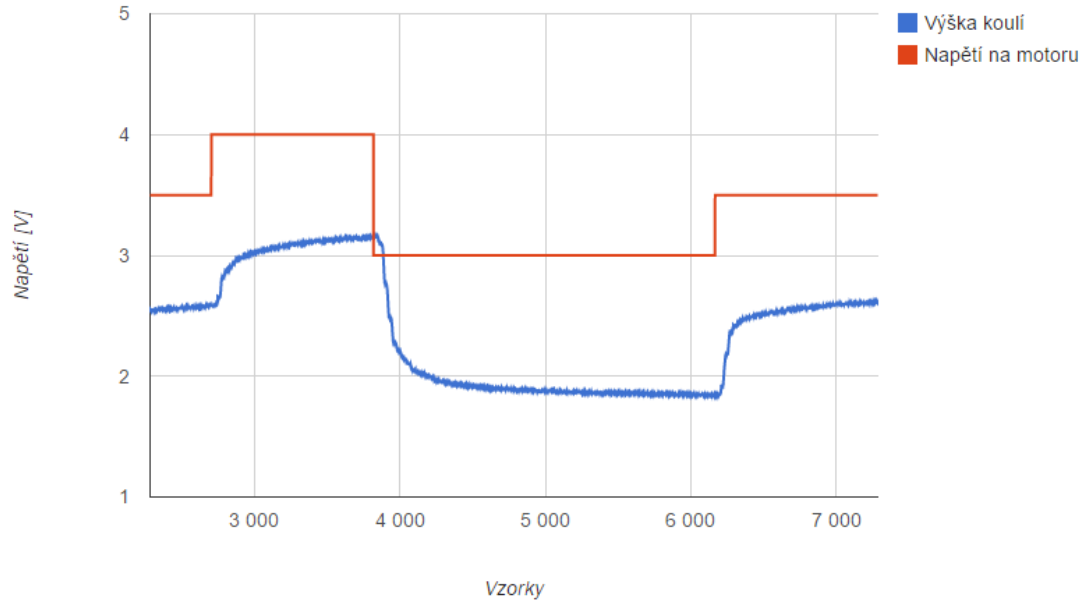
Obrázek 2.7 Statická charakteristika wattova regulátoru

Byl vybrán pracovní bod na vstupu 3,5V. Okolo něj byla zvolena přibližně lineární pracovní oblast v rozmezí od 3V do 4V. Vně této oblasti je systém popsán jiným přenosem. Bylo provedeno několik po sobě jdoucích skoků vstupní hodnoty (obrázek 2.8). Při identifikaci se zjistilo, že v tomto rozmezí se zesílení systému liší v hodnotách 1,1 až 1,5.

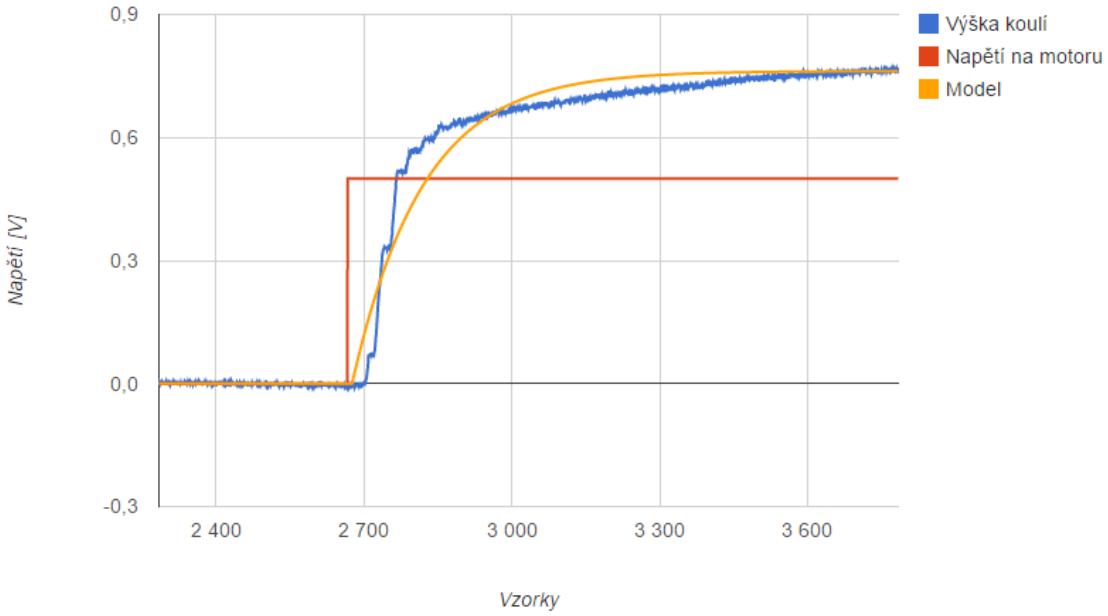
$$G(s) = \frac{1{,}3}{1{,}55s + 1} \tag{4}$$

Výsledný přenos pro tento pracovní bod je zapsán v rovnici 4.

Porovnání simulovaného modelu a reálných naměřených dat je vidět na obrázku 2.9.



Obrázek 2.8 Měření pro identifikaci modelu



Obrázek 2.9 Porovnání simulovaného modelu a naměřených dat

3 Možnosti vzdáleného připojení

Jelikož lze CompactRIO díky ethernetovému konektoru připojit k internetové síti, existuje několik možností jak k němu vzdáleně přistupovat. Některé přístupy jsou založené pouze na podpoře LabView, jiné využívají jiné programovací jazyky a komunikují s CompactRIO pomocí síťových protokolů, které LabVIew samozřejmě také podporuje. Přístupy lze dělit do dvou skupin podle toho, zda využívají tenkého nebo tlustého klienta. Tenký klient znamená, že pro připojení k cRIO je využívaný pouze webový prohlížeč. Výhodou je, že uživatel pouze spustí internetový prohlížeč a nemusí nic instalovat. Naproti tomu při použití tlustého klienta je buď potřeba stáhnout samotnou aplikaci, nebo je vyžadována instalace nějaké součásti. Tlustý klient je pro uživatele náročnější na spuštění. Bude popsáno pět možností přístupů. První je pomocí nástroje Remote front panel v LabView. Další možnosti jsou pomocí shared variables nebo network stream, což jsou funkce, které umožňují komunikaci mezi více samostatnými LabView programy. Další možnost využívá webové služby a grafické rozhraní ve webovém prohlížeči, což je jediný zástupce tenkého klienta. Nakonec je zmíněný postup pro vytvoření komunikace pomocí TCP/IP protokolu.

3.1 Remote front panel

Remote front panel dovoluje prohlížení a interakci z jiného programu LabView nebo z webového prohlížeče. První věc, se kterou se člověk seznámí při programování v LabView je, že každý program se skládá z front panelu a blokového diagramu. Front panel reprezentuje grafické rozhraní. Naproti tomu blokový diagram reprezentuje funkce programu. Tyto dvě části jsou spolu vždy spojeny ve VI. V LabView je možné pracovat s front panely na zařízení, které je fyzicky umístěno jinde než je spuštěný program. Takové vzdálené připojení se nazývá remote front panel [6]. Pokud se tedy pro vzdálený přístup použije remote front panel, je ovládání naprosto jednoduché, naprosto stejné jako v LabView s omezením několika málo funkcí. Jedna velká nevýhoda je, že na počítači z něhož se vzdáleně přistupuje, je nutné mít nainstalované vývojové prostředí LabView. Z důvodu ceny licencí toto není vhodný přístup.

Tento problém lze částečně obejít využitím LabView RTE a nástroje Web publishing tool. Je to varianta obyčejného remote front panelu a dovoluje zobrazit remote front panel ve webovém prohlížeči. Výhodou je, že není potřeba mít nainstalované LabView, ale pouze LabView RTE, které ke spuštění nepotřebuje licenci. Nevýhodou jsou zase omezené vlastnosti. Uživatel nemůže front panel nijak upravovat a může používat pouze tlačítka, která jsou na front panelu zobrazena.

Remote front panel ve webovém prohlížeči pracuje na základě architektury klient-

-server [7]. Když webový prohlížeč nahrává webovou stránku, která obsahuje Web front panel, snaží se vyhledat LabView plug-in (instaluje se spolu s LabView RTE). Pokud ho najde, nahraje ho, a ten poté spustí LabView RTE. Engine dále komunikuje se serverem, který je spuštěný například na cRIO. Umožňuje stahovat front panel a dostávat pravidelné aktualizace. Veškerý běh programu je uskutečňován na serveru (cRIO), takže klient pouze ví, jak aktualizovat a dostávat nové informace. Do hotového programu není nutné nijak zasahovat. Pokud program spolehlivě funguje v LabView, využije se jednoduchý nástroj Web publing tool, integrovaný v LabView, který z front panelu programu vytvoří webovou stránku.

Tato metoda jako taková má několik aspektů, které je před nasazením třeba prozkoumat. Aplikace musí být schopna poskytovat aktuální data všem připojeným uživatelům. To je v pořádku, pokud přenášená data jsou např. teplota, indikátor stavu nebo aktuální hodnota otáček motoru. Pokud ale přenášené informace obsahují grafy v reálném čase, potřebujeme vysokou přenosovou rychlost. Toho lze dosáhnout využitím špičkového serveru, ale ne vždy je takové řešení možné. Proto je příhodné zjistit další cesty k optimalizaci přenosu.

Jednou takovou cestou je redukování přenášených dat [8]. V LabView existuje nástroj, který dokáže zobrazit hodnotu přenášených dat. V LabView se vybere Tools >> Remote panel Connection Manager. Zde lze poté sledovat, jak se mění velikost přenášených dat v závislosti na úpravách VI. Jako příklad poslouží monitorování teploty. Pokud je na front panelu zobrazena pouze aktuální teplota, velikost přenášených dat je pouze 1-2 kb/s. Pokud se zobrazuje graf s 1000 body měření tak velikost přenášených dat stoupne na 8 kb/s. Další cestou ke snížení přenosu dat je omezení aktualizací. Je to jednoduché. Do spuštěného VI, ve kterém pravděpodobně běží while smyčka, se přidá zpoždění. V příkladu s aktuální teplotou to znamená, že při zpoždění 1 sekunda klesne velikost přenášených dat z 1 – 2 kb/s na 60 b/s, což je asi 30 krát méně. Poslední cestou je vyhýbání se komponentám Property node a events. Tyto komponenty pro komunikaci také používají architekturu klient-server, a proto v případě použití velkého množství by mohlo dojít k zahlcení serveru a přetečení fronty.

Další důležité věci, na které je třeba pamatovat, jsou rozdíly ve funkcích přímo v LabView a ve webovém prohlížeči. Některé aplikace, které používají dialogová okna nebo okna subVI, nemusí správně fungovat z důvodu reprezentace ve webové prohlížeči pomocí plug-inu. Ten se snaží zachovávat přesnost uživatelského rozhraní, ale ne vždy se mu to daří. Je potřeba se vyhýbat while smyčkám, které v sobě nemají žádnou funkci Wait. Program se tuto smyčku snaží vykonávat co nejrychleji a spotřebovává 100% výkonu procesoru, což zpětně znemožňuje serveru provádět další úkony, a proto může front panel přestat odpovídat. Pokud aplikace obsahuje dialogové okno na otevření souboru, prohlížeč obdrží chybové hlášení, protože není možné procházet soubory vzdáleně. Ze stejného důvodu je zakázané tlačítko pro výběr cesty při ukládání souboru. Jednou z nej-

větších nevýhod je nemožnost využití komponenty events a jejich obsluha.

Při vytváření aplikace pro vzdálený přístup se předpokládá, že na zařízení (v tomto případě cRIO) je spuštěné VI, které řídí nebo sbírá data ze systému. Pomocí Web Publishing Tool se vytvoří webová stránka, která obsahuje front panel daného VI [9]. Ta se nahraje do zařízení a je možné jí prohlížet a ovládat ve webové prohlížeči. Ke stránce lze přistupovat klasicky pomocí IP adresy zařízení a názvu webové stránky, který byl zadán při vytváření. Tato metoda má výhodu, že je jednoduchá k vytvoření a zprovoznění. Další výhodou je pěkné uživatelské rozhraní, které se jednoduše ovládá a dokáže zobrazovat měřenou veličinu téměř v reálném čase. Uživatel, který se chce připojit k webu, musí mít nainstalovaný LabView run-time engine. Dále musí mít nainstalovaný plug-in ve webovém prohlížeči. Většinou se tento plug-in instaluje spolu s LabView run-time enginem, ale mohou se vyskytnout problémy u nepodporovaných prohlížečů, které se však dají vyřešit. Potřebné knihovny se vždy nainstalují do webového prohlížeče Internet explorer. Ty poté lze vzít a zkopírovat do příslušných složek u jiných webových prohlížečů. I přes tyto nevýhody může být vzdálený přístup vytvořený tímto způsobem velice intuitivní a jednoduchý k používání.

3.2 Samostatná aplikace LabView

V LabView existuje zajímavá varianta, která dovoluje zkompilovat program do samostatně fungující podoby. Buď se může jednat o instalační program, nebo rovnou spustitelný bez nutnosti instalace. To je pro vzdálený přístup velice zajímavá možnost. Není potřeba mít nainstalované LabView, ale pouze LabView RTE. Grafické rozhraní je tvořené klasickým front panelem. Další výhodou toho přístupu může být rozdělení výpočetní náročnosti mezi server a uživatelský počítač. Část výpočtu se může provádět rovnou na serveru (cRIO) a druhá část se může počítat už u uživatele. Může tím dojít ke snížení objemu přenášených dat i úlevě procesoru serveru, což je vždy žádoucí. Jediné, co se musí řešit, je vytvoření datové komunikace mezi jednotlivými programy. Jeden, spuštěný na cRIO, a druhý, který se spuštěný na uživatelském počítači. V LabView jsou k dispozici dvě možnosti jak vytvořit efektivní komunikaci: Network shared variables a Network streams.

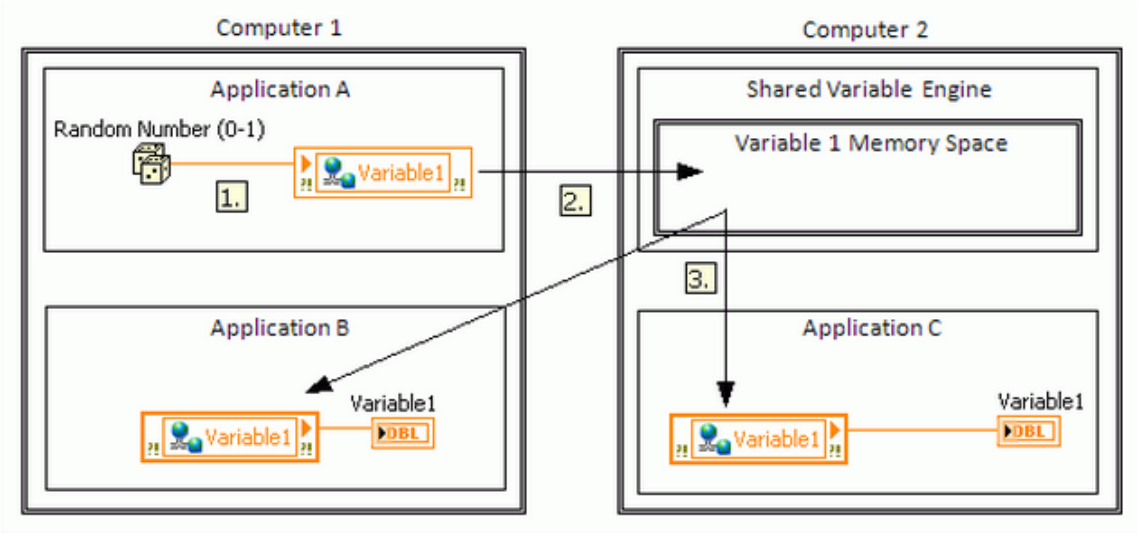
3.2.1 Network shared variables

National instrument vyvinuli network shared variable, aby pokryli potřeby přenosu dat z měřící stanice do uživatelského rozhraní. S protokolem TCP/IP je nutností transformovat naměřená data do nestrukturovaného proudu bytů ve vysílající aplikaci a v přijímající aplikaci potom tento proud zase převádět do původního formátu. Network shared variable převádí data z proudů bytů do strukturovaných data samo a tím odpadá nutnost

programovat složité transformační programy. Tyto proměnné mohou reprezentovat širokou škálu datových typů od typu string a boolean, přes waveform až po pole čísel typu double nebo integer.

Network shared variable posílají data pomocí softwaru zvaný Shared variable engine (SVE). Tento software se instaluje to počítače spolu s LabView a je nainstalovaný také na cRIO. Provádí aktualizace daných proměnných pomocí NI Publish-subscribe protocol [10]. Název publish-subscribe popisuje model komunikace, kde jedna strana (publisher) pošle aktualizace na server (SVE) a druhá strana (subscriber) poté obdrží tyto aktualizace přímo od serveru. Na obrázku 3.1 je jednoduchý příklad s network shared variable. V kroku jedna se do proměnné Variable1 zapisuje náhodné číslo. V kroku dvě se posílá aktualizace na SVE, aby v následujícím kroku SVE opět poslal aktualizaci do aplikace B a C. Z příkladu je vidět, že aplikace A a B spolu nemohou komunikovat přímo, ačkoliv jsou na stejném počítači. Díky této vlastnosti může v systému vznikat zpoždění, které při spojitém přenosu dat může působit problémy, ale pro aktualizaci jednotlivých hodnot je dostatečně rychlý [11]. Problémy se spojitým přenosem se dají obejít posíláním dat v dávkách, anebo se musí použít jiná technologie (network streams).

Vzdálený přístup k měření je v tomto případě udělán pomocí dvou aplikací, které mezi sebou komunikují právě pomocí network shared variables. Jedna aplikace (bude se nazývat programová) je spuštěná na cRIO a je v ní realizovaný veškerý přístup k soustavě, sběr dat, výpočty, atd. Druhá aplikace (uživatelská) obsahuje pouze zobrazovací prvky a možnost zadávání parametrů pro řízení. Důležitá data se přenášejí pomocí network shared variables. Tato metoda má několik nevýhod. Jak bylo psáno výše, první problém nastává při spojitém přenosu dat. Další nevýhodou je, že pro používání samostatné aplikace, která byla vytvořena v LabView, je potřeba mít nainstalovaný LabView runtime engine. Výhodou je pěkné grafické rozhraní díky front panelu LabView a intuitivní ovládání.



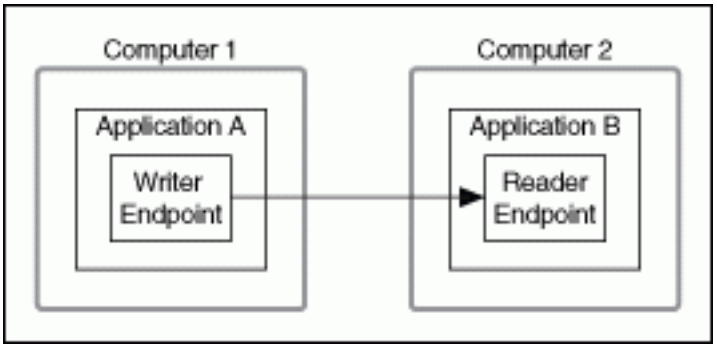
Obrázek 3.1 Real-time operační systém [12]

3.2.2 NETWORK STREAMS

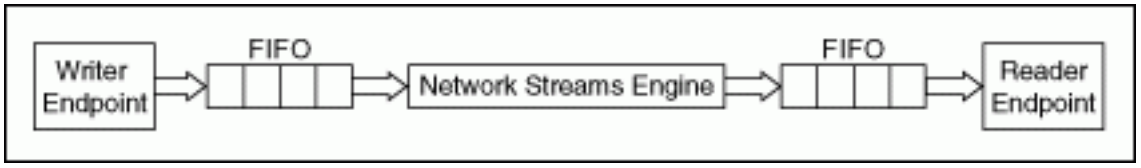
Pro posílání dat mezi dvěma LabView aplikacemi vytvořil National instrument nástroj, který poskytuje bezeztrátový jednosměrný přenos dat mezi dvěma místy, kde na jedné straně je zapisovač (writer) a na druhé je čítač (reader). Pomocí network streams se dají posílat téměř veškeré LabView typy s výjimkou LabView class a refnum, ale nejrychlejší přenos mají typy: číselný skalár, boolean a jejich jednorozměrné pole. Obrázek 3.2 zobrazuje přenos dat z aplikace A na počítači 1 do Aplikace B na počítači 2. Stejně tak lze posílat data mezi aplikacemi na stejném počítači. Je důležité si uvědomit, že přenos dat je jednosměrný, takže pro vytvoření obousměrného toku dat je potřeba využít network stream dvakrát.

Samotná data jsou přenášena mezi FIFO (first in first out) buffery koncových instancí pomocí softwaru zvaný Network streams engine (NSE). Obrázek 3.3 zobrazuje tok dat při přenosu. V prvním kroku zapisovač zapíše data do svého FIFO bufferu. V dalším kroku NSE přenese data z bufferu zapisovače do bufferu čítače a v posledním kroku čítač přečte data ze svého FIFO bufferu [14].

Ačkoliv network streams poskytují jednoduché vytvoření komunikace mezi dvěma body, stále existují situace, kdy jejich využití v aplikaci nemusí být výhodné. Obecně lze říci, že network streams nejsou vhodné pro použití v řídících algoritmech. Ethernet typicky není spolehlivá komunikace pro řídící aplikace a navíc čtení a zápis do network streams není deterministický. Přesto se může využít pro komunikaci se vzdáleným HMI (human machine interface), ale nesmí být v časově kritické smyčce. Místo toho se může využít RT FIFO nebo queues pro komunikaci mezi časově kritickou a komunikační smyčkou a následně použít network streams pro výměnu dat mezi komunikační smyčkou a vzdáleným HMI.



Obrázek 3.2 Princip přenosu dat u network streams [14]



Obrázek 3.3 Tok dat při přenosu network streams [14]

Vzdálený přístup k měřené úloze je potom podobný jako při použití shared variables. Komunikují mezi sebou dvě aplikace, které jsou obě vytvořené v LabView a pro fungování musí mít nainstalovaný LabView runtime engine. Pro posílání dat se ale nepoužívá shared variable ale network stream, nebo jdou obě technologie kombinovat. Pro přenos měřených dat z cRIO se použije network stream a pro zadávání vstupních parametrů shared variable. Opět největší nevýhodou je, že network streams i shared variables jsou prvky LabView a nejdou využít žádným jiným softwarem.

3.3 Web services

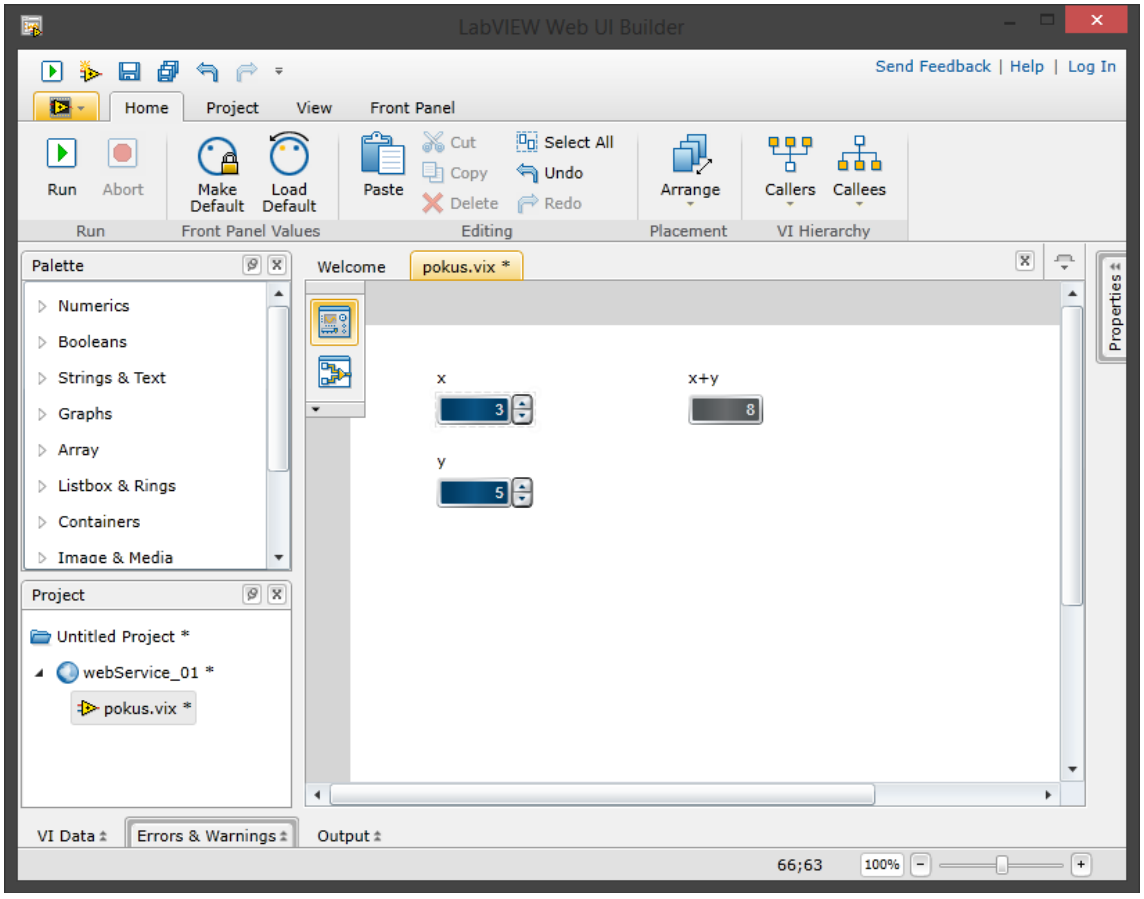
Webový prohlížeč může komunikovat se vzdálenou LabView aplikací pomocí web services, neboli webových služeb. LabView dovoluje vytvořit speciální VI, které se chovají jako webové metody na straně serveru. Webový server je spuštěný na hostujícím počítači nebo na real-time zařízení, např.: cRIO. Tento způsob komunikace má výhodu ve schopnosti komunikovat s více zařízeními, která s LabView nemají nic společného. Musejí pouze umět komunikovat protokolem HTTP. Jednoduchost použití a efektivita hodně záleží na webovém prohlížeči a kvalitě naprogramované webové aplikace.

Data jsou vyměňována s webovými metodami za pomoci URL a klasických HTML požadavků [15]. Například, je možné poskytovat vstupní data z webového prohlížeče pro řízení ve webové metodě za použití standartního http požadavku GET nebo POST. Data jsou vrácena klientovi přes VI výstupní terminály. V tomto případě, když webový server obdrží požadavek, tak navrátí jakákoliv data, která jsou připojena na výstupní teminály v XML, HTML, JSON nebo čistém textu. Tvar výstupních dat lze zvolit v nastavení.

Vytvoření systému pro vzdálené měření na základě využití webových služeb obsahuje tři kroky. Vytvoření programu v LabView, který bude spuštěný na cRIO. Tento program by měl být schopný obsluhovat vstupy a výstupy reálné soustavy. To probíhá v časově kritické smyčce, zároveň se všemi potřebnými výpočty pro řízení. Další součástí programu je komunikační smyčka, která čte a uchovává data z časově kritické smyčky. To je vytvořeno pomocí RT FIFO funkcí nebo Queue funkcí. Další součástí systému jsou webové metody pro požadované vstupy a výstupy [16]. Tyto metody zároveň komunikují se základním programem na cRIO a to buď za pomoci shared variables nebo network streams. To závisí na ty přenášených dat. Třetím krokem je vytvoření webové aplikace, která posílá dotazy na vytvořené webové služby a umožňuje tím nastavení parametrů řízení a zobrazování naměřených dat. Existuje mnoho cest jak vytvořit grafické rozhraní ve webové prohlížeči. V této práci se řeší dva.

3.3.1 LabView web UI builder

LabView web UI builder je nástroj pro vytváření tenkých webových klientů pomocí grafického programování. Umožňuje vytvářet pohledná grafická rozhraní, která dovolují vzdáleně přistupovat k měřícím a řídícím aplikacím, založených na LabView, pomocí webového prohlížeče [17]. Pro komunikaci využívá webové služby a postup tvorby je následující. Načte se webová služba, se kterou je potřeba komunikovat, která automaticky zobrazí vstupy a výstupy (obrázek 3.4). Ty se podobně jako v LabView připojí na požadované indikátory nebo ovladače [18]. Je možné provádět i další výpočty, nebo nahrávání a ukládání souborů. Jsou tu ovšem i nějaká omezení a paleta neobsahuje všechny nástroje jako samotné LabView. Výhodou tohoto nástroje je jednoduchost a rychlost vytváření grafických rozhraní. Další výhodou je nutnost instalace pouze softwaru Microsoft Silverlight, který už ve většině případů na počítačích nainstalovaný je. Najdeme ale i jednu velkou nevýhodu. Pro využití všech možností nástroje a spuštění stránky samostatně je potřeba mít zakoupenou licenci, která stojí řádově desítky tisíc korun [19]. Proto byly hledány jiné postupy, které by dovolovaly vytvořit grafické rozhraní ve webovém prohlížeči pouze za pomoci volně dostupných nástrojů.



Obrázek 3.4 Grafické rozhraní programu LabView web UI builder

3.3.2 Boostrap a Google charts

Bootstrap je velice oblíbený HTML, CSS a JavaScript Framework pro vytváření „responzivních“ webových aplikací [20]. V podstatě tento Framework pokrývá větší nebo menší části úkolů, které dnes při vývoji webu musí kodér splnit. Jedná se o práci s typografií, tvorbou grafického návrhu, vytváření elementů uživatelského rozhraní a zároveň ošetřuje zobrazení na různých platformách. Hlavní výhoda spočívá právě v zobrazování na různých platformách. Velikost stránky i jednotlivých rámečků a elementů se mění podle rozlišení obrazovky. Přitom grafický návrh může být navržený od profesionálního grafika a lze jej do frameworku zabudovat.

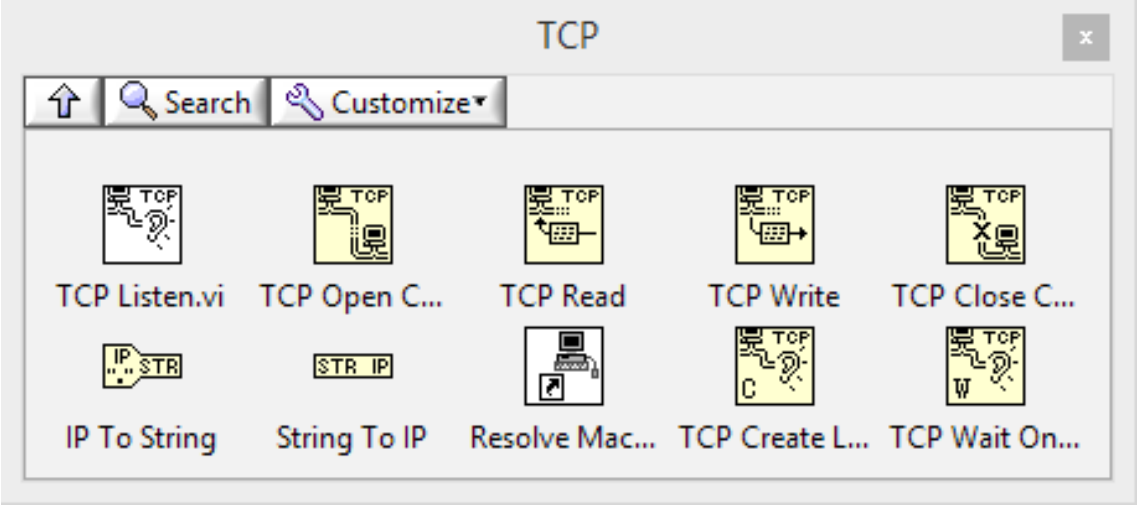
Pro vytvoření webové aplikace, se kterou by bylo možné vzdáleně řídit a sledovat reálný proces, se Bootstrap Framework velice hodí, protože zjednodušuje vytváření grafického rozhraní. Sám o sobě ale není dostačující, protože neposkytuje možnost zobrazovat měřená data do grafu. K tomu je potřeba využít další komponentu, kterou poskytuje např.: Google charts [21] nebo Highcharts [22]. Tyto komponenty dokáží zobrazovat data v interaktivních grafech, které, co se týče typu, mají mnoho různých možností od sloupcových, přes koláčové až po spojnicové. Google charts dokáží dokonce zobrazit graf ve stylu analogového tlakoměru apod. Grafický vzhled se může upravit pomocí CSS.

Jak by tedy taková webová aplikace komunikovala s programem na cRIO? Základem jsou webové služby, které jsou spuštěny na cRIO a AJAX (Asynchronous JavaScript and XML). AJAX je technologie, která umožňuje měnit obsah webových stránek bez nutnosti jejich kompletního znovunačítání [23]. Dovoluje tedy posílat asynchronní http požadavky a dané webové služby. Toho se právě využije pro komunikaci. Na pozadí stránky se AJAX bude dotazovat na měřená data na server. Ten odpovídá s naměřenými daty, které se následně zobrazí v grafu. Jiný požadavek může předávat vstupní data serveru apod. AJAX má výhodu, že odstraňuje nutnost načítání celých stránek, čímž teoreticky snižuje zátěž serveru i internetové sítě. Ovšem při nesprávné implementaci se může zátěž naopak zvětšit. AJAX usnadňuje práci v prohlížeči, kde (zejména u rychlejšího internetového připojení) se plynulost práce blíží desktopovým aplikacím. Naopak se zase ztěžuje návrh stránek a vyžaduje se více času a práce na implementaci AJAXu. Problémem také může být síťové zpoždění, takže by bylo mělo být jasně zobrazeno, že se asynchronní požadavek zpracovává, jinak by uživatel mohl způsobit něco, co neměl v úmyslu. Veškeré programování AJAXu a Google charts probíhá v JavaScriptu, je tedy potřeba vyhledat jen příslušné funkce a dodržovat zásady pro psaní kódu, např.: Same-origine policy [24], délka předávané URL apod.

Ve výsledku je to vhodná metoda pro vytvoření aplikace pro vzdálené měření, jelikož využívá bezplatných komponent. Na druhou stranu je třeba ovládat JavaScript a kódování v HTML a CSS.

3.4 TCP/IP PROTOKOL

IP (Internet protocol) a TCP (Transmission control protocol) jsou základní nástroje pro síťovou komunikaci. Protokol IP je základním protokolem na síťové vrstvě, který se používá v počítačových sítích a internetu. Sám o sobě neposkytuje spolehlivý přenos dat. Pouze na základě IP adres rozlišuje jednotlivá síťová rozhraní. Aby přenos dat po síti byl spolehlivý a nikde se žádná data neztrácela, byl vyvinut protokol TCP, který představuje transportní vrstvu. Díky tomuto protokolu mezi sebou server klient vytvoří spojení, po kterém si následně předávají data. Protokol si sám říká o opětovné poslání dat, upravení přenosové rychlosti, apod. aby nedocházelo ke ztrátám dat. Pro vzdálený přístup na základě protokolu TCP/IP se využívá skutečnosti, že LabView obsahuje funkce, pro jednoduché naprogramování komunikace pomocí protokolu TCP/IP. Veškeré potřebné funkce jsou v paletě functions >> Data Communication >> Protocols >> TCP (obrázek 3.5). Pomocí funkcí open connection, close connection, read a write lze sestavit klient i server. Takto vytvořený program by byl spuštěný na compactRIO a vzdáleně by se k němu přistupovalo z aplikace, která by byla puštěná na vzdáleném počítači. Aplikace pro počítač by mohla být napsaná např. v programovacím jazyku Java, který také obsahuje metody pro vytvoření spolehlivé TCP/IP komunikace. Tyto metody jsou obsaženy ve třídě java.net. Metody, které využívá serverová aplikace, jsou obsaženy ve třídě java.net. ServerSocket. Poskytují možnost obstarat port a naslouchat požadavkům klienta. Ve třídě java.net.Socket jsou metody, které umožňují vzájemnou komunikaci mezi serverem a klientem. Aplikace pro přístup na cRIO by také šla vytvořit jako webová aplikace. Nyní již existují API, které umožňují přímý přístup k TCP a UDP socketům, např.: TCP and UDP Socket API [25]. Vzdálený přístup na principu TCP/IP je velice univerzální metoda, která by poskytovala vysokou flexibilitu pro rozvoj do budoucna a nejlepší možnosti k úpravě podle nároků uživatelů i měřeného systému. Poskytuje vyšší přenosovou rychlost



Obrázek 3.5 Paleta TCP/IP

než komunikace vytvořená na základě shared variables nebo network streams. Naopak je zase náročnější na realizaci, protože se musí vytvořit komunikační rozhraní ve dvou různých programovacích jazycích, což pro zachování rychlosti a spolehlivosti nemusí být triviální úkol.

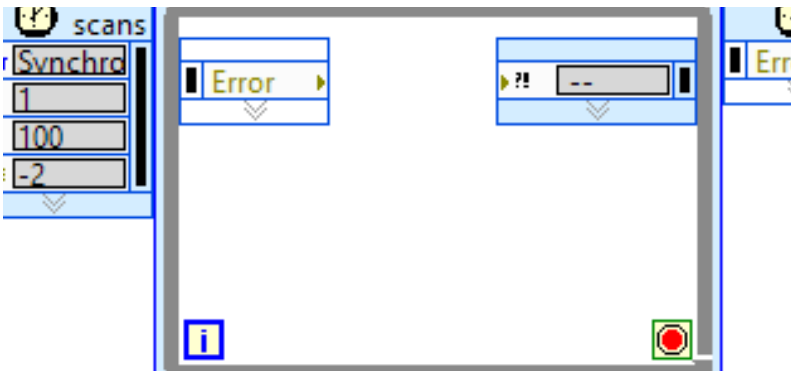
4 LabView program

V této kapitole bude popsán program, který zprostředkovává komunikaci mezi CompactRIO a připojeným systémem (Wattův regulátor). Základním kamenem programu jsou dvě paralelní smyčky while. První je časově kritická, která vykonává vždy přesně ve stanovený čas. Je deterministická. Druhá je nedeterministická a provádí se přibližně jednou za určený čas, ale pouze v případě, že procesor má dost volného času.

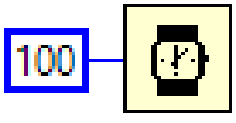
4.1 Časově kritická smyčka

Časově kritická smyčka je obdobou obyčejné smyčky while s rozdílem, že časově kritická smyčka se provádí vždy po zvoleném časové periodě. Tato vlastnost je v měření a řízení velice důležitá, protože většinou je potřeba sbírat data opakované ve stejném časovém rozmezí. To obyčejná smyčka while nezvládá a proto nastupuje smyčka časově kritická (obrázek 4.1).

V LabView se při programování smyšek while musí počítat s jednou významnou vlastností. Celý program se vždy vykonává co nejrychleji a snaží se využít 100% dostupného procesoru. To je všeobecně velmi dobrá vlastnost. Ne však v případě smyček. Všechny smyčky se také snaží vykonávat co nejrychleji a využitím 100% procesoru. Jakmile se vypočítá obsah smyčky, tak se hned v zápětí spustí další iterace. Obyčejnou smyčku while vůbec nezajímá, jak dlouho jí výpočet trval a počítá vždy dál, jakmile dostane možnost. Pokud jsou v programu i jiné výpočty, které musí procesor provádět, může se stávat, že na ně nezbývá čas, protože veškerý výkon procesoru si bere smyčka while. Může se to stávat např. u grafického rozhraní, kde interakce s tlačítky je velice pomalá. Proto se do smyček while vkládají funkční bloky Timer (obrázek 4.2). Díky těmto blokům se vykoná obsah smyčky a poté se čeká do té doby, než timer dosáhne požadované hodnoty. Minimální délka smyčky je hodnota uvedená u Timeru, ale smyčce stále nevadí, pokud se výpočet zpozdí a smyčka se provede déle.



Obrázek 4.1 Časově kritická smyčka



Obrázek 4.2 Timer nastavený na 100 ms

Pro aplikace, kde je požadováno vyloženě striktní vzor-

kování, je v LabView dostupná časově kritická smyčka [26]. Tato smyčka se provádí, vždy za určený čas a je řízena vnitřními hodinami hardwaru. U tohoto typu smyčky nastávají při výpočtu dvě možnosti. První možnost je, že se obsah smyčky spočítá rychleji, než je určená vzorkovací perioda. Tento stav je správný, veškeré výpočty proběhnou správně. Druhá možnost je, že výpočet obsahu smyčky trvá déle. Tento stav je nežádoucí a v podstatě by ani neměl nastat. Při kompilaci programu probíhá široká analýza, jestli je vůbec možné spočítat obsah smyčky za daný čas. Pokud se zjistí, že to možné není, tak kompilace skončí s chybou. Pokud kompilace proběhne v pořádku, stále se může stát, že časově kritická smyčka neproběhne v čas. To například může nastat v případě, že se ve smyčce přistupuje k souborům na permanentní pamětní. Permanentní paměť může mít různou dobu odezvy v závislosti na vytížení, a proto může vykonání smyčky trvat déle než obvykle. V tomto případě smyčka ohlásí chybu a další běh programu je dán tím, jak se chyba zpracuje.

Je vidět, že při vytváření časově kritických smyček musí být programátor velice opatrný, co všechno do smyčky vkládá a musí počítat s tím, že má omezený výpočetní výkon v závislosti na vzorkovací periodě smyčky. Pokud se kompilace nedaří, je potřeba buď omezit výpočty, nebo zvýšit vzorkovací periodu.

Ze stejných důvodů také vyplývají jistá doporučení, co všechno do časově kritické smyčky umisťovat. Je velice přínosné v této smyčce přistupovat ke vstupům a výstupům na cRIO. Dále je možné provádět veškeré aritmetické výpočty a měřenými daty. To není žádný problém. Naopak se nedoporučuje přistupovat k souborům na permanentní paměti nebo provozovat síťovou komunikaci. Tyto operace jsou nedeterministické a pravděpodobně by způsobovaly problémy.

4.2 Scan mód

Když se vytváří projekt pro cRIO, jsou na výběr dvě možnosti: tvořit program v FPGA módu nebo scan módu. Tyto dva módy se zásadně liší v přístupu na FPGA. V FPGA módu je nutné veškerou komunikaci s FPGA ručně naprogramovat. Vývoj takových programů je delší, složitější a náročnější. Výhodou naopak je možnost dosažení vyšší výpočetní rychlosti a vzorkovací frekvence (jednotky kHz). Naproti tomu při využití scan módu, se doba vývoje značně zkracuje, protože o některé úkoly při propojování s FPGA je automaticky postaráno. Veškeré vstupy a výstupy jsou automaticky spojeny s příslušnými I/O (input/output) proměnnými. To umožňuje jednoduchý zápis a čtení dat bez nutnosti programovat FPGA a zdlouhavého kompilování. Toto zjednodušení je vykoupeno nižší frekvencí, která je omezena na 1kHz. Porovnání frekvencí v závislosti na počtu kanálů (vstupy a výstupy) je možné vidět v tabulce 4.1.

Scan mód využívá dvě technologie: NI scan engine a RIO scan interface. RIO scan

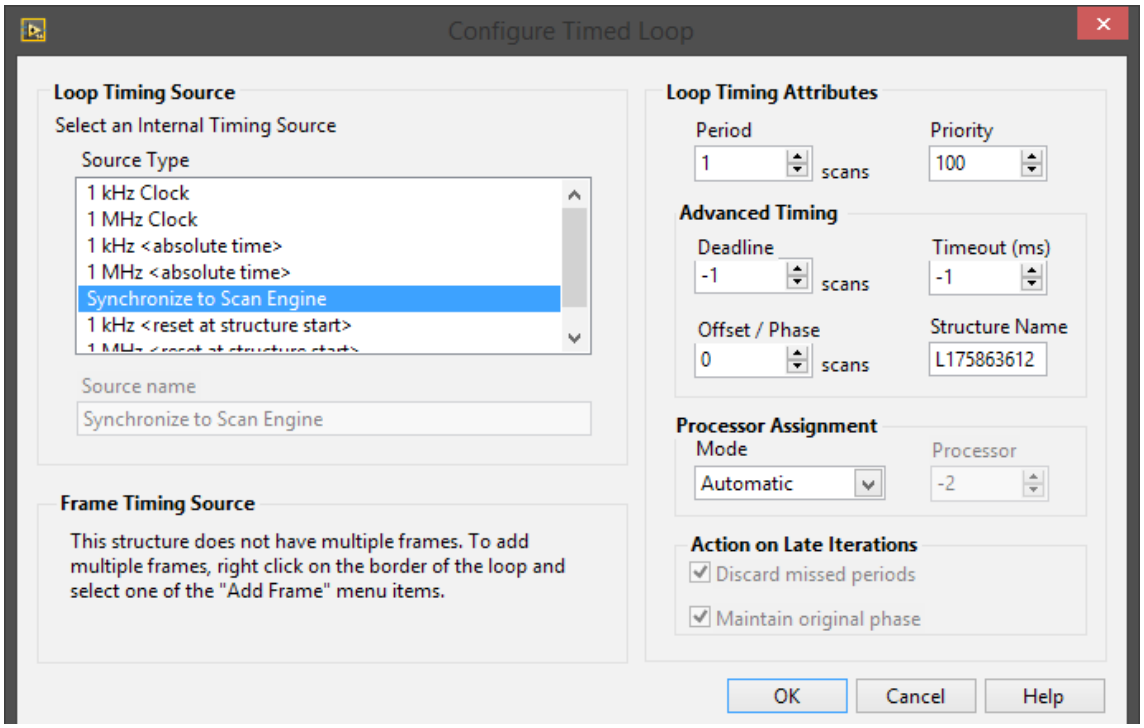
Počet připojených I/O	Scan mód (Hz)	FPGA mód (Hz)
1	1000	7100
16	1000	3500
80	399	974

Tabulka 4.1 Porovnání možných dosažitelných frekvencí u Scan a FPGA módu [28]

interface je skupina nástrojů, vyvinutých National Instrument, které jsou stažené v cRIO a starají se o detekci modulů, časování, synchronizaci a komunikaci. V hardwarově řízené časové smyčce se aktualizují hodnoty na vstupech a výstupech, které tam jsou fyzicky měřené. Jsou používány dva DMA (direct memory access) kanály pro transport dat z FPGA do RTOS [29].

NI scan interface je komponenta LabView Real-time, která má ještě větší prioritu než časově kritické struktury. Pokaždé když interface dokončí poslední skenování vstupů a výstupů, LabView automaticky přidá I/O proměnné do globální paměti scan enginu a zároveň aktualizuje jejich hodnoty. Každá I/O proměnná lze nastavit, aby využívala skenování nebo přímý přístup. Přímý přístup vždy přepisuje skenování a následně komunikuje přímo s I/O zařízením. Pokud se kombinuje FPGA mód a scan mód, tak se tyto dvě části zkompilují do jednoho souboru a jsou spuštěny na cRIO. Pokud se všechny moduly ve scan módu přepíší přímým přístupem, tak se RIO scan interface nekompiluje a tím se šetří místo v paměti.

Pokud se tedy používá scan mód, tak se v časové kritické smyčce nastaví synchronizace se NI scan engine. Vzorkovací frekvence se nastavuje přímo v nastavení projektu (properties) v záložce Scan engine. Jak bylo řečeno výše, maximálně lze dosáhnout 1kHz,



Obrázek 4.3 Nastavení časově kritické smyčky

což odpovídá vzorkovací periodě 1ms. Pokud je potřeba vyšší vzorkovací frekvence nebo I/O modul scan mód nepodporuje, je nutné použít FPGA mód. Ke vstupům a výstupům se přistupuje v okně projektu pod záložkami jednotlivých modulů. Proměnné se pouze přetáhnout do blokového schéma a hned je možné je používat.

4.3 Komunikační smyčka

Jelikož časově kritická smyčka není vhodná pro přístup k souborům na permanentní paměti nebo síťové komunikaci, musí se použít obyčejná smyčka while. Ta není řízená hardwarovými hodinami, a proto probíhá nedeterministicky. Zde tedy problém se zapisováním do souborů nebo komunikací přes protokol TCP/IP není. Stále se musí počítat s tím, že smyčka while se vykonává tak často, jak jen může, a proto se nesmí zapomenout na časovač, který vykonávání smyčky zpomalí. Může však probíhat i déle.

4.4 Předávání dat mezi smyčkami

Nyní však vyvstává další problém. Naměřená data ze soustavy jsou v časově kritické smyčce, ale je potřeba je zapsat do souboru nebo odeslat přes internet. A naopak, příchozí řídící signály jsou v komunikační smyčce a je potřeba je předat na vstupy systému. Jelikož neexistuje triviální řešení na předávání dat mezi paralelními smyčkami, je v LabView vytvořeno několik funkčních postupů, které se využívají podle charakteru přenášených dat.

Nejjednodušší způsob předávání dat mezi smyčkami je pomocí network shared variable. Bohužel se nehodí pro předávání dat často za sebou. Nejlépe se využijí na předávání řídicích parametrů nebo u typu dat, která se měří méně často, např.: měření teploty jednou za pět minut. Pokud se data tvoří řádově v milisekundách nebo i méně, doporučuje se využít jiné metody, které jsou popsané níže. Výhodou je jednoduchá implementace a také možnost sdílet tyto proměnné rovnou přes internet nebo místní síť.

Fronta (Queue) se využívá pro komunikaci uvnitř programů LabView. Udržuje pořadí prvků ve formě FIFO (first in, first out). Je možné to přirovnat ke klasické frontě v obchodě. Kdo dříve přijde, ten bude dříve obsloužen. Na stejném principu se pracuje i s daty.

Fronty jsou užitečné při datovém modelu Producent a konzument. Tato situace vzniká právě například u dvou paralelní smyček jako v tomto případě. V časově kritické smyčce se tvoří data (producent) a komunikační smyčka je používá (konzument). Výhoda front v tomto případě je, že rychlost obou smyček nemusí být stejná. Jestliže konzumace je pomalejší než produkce, fronta se naplní a producent bude nucen počkat, dokud konzument

některé prvky z fronty neodebere.

Na rozdíl od pole není možné přistupovat k jednotlivým prvkům ve frontě náhodně. Je to jen buffer, který dovoluje přidávat a odebírat prvky ve frontě. Jediná možnost jak zpřístupnit všechny prvky ve frontě je odebrat je z ní jeden po druhém dokud fronta není prázdná.

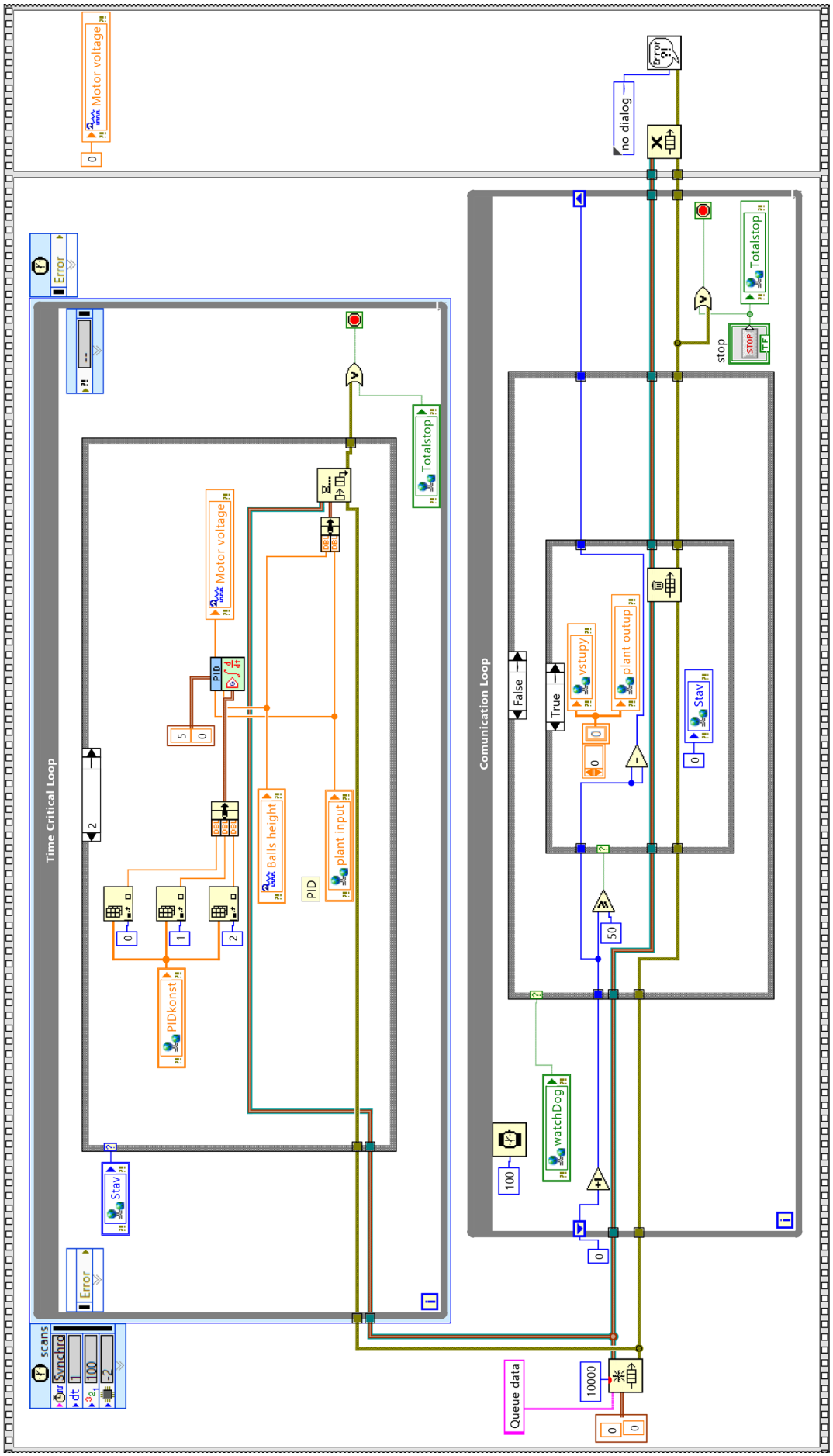
Ačkoliv byl network stream vyvinutý pro komunikaci mezi dvěma různými programy, je možné je využít i na přenášení dat mezi paralelními smyčkami. Chová se to velice podobně jako fronty, ale je to postaveno na TCP/IP komunikaci. Umožňují přenos dat mezi smyčkami, dokonce i na jiných strojích. Je to bezeztrátová komunikace, která je odolná proti přerušení spojení. Některé typy dat nejsou při použití network stream dovolena. Jedná se o referenci na objekt. Je to z důvodu komunikace mezi jinými stroji, protože když se přenese reference na jiný stroj, je velice pravděpodobné, že nebude dávat žádný smysl, a proto jsou zakázané. Network stream je optimalizovaný na přenos velkého množství dat a proto posílání malého množství, jako jsou řídicí pokyny, mohou mít značné zpoždění.

4.5 Sestavení programu pro cRIO

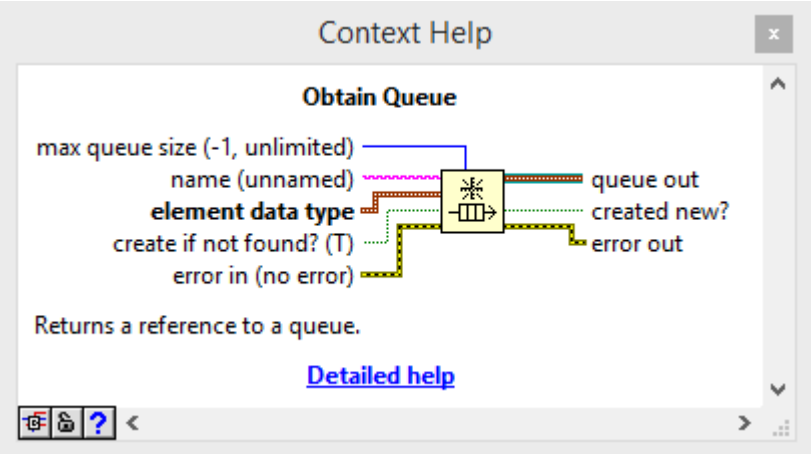
Na obrázku 4.4 je vidět část blokového diagramu programu. Hlavní části jsou dvě paralelní smyčky. Jedna je časově kritická, druhá je komunikační. Ke smyčkám se přistupuje v paletě funkcí pod záložkou Structures. Obě smyčky jsou spojené frontou, která se tvoří funkcí Obtain queue (obrázek 4.5). Tato funkce má několik parametrů, které mohou být pochopeny jako inicializace nebo prvotní nastavení. Jediným povinným prvkem je nastavení typu přenášených dat. Vzhledem k typu dat se přenáší klastr složený ze dvou hodnot typu double. Jedna hodnota představuje vstup do soustavy a druhá měřenou odezvu v daném okamžiku. Oba signály jsou důležité pro správné zobrazení. Dále se nastavuje buffer na 10000 prvků, což několikrát více než je ve skutečnosti potřeba, ale nějaká rezerva tam být musí. Také se zde může nastavit jméno fronty, které ulehčuje orientaci v programu.

Časově kritická smyčka má nastavenou periodu vzorkování shodnou se scan interface. V tomto případě byla zvolena vzorkovací perioda 10 ms. Použité vstupy a výstupy jsou přístupné na modulu NI 9381. Vstup do cRIO je AI7 pojmenovaný jako Balls height. Pro upřesnění: vstup do cRIO je výstup z řízené soustavy. Naopak výstup z cRIO je A04 pojmenovaný jako Motor voltage a je to vstup do měřené soustavy. Na veškerých těchto vstupech a výstupech se měří napětí a jsou hardwarově omezeny rozsahem 0-5V.

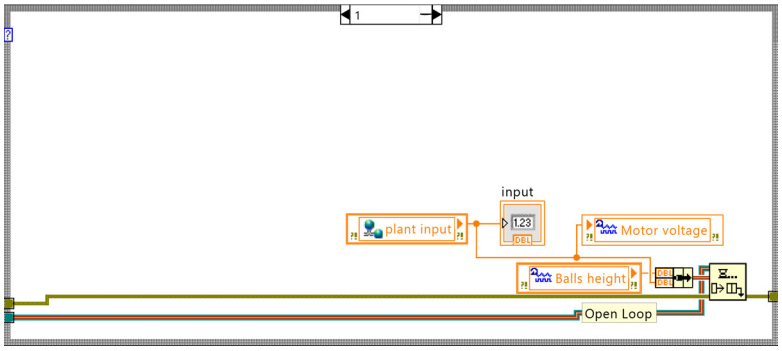
V časově kritické smyčce, je důležitá struktura case. Je to v podstatě reprezentace stavového automatu. Nyní jsou zde tři stavy. Nultý stav je takzvaný safe state, neboli bezpečný stav, do kterého se program dostane pokaždé, když se dlouho nic neděje.



Obrázek 4.4 Celkový blokový diagram prgramu na cRIO



Obrázek 4.5 Funkce Obtain Queue



Obrázek 4.6 Stav otevřené smyčky

To je z důvodu bezpečnosti a prodloužení životnosti měřeného systému. V tomto stavu se výstup napětí na motoru nastavuje nula a zároveň se nastavují nuly v komunikačních proměnných.

První stav je open loop (obrázek 4.6), neboli otevřená smyčka, v níž se provádí přímé měření reakce výstupu na vstup. Vstupní signál, který je reprezentovaný sdílenou proměnnou nazvanou plant input, přenáší zadanou hodnotu na výstup Motor voltage. Ze vstupu Balls height se rovnou, bez žádných další výpočtů, snímá odezva. Z obou signálů se pomocí funkce Bundle tvoří klastr, který se pomocí funkce Lossy enqueue element (obrázek 4.7) vkládá do fronty a přenášen do druhé smyčky. Funkce Lossy enqueue element je funkce pro vkládání prvku do fronty s jednou speciální vlastností. Když je buffer fronty plný, smyčka se nezastaví, ale funkce Lossy enqueue element vezme nejstarší prvek fronty a smaže. Na vzniklé místo může poté vložit prvek nejnovější. Tato varianta byla zvolena z důvodu, aby se časově kritická smyčka nikdy nezastavil. Je lepší, když se data začnou ztrácet, než aby se zastavil celý program.

Druhý stav je regulace pomocí PID regulátoru (obrázek 4.8). Jako PID regulátor se používá funkce, která je dostupná z palety pod záložkou Control&Simulation >> PID. Na obrázku 4.9 je vidět zjednodušené blokové schéma regulátoru. Regulační odchylka e(k) se počítá jednoduše jako rozdíl regulované y(k) a referenční veličiny w(k).

$$e(k) = y(k) - w(k) \tag{5}$$

Proporcionální složka se počítá jako regulační odchylka násobená zesílením regulátoru Kc.

$$u_p(k) = K_c * e(k) \tag{6}$$

Integrační složka regulátoru je aproximován lichoběžníkovou aproximací známou také jako Tunstinova aproximace. Výpočet této aproximace je popsán rovnicí 7, kde deltaT je vzorkovací perioda a Ti je integrační časová konstanta

$$u_i(k) = u_i\,(k-1) + \frac{K_c}{T_i}\left(\frac{e(k) - e(k-1)}{2}\right)\Delta T \tag{7}$$

Derivační složka se nevypočítává z regulační odchylky, ale rovnou zregulované veličiny. Důvod je takový, že referenční veličina může dělat skoky hodnot. To se na derivační složce projevuje velikými derivačními špičkami. Následující rovnice představuje výpočet derivační složky, která se vyhýbá derivačním špičkám.

$$u_d(k) = -K_c\left(\frac{T_d}{\Delta T}\right)\big(y(k) - y(k-1)\big) \tag{8}$$

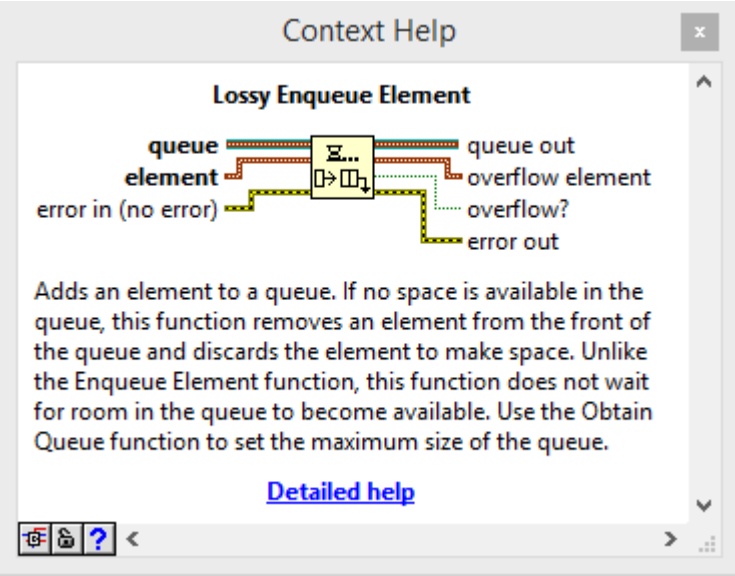
Výstup regulátoru je pak jednoduchý součet včech složek regulátoru.

$$u_k(k) = u_p + u_i + u_d \tag{9}$$

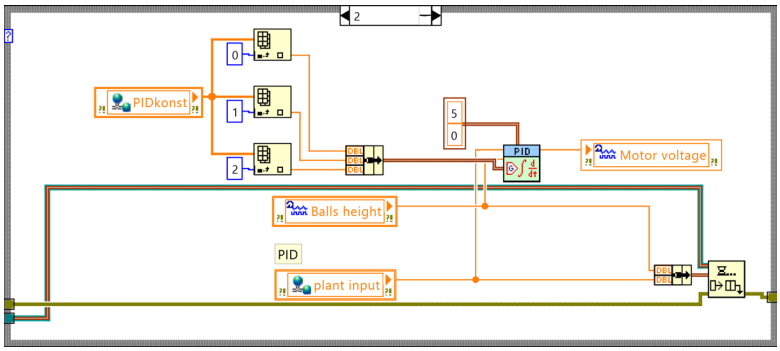
Regulátor obsahuje možnosti omezení výstupu, tedy akční veličiny, a zároveň poskytuje ochranu proti fenoménu windup. Jelikož je akční veličina fyzicky omezena na 0-5V, Bylo vhodné toto omezení nastavit i u regulátoru. Při výpočtu potom regulátor s těmito omezením počítá a správně implementuje windup ochranu. Jednotlivé koeficienty regulátoru (PID gains) je možné měnit kdykoliv za běhu, protože při změně se regulátor snaží o beznárazové přepnutí na nové koeficienty.

V programu je na výstup (output) regulátoru připojena proměnná Motor voltage, tedy akční veličina, která řídí soustavu. Na konektor set point je připojena proměnná plant input, kterou nastavuje uživatel. Uživatelem zadávané koeficienty regulátoru jsou spojeny do klastru a připojeny na konektor PID gains. Posledním zapojeným konektorem je proces variable, ke kterému je připojena proměnná Balls height, protože tento signál chceme řídit. Funkce Lossy enqueue element se opět stará o zapsání nového vzorku do fronty. Podle počátečního nastavení fronty je potřeba signály Balls height a plant input spojit funkcí bundle do klastru. Důležité je poznamenat, že integrační a derivační časová konstanta je v minutách, proto je nutné tyto konstanty při klasických návrzích regulátoru přepočítávat.

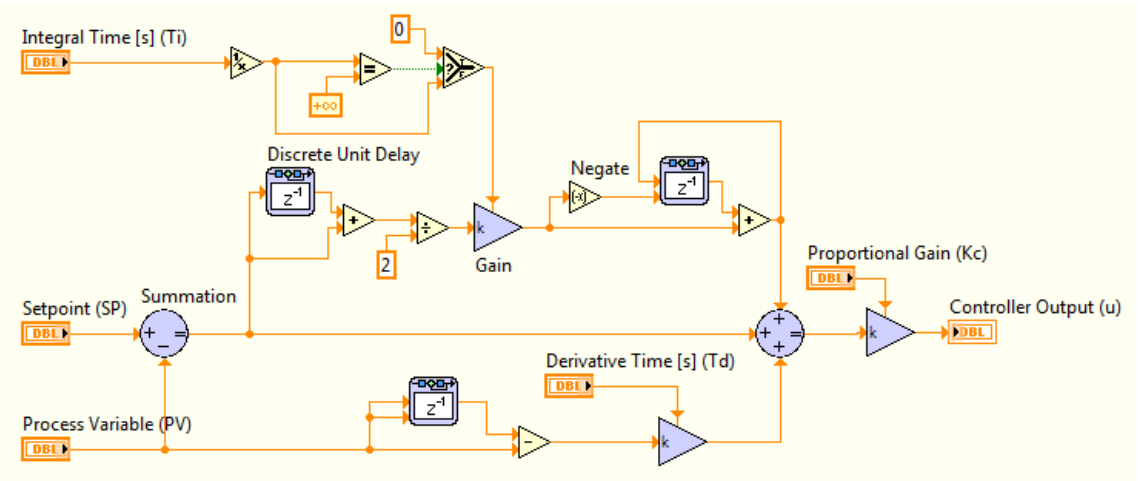
Komunikační smyčka je spuštěna s časovačem nastaveným na 100 ms. To znamená, že smyčka proběhne maximálně jednou za 100ms. Hlavní součástí je struktura case,



Obrázek 4.7 Funkce Lossy enqueue element



Obrázek 4.8 Stav uzavřené smyčky (PID regulace)

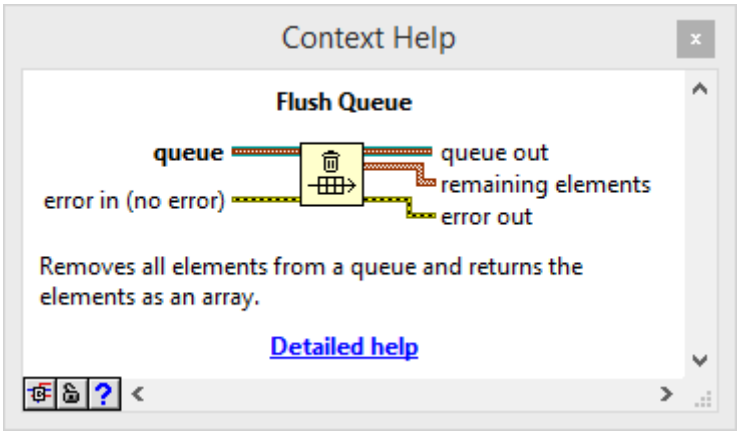


Obrázek 4.9 Schéma PID regulátoru

která je řízená přes softwarový watchDog. Je to sdílená proměnná typu boolean. Tento watchDog se nastavuje do hodnoty pravda pokaždé, když přijde dotaz na čtení dat (bude vysvětleno v kapitole 6.2). Ve větvi pravda se nejprve hodnota proměnné watchDog změní zpět na nepravda a souběžně s tím se provede přečtení dostupných dat z fronty funkcí Flush queue (obrázek 4.10). Tato funkce je použita jako alternativa k funkce Dequeue element, protože na rozdíl od ní nevrací pouze jeden prvek z fronty ale všechny. V době čtení požadavku je potřeba přečíst všechny prvky fronty najednou a předat je dále. Právě proto je použita funkce Flush queue. Jelikož funkce Flush queue vrací pole klastrů a pro další přenos je potřeba mít data ve formátu dvou polí, tak je použita struktura for pro převod. V ní se postupně vezme každý prvek pole (funkce Index Array) a z klastru se vytvoří dvě proměnné typu double (funkce Unbundle). Tyto proměnné se následně skládají za sebe do pole. O to se stará speciální funkce při výstupu ze struktury for. Jmenuje se Auto-Indexed tunnel a v každém cyklu automaticky zařadí proměnnou na konec vznikajícího pole. Tato pole poté předávají svůj obsah sdíleným proměnným (plant outup, vstupy). Struktura for proběhne tolikrát jako je délka pole předané z funkce Flush queue. Délka pole se získá funkcí Array size.

Součástí komunikační smyčky také je počítadlo, které hlídá, aby komunikační smyčka neproběhla více než padesát krát bez obdržení dotazu na čtení dat. V praxi to znamená, že pokud uběhne přibližně 5 sekund od posledního dotazu na předání dat, program se přepne do tzv. safe state (bezpečný stav), protože předpokládá, že došlo k ukončení měření nebo nastala chyba ve spojení. V bezpečném stavu jsou nulovány sdílené proměnné vstupy a plant outup, je vyčištěna fronta od všech prvku funkcí Flush queue a je vynulované počítadlo. Bezpečný stav se zpětně mění i v časově kritické smyčce, kde se nuluje proměnná Motor voltage.

Poslední součástí je Flat sequence struktura, která má za úkol vykonávat části programu po sobě. V tomto programu má pouze dvě části. V první jsou veškeré smyčky a inicializace. Ve druhé jsou bloky, které zaručují ukončení programu bez chyb a v bezpečném stavu. Kdyby to tak nebylo, může se stát, že poslední hodnotou na vstupu by nebyla nula. Program by se vypnul, ale na vstupu by stále nebyla nula a systém by byl stále v provozu, dokud by se cRIO nerestartovalo, nebo se nespustil nový program.



Obrázek 4.10 Funkce Flush queue

5 Bootstrap a HTML stránka

Aby uživatel mohl používat funkce cRIO je potřeba vytvořit uživatelské rozhraní, které by to dokázalo. V kapitole 3 bylo zmíněno několik variant, jaký způsobem toho dosáhnout. V této práci byl zvolen způsob pomocí bootstrap a google charts. Hlavní důvod výběru tohoto přístupu je ten, že se potom jedná o tenkého klienta. To znamená, že uživatel nemusí nic instalovat. Otevře si pouze internetový prohlížeč a může rovnou začít měřit. Příprava je prakticky minimální a navíc se do počítače nemusí zbytečně instalovat programy na jedno použití.

5.1 Proč bootstrap?

Bootstrap je balíček CSS souborů a souborů java script, které zjednodušují práci při vytváření HTML stránek. Základní funkcí je rozdělení stránky do sloupců [30]. Pro aplikaci se musí použít rámec , který je označený třídou row (class="row"). To značí jeden řádek. Do tohoto rámce se poté vkládají další rámce, které slouží jako jednotlivé sloupce. U každého sloupce se nastavuje třída a velikost. Bootstrap obsahuje 4 třídy:

- xs (mobilní telefony)
- sx (tablety)
- md (počítačové obrazovky)
- lg (velké počítačové obrazovky)

Velikosti se mohou různě kombinovat až do maximálního počtu sloupců 12. Je tedy možné mít tři sloupce po čtyřech, šest po dvou, nebo dva po čtyřech a osmi. Tento systém je „responzivní" a sloupce se automaticky přerovnávají při každé změně velikosti obrazovky. Proto jsou zde 4 třídy.

Další funkce, kterými bootstrap disponuje, je např. tvorba interaktivního menu. Jsou zde již připravené funkce v javascriptu. Menu potom může být statické na začátku stránky, nebo se může dynamicky posouvat se stránkou (třída navbar a ostatní). Nechybí funkce na vyskakovací menu (třída dropdown a podobné). Stejně tak jsou zde třídy pro tlačítka a tabulky. V podstatě veškeré HTML tagy mohou být opatřeny bootstrap třídou a tím změnit jejich vzhled a funkčnost. Pořád se však jedná o vzhled, který je implicitně v bootstrap nastavený. Jsou to neutrální odstíny bílé a černé s občasnými jinými informačními barvami (červená – nebezpečí, žlutá – pozor, modrá - informace). To ale neznamená, že by to nebylo možné změnit. Jelikož je bootstrap balíčkem CSS a javascriptových souborů není problém je vzít a předělat podle požadavků. Zde jsou pak nekonečné možnosti, co se vzhledu týče a omezení klade jen vlastní představivost.

5.2 Graf na webové stránce

Aby celý systém nějakým způsobem fungoval je potřeba naměřená data na cRIO zobrazovat. Čistá naměřená data jsou v textové podobě značně nečitelná, a proto se přistoupilo k použití Google charts jako zobrazovacímu médiu.

Google charts je nástroj, který umožňuje zobrazovat jakákoliv data v interaktivním grafu. K dispozici je mnoho druhů grafů. V této práci byl použitý spojnicový graf, ve kterém se nejlépe zobrazuje tento charakter dat. Základní předpokladem fungování je nahraný javascript v HTML kódu. Ten se nahrává pomocí příkazu:

```
<script src="https://www.google.com/jsapi"></script>
```

Na vykreslení grafu na stránku se používá funkce:

```
chart.draw(data, options);
```

Proměnná options je nastavení tabulky. Lze nastavit velikost, pojmenovat osy, vytvořit legendu, měnit barvu, atd. Proměnná data představuje zobrazená data. Je to vlastně tabulka, která obsahuje sloupce a řádky. První sloupec představuje osu x. Většinou se jedná o číslo vzorku nebo čas. Každý další sloupec představuje hodnotu dat na ose y. Z toho vyplývá, že datová tabulka musí mít nejméně dva sloupce. Sloupce se přidávají příkazem:

```
data.addColumn('number', 'X');
```

První argument funkce představuje typ a druhý název. Potom se do tabulky přidávají řádky. Zde se již přidávají jednotlivá data. Když je tabulka o třech sloupcích, tak se řádek přidává příkazem:

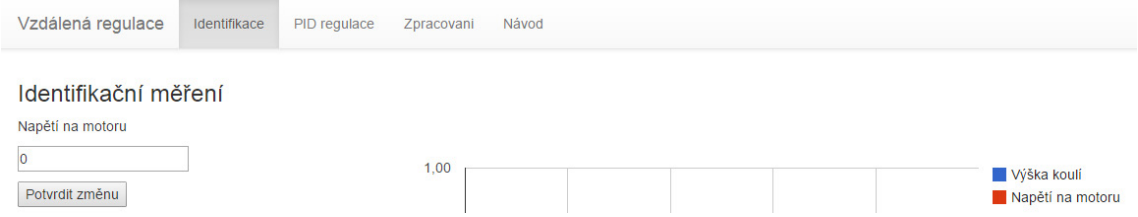
```
data.addRow([3, 23, 31]);
```

První prvek pole představuje osu x a zbylé představují osu y. Graf tedy bude obsahovat dvě čáry. Je možné také použít příkaz na přidání více řádků najednou:

```
data.addRow([[4, 21, 34], [5,26,25]]);
```

5.3 Složení HTML stránky

Na začátku je menu, které přepíná mezi jednotlivými stránkami (obrázek 5.1). Je vytvořené pomocí bootstrap a umožňuje přidání dalších záložek. Tomu odpovídá kód:

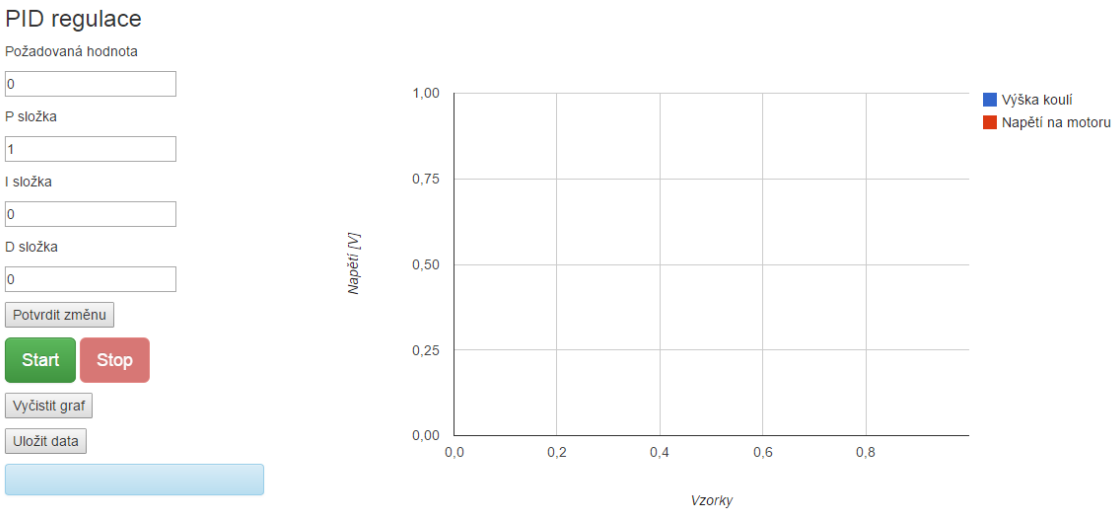


Obrázek 5.1 Hlavní menu

```
<nav class="navbar navbar-default navbar-fixed-top">
      <div class="container">
        <div class="navbar-header">
          <a class="navbar-brand" href="">Vzdálená regulace</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li class="navig1 active" data="1">
               <a href="">Identifikace</a>
            </li>
            <li class="navig1" data="2" >
               <a href="">PID regulace</a>
            </li>
            <li class="navig1" data="3"><a href="">Zpracovani</a>
            </li>
            <li class="" data="4"><a href="navod.html">Návod</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
```

Jedná se o vytvořené menu pomocí tagů <ul> a <li>, které primárně slouží k vytváření seznamů. Když jsou tyto tagy doplněné o příslušné třídy a styly, začnou tvořit menu. Zde je vytvořené třídami bootstrap. Atribut data je přidaná informace, která umožňuje rozpoznání mezi jednotlivými záložkami.

Následuje další rámec, který je rozdělen do sloupců v poměru 2:10 (obrázek 5.2). V levém úzkém sloupci jsou ovládací prvky (tlačítka, formuláře). Celý pravý sloupec je vyhrazený pro zobrazení grafu. Při přepínání mezi jednotlivými záložkami se vždy nenačítá celá stránka. Pouze se chovávají a objevují rozdílné ovládací prvky, které nejsou u všech měření stejné. Toho je dosaženo javascriptovou funkcí:



Obrázek 5.2 Rozložení webové stránky

```
$(document).ready(function() {
  $(".navig1").click(function(e) {
    e.preventDefault();
    var k1 = $(this).attr("data");
    $(".levemenu[data=" + k1 + "]").show();
    $(".levemenu[data!=" + k1 + "]").hide();
    if(k1==3){
      $("#start").hide();
      $("#stop").hide();
      $("#send").hide();
      loadedData = chart_data.toJSON();
    }else{
      $("#start").show();
      $("#stop").show();
      $("#send").show();
    }
    $(".navig1[data!=" + k1 + "]").removeClass("active");
    $(this).addClass("active");
    Stop();
  });

  $(window).resize(function(){
      drawChart(chart_data, 0);
  });
});
```

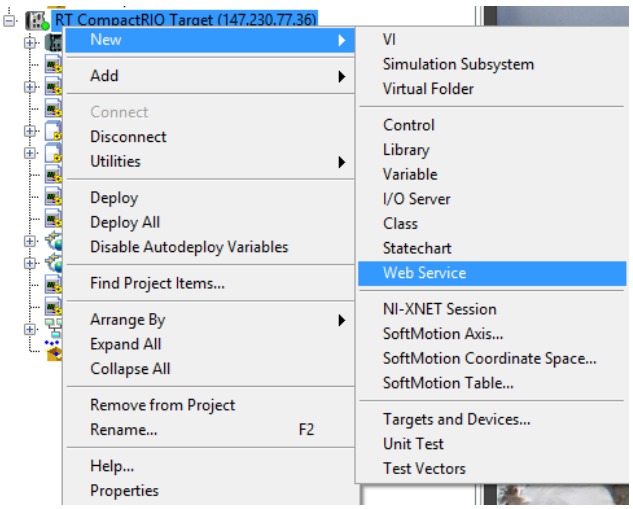
S webovou stránkou nelze bezpečně manipulovat, pokud není celá připravená. Proto je na prvním řádku kódu funkce, která přesně toto zajišťuje. Spouští se vždy až po načtení celé stránky. Na druhém řádku je funkce, která kontroluje, zda nebylo kliknuto na komponentu s třídou navig1 (odkazy v hlavním menu). V jquery se třídy označují tečkou před názvem, id se označuje křížkem (hash). Pokud je tedy na tuto komponentu kliknuto, začne se funkce provádět. Nejprve se načte dodatečná informace z přidaného atributu data. Díky tomu se rozezná přesná poloha komponenty. Potom se provádí zobrazení nebo schování další komponenty, v tomto případě označené třídou levemenu (rámec se vstupy). Tento rámec má také přidaný atribut data, který odpovídá atributu data u třídy navig1. Hodnoty se porovnají a v případě shody se rámec zobrazí pomocí funkce show(). Všechny ostatní rámce se schovají pomocí funkce hide(). Pokud se atribut rovná číslu 3, tak to znamená, že se kliklo na odkaz zpracování. Na této stránce jsou potřeba ani tlačítka Start, Stop a Potvrdit změnu, a proto jsou všechny také schovány. Následně se pomocí funkcí removeClass() a addClass() nastavuje třída active, která v hlavním menu představuje vybranou stránku. Při přepnutí stránek se také zastavuje veškeré posílání požadavků (podrobněji v kapitole 6.2), aby nedocházelo k vícenásobnému spuštění. Ve funkci ready() je také obsažena funkce resize(), která upravuje velikost grafu při každé změně velikosti prohlížeče.

6 AJAX A WEBOVÉ SLUŽBY

Nyní už je vytvořený program na cRIO, který komunikuje s měřenou soustavou, a grafické rozhraní, které umožňuje řízení uživatelem a prezentaci veškerých dat. Komunikace mezi těmito dvěma částmi je vytvořena webovými službami a asynchronními http požadavky na tyto služby.

6.1 TVORBA WEBOVÝCH SLUŽEB

Webovou službu je v LabView možné vytvořit jak pro místní počítač, tak i pro vzdálené připojené zařízení. Tvoří se kliknutím pravým tlačítkem myši na záložku zařízení v okně projektu [31]. Poté se vybere New >> Web service (obrázek 6.1). Záložka webové služby obsahuje pouze startovací VI a Web resources, kde jsou obsaženy veškeré http metody. Dále je možné pravým kliknutím na webovou službu přidat veřejný nebo skrytý obsah [32]. Rozdíl mezi veřejným a skrytým obsahem je v tom, že veřejný je přístupný z jiných webových adres. Naproti tomu skrytý obsah je přístupný pouze z cRIO. Webové stránky tedy budou umístěny ve veřejném obsahu, kdežto funkce pro server budou obsaženy ve skrytém obsahu, aby se k nim nikdo nemohl dostat a nebyly zneužity. Výhoda použití veřejného obsahu se projeví například u souvisejících webových stránek, které se nahrají do cRIO při spuštění webové služby a nemusí se to dělat zvlášť ručně. Další výhodou při využívání veřejného obsahu je, že se tím dá vyhnout omezení Same origin policy [24]. Toto omezení je další bezpečnostní omezení pro webovou komunikaci. Toto omezení umožňuje skriptu, obsaženého v jedné stránce, přistupovat k datům na stránce druhé, pouze pokud mají stejný „počátek“. Ten je definován jako protokol, jméno domény a číslo portu. Pokud se některá z těchto částí liší, tak webový prohlížeč automaticky blokuje přístup. Příklady uvedeny v tabulce 6.1, jsou porovnávány s adresou „http://www.priklad.cz/stranka.html“. Z tohoto hlediska může být největším problémem při spouštění webové služby na cRIO právě rozdílné porty.



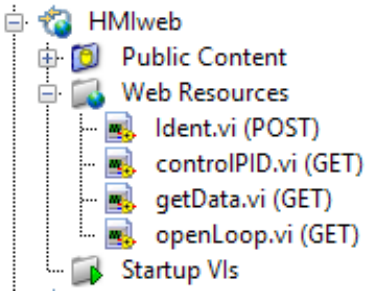
Obrázek 6.1 Vytvoření webové složby

Problémy se Same origin policy se vyskytly u prvotního vývoje. Veškeré webové služby byly přístupné na portu 8001, ale všechny webové stránky a skripty se spouštěly

Porovnávaná adresa	Výsledek	Důvod
http://www.priklad.cz/stranka2.html	Úspěšný	Stejný protokol, doména i port
http://www.priklad.cz/slozka/stranka3.html	Úspěšný	Stejný protokol, doména i port
http://www.priklad.cz:8080/stranka4.html	Neúspěšný	Rozdílný port
https://www.priklad.cz/stranka5.html	Neúspěšný	Rozdílný protokol
http://novy.priklad.cz/stranka6.html	Neúspěšný	Rozdílná doména

Tabulka 6.1 - Příklady Same origin policy

z klasického portu 80. Důvod byl takový, že všechny soubory byly nahrávány přes FTP do složky C:/ni-rt/system/www/. Problémy se vyřešily využitím veřejného obsahu (obrázek 6.2). Jsou v něm nyní obsaženy všechny webové stránky, kaskádové styly i javascript soubory. Pod záložkou Web resources jsou všechny http metody, které tvoří komunikaci. Pro každou metodu je vyhrazené samostatné VI, které se při kompilaci přemění na http metodu. Vlastností každého VI je typ http metody, která se bude kompilovat. Typy jsou uvedeny v závorkách za jmény VI. Vlastnosti jednotlivých typů metod jsou uvedeny v tabulce 6.2.



Obrázek 6.2 Webová služba

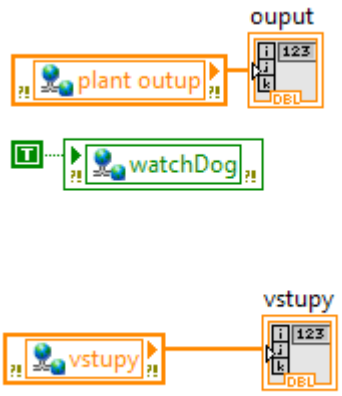
6.1.1 Metoda getData.VI

Tato metoda (obrázek 6.3) se využívá pouze pro odesílání dat z cRIO do webové stránky. Při každém tomto dotazu je nastavena proměnná watchDog na pravda. To spustí čtení dat z fronty (popsáno v kapitole č. 4.5). Tato přečtená data jsou předána proměnným plant outup a vstupy. Tyto proměnné jsou připojeny na indikátory a ty na vnější konektory VI. LabView potom samo namapuje konektory jako proměnné, které se mají předávat spolu s požadavkem.

GET	Požadavek na určitý objekt se zasláním případných dat. Celkově nejpoužívanější.
PUT	Nahraje data na server. Používá se velmi zřídka, protože pro nahrávání dat na server se běžně používá FTP
POST	Odesílá uživatelská data na server. Používá se, pokud je délka data příliš velká (velikost dat u požadavku GET je omezena na 512 bajtů), nebo pokud není vhodné zobrazit přenášená data v URL
DELETE	Smaže objekt ze serveru. Jsou nutná oprávnění stejně jako u metody PUT

Tabulka 6.2 - Vlastnosti http metod

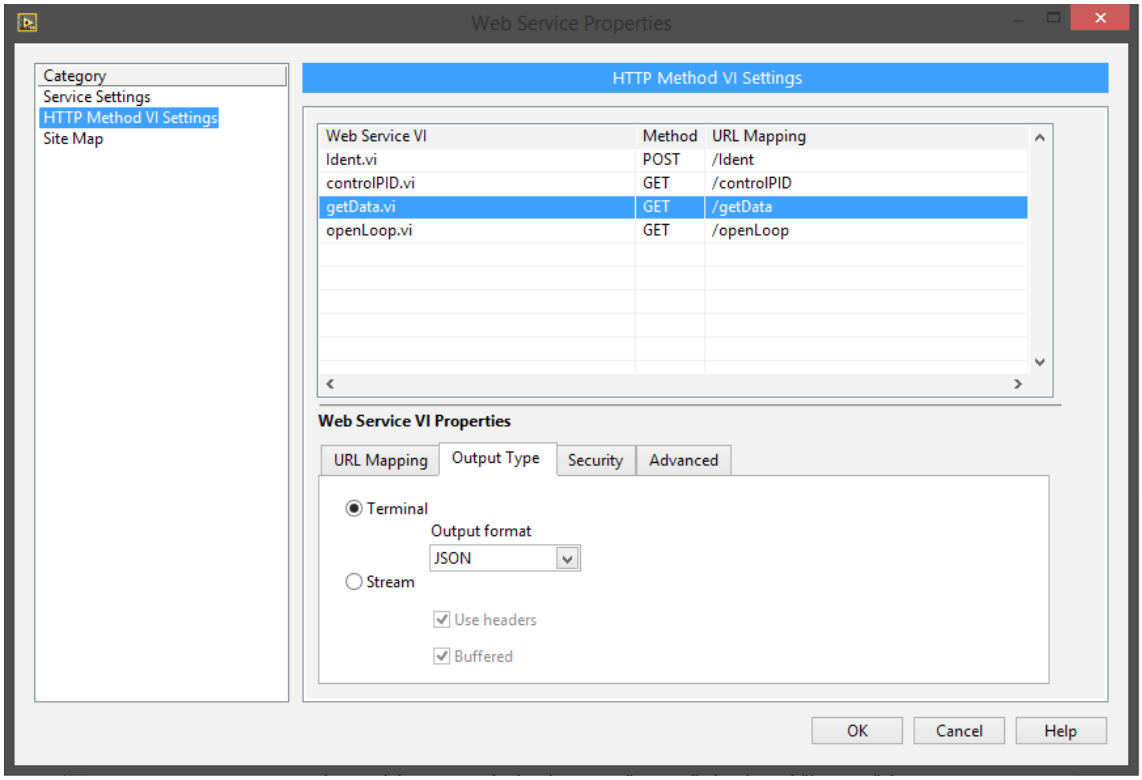
Jelikož metoda getData.VI nemá žádné vstupní parametry, přistupuje se k ní přes adresu 147.230.77.36:8001/HMIweb/getData. Pokud web server přijme požadavek, proběhne vnitřek VI a zpět se odešlou data, která jsou připojena na vnější konektory VI v tomto případě dvě pole typu double. Data jsou posílána ve formátu JSON. Formát výstupních data se možné nastavit v možnostech webové služby. K těm se dostane pravým kliknutím na webovou službu a následně kliknutím na Properties. V záložce HTTP Method VI Settings je možné vybírat mezi jednotlivými dostupnými metodami. Zde v záložce URL Mapping se nastavuje typ metody (v tomto případě GET). Je tu vidět i adresa metody. V záložce Output Type se vybírá typ odchozích dat (obrázek 6.4).



Obrázek 6.3 Metoda getData.VI

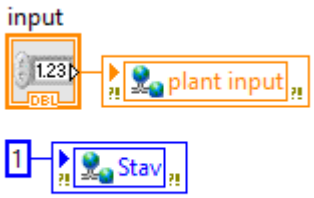
6.1.2 Metoda openLoop.VI

Tato metoda (obrázek 6.5) se používá pro předávání řídících při měření v otevřené smyčce. Metoda je opět typu GET, ale je odlišná od té předchozí. Nevrací žádná data, pouze je předává do řídící smyčky na cRIO. Přistupuje se k ní na adrese 147.230.77.36:8001/HMI/openLoop/?input={value}. Část adresy, která je za posledním lomítkem, je předá-



Obrázek 6.4 Nastavení vlastností http metody

vána proměnná, která byla přečtena na webové stránce a zadal ji uživatel. Toho bylo dosaženo připojením kontrolního prvku na vnější konektor VI a LabView samo interpretuje tuto změnu jako proměnnou v adrese. Hodnota proměnné input reprezentuje vstupní napětí na motoru a je předávána proměnnou plant input do časově kritické smyčky programu na cRIO.



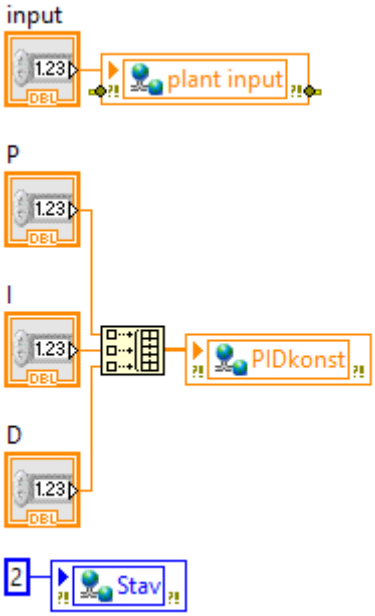
Obrázek 6.5 Metoda openLoop.VI

Důležitou součástí metody je nastavení proměnné stav na hodnotu 1. To zajišťuje, že měření bude probíhat v módu otevřené smyčky. Program takto zůstane nastaven do té doby, než bude stav změněn jiným požadavkem, nebo než program automaticky přejde do stavu klidu po dlouhé době nečinnosti.

6.1.3 Metoda controlPID.VI

Metoda controlPID.VI (obrázek 6.6) se od předchozí metody liší pouze v počtu vstupů a nastavení stavu. Opět se jedná o metodu typu GET, ale nyní jsou předávány 4 vstupní parametry. Parametry P, I a D představují parametry PID regulátoru (P – proporcionální zesílení, I – integrační časová konstanta, D – derivační časová konstanta). Poslední parametr input představuje referenční hodnotu, na kterou PID regulátor reaguje. Všechny tyto proměnné jsou opět připojený na vnější konektor VI a tomu odpovídá adresa 147.230.77.36:8001/HMI/controlPID/?D={value}&I={value}&P={value}&input={value}.

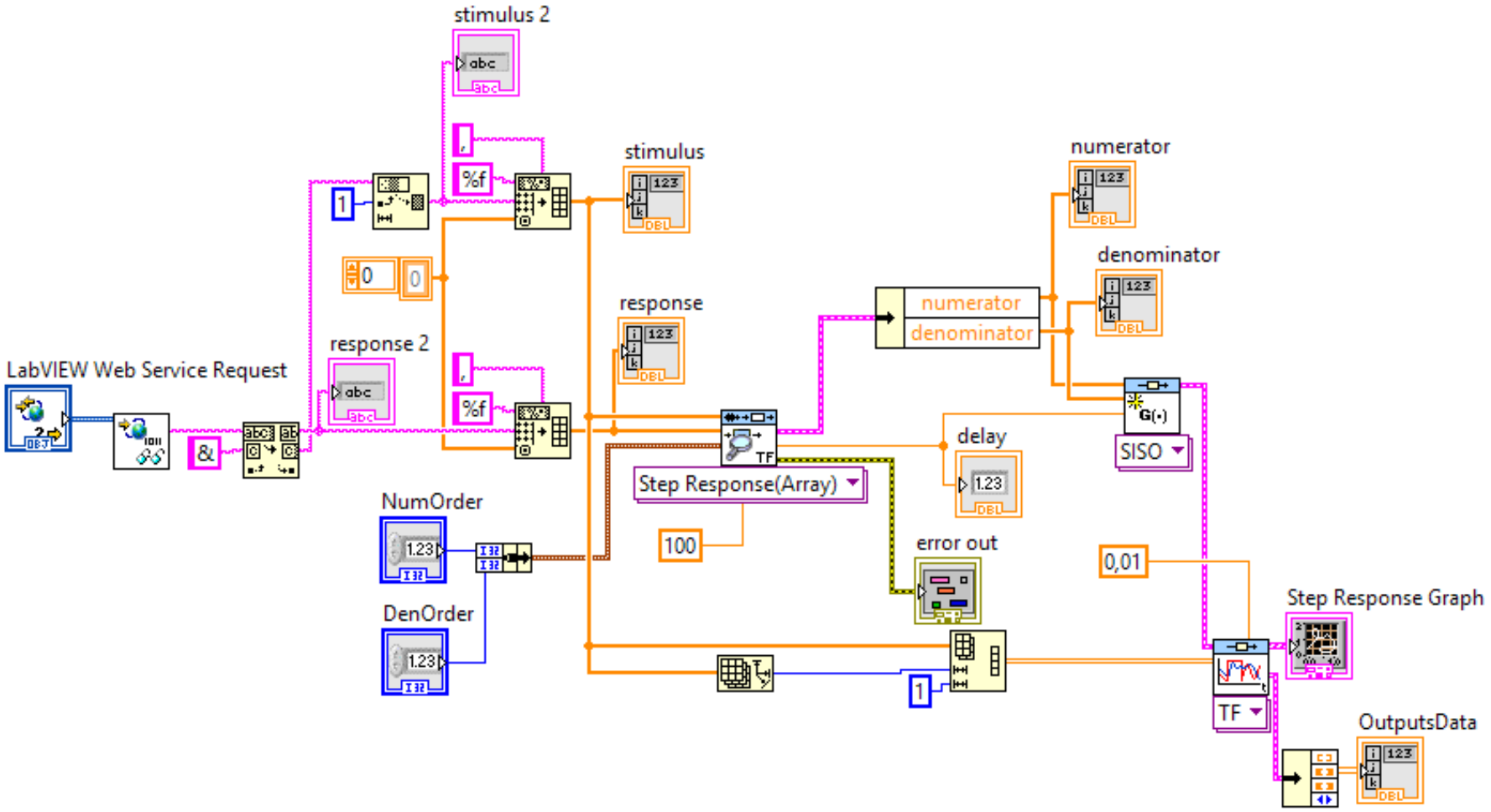
Proměnná stav je nastavena na hodnotu 2, čemuž odpovídá mód zavřené smyčky. Měření tedy probíhá ve zpětnovazebním regulačním obvodu s regulátorem typu PID. Tento stav trvá do té doby, než je přepnut další požadavkem, nebo než program automaticky přejde do stavu klidu po dlouhé době nečinnosti.



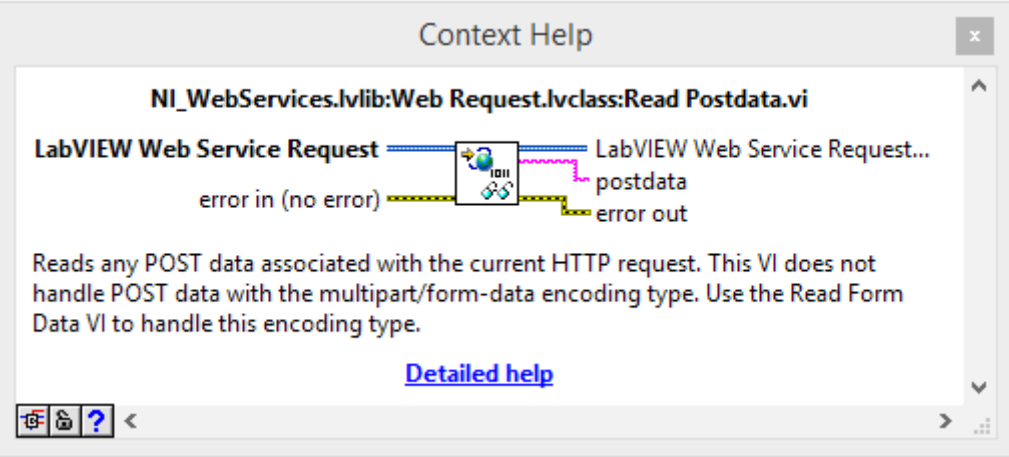
Obrázek 6.6 Metoda controlPID.VI

6.1.4 Metoda Ident.VI

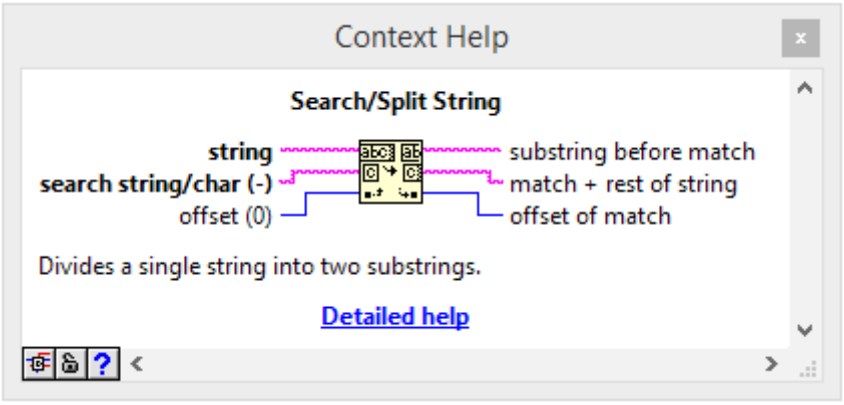
Poslední vytvořenou metodou je Ident.VI (obrázek 6.7). Jelikož je tato metoda využívána pro identifikaci modelu, je výrazně odlišná od všech ostatních. Prvním velkým rozdílem je, že je typu POST. To je z důvodu, že z webové stránky jsou přenášena data ve větším objemu. Data obsahují část naměřených dat, která jsou vhodná pro identifikaci. Obsahují desítky vzorků, a proto je velikost větší než 512 bajtů (omezení velikosti požadavku GET).



Obrázek 6.7 Metoda Ident.VI



Obrázek 6.8 Funkce Read PostData



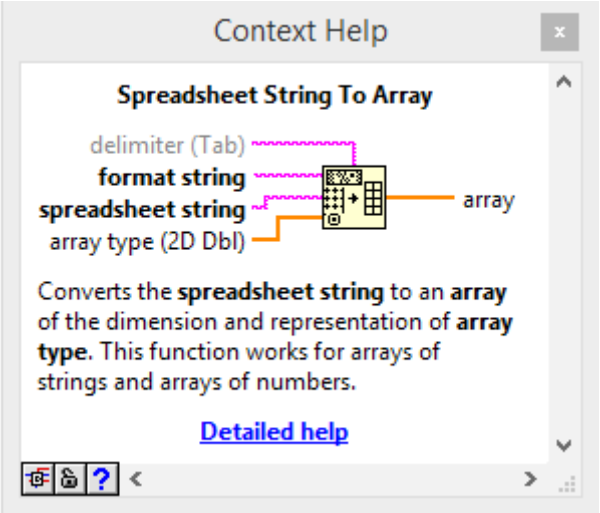
Obrázek 6.9 Funkce Search/Split string

Jelikož se jedná o metodu POST, není možné přečíst obdržená data tak jednoduše jako v případě metody GET. Nejsou totiž obsažena v adrese, ale ve vnitřní proměnné požadavku. Aby se mohla využít veškerá data, která přišla s požadavkem, je nutné využít blok LabVIEW Web Service Request [33]. V tomto objektu je obsažený kompletní http požadavek (hlavička, tělo, proměnné, atd.). Následně je použita funkce z palety Connectivity >> Web services Read Postdata (obrázek 6.8), která přečte data v požadavku a předá je dál jako proměnnou typu řetězec (string). Přečtená data obsahují budící signál, který vstupoval do soustavy a jeho odezvu. Tyto dvě části je nyní třeba rozdělit. Naštěstí jsou oddělena znakem &, a proto je možné použít funkci Search/Split string (obrázek 6.9). Nyní jsou data rozdělena do dvou řetězců, které je dále potřeba přeměnit na pole hodnot, aby vyhovovala podmínkám identifikace. Jednotlivé hodnoty v řetězci jsou odděleny čárkou, a proto se využije funkce Spreadsheet string to array (obrázek 6.10). Tato funkce na základně oddělovače (delimiter) rozdělí řetězec a vytvoří z něj pole. Díky vstupnímu parametru formát řetězce (format string) umí funkce správně reprezentovat číslo typu double.

Vedle samotných dat jsou k identifikaci potřeba ještě další vstupní parametry. Parametry pro určení velikosti řádu čitatele a jmenovatele přenosové funkce jsou předávány v adrese stejně jako u metody GET. Kontrolní prvky NumOrder a DenOrder jsou připojeny na vnější konektor VI. Tomu odpovídá adresa http://147.230.77.36:8001/HMIweb/

Ident/?DenOrder={value}&NumOrder={value}.

Samotnou identifikaci provádí funkce Estimate Transfer Function Model, která je dostupná na paletě pod záložkou Control & Simulation >> Systém identification >> Parametric. Výstupem funkce jsou hodnoty čitatele a jmenovatele přenosové funkce (numerator, denominator) a hodnota dopravního zpoždění (delay). Tyto hodnoty jsou posílány zpět, aby byly zobrazeny uživateli, a proto indikátory numerator, denominator a delay jsou připojeny na vnější konektor VI.



Obrázek 6.10 Funkce Spreadsheet string ti array

Při zobrazování výsledků je vhodné porovnat měřená data se simulovanou odezvou vypočítaného modelu. Z toho důvodu se používají funkce Contruct Transfer Function Model a Linear Simulation. První funkce tvoří hodnot numerator, denominator a delay vhodný formát přenosové funkce, tak aby se mohl použít ve funkci Linear Simulation. Tato funkce vypočítává odezvu systému, který je popsaný přenosovou funkcí, na vstupní data (Inputs). Výsledek je opět připojený na vnější konektor VI a odesílán zpět jako odpověď na požadavek.

6.2 AJAX

Webová služba i s příslušnými http metodami jsou naprogramované a zbývá pouze skript, který by požadavky podával. O to se stará webový nástroj AJAX. Aby bylo možné ho používat, musí v HTML kódu být umístěný příkaz, který nahraje příslušnou knihovnu:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/
jquery.min.js"></script>
```

AJAX se používá spolu s jquery, což je zjednodušený javascript. Základní tvar příkazu vypadá takto:

```
$.ajax({
  type: 'POST',
  url: 'Ident/?DenOrder=' + y + '&NumOrder=' + x,
  data: postData,
  success: function(data){
      document.getElementById("alert").innerHTML = JSON.
stringify(data)},
  error: function(jqXHR, textStatus, errorThrown){}
});
```

U parametru type se nastavuje typ dotazu. Parametr url představuje kompletní adresu http metody. Lze používat absolutní i relativní adresu. Při vývoji se lépe osvědčila adresa relativní. Adresa je normální řetězec, a proto je možné ji jednoduše skládat z více částí. Na příkladu je vidět dvě číselné proměnné (x a y), které jsou do adresy přidávány. Parametr data jsou vnitřní data požadavku. Tento atribut je relevantní pouze u dotazu POST. Jsou to ta data, ke kterým se v LabView musí přistupovat přes objekt požadavku a funkci Read Postdata. U požadavku typu GET tento parametr nemá význam, protože veškerá data se předávají v adrese. V tuto chvíli je požadavek odeslán a program čeká na odpověď. Zde může nastat více situací. V první situaci vše proběhlo v pořádku, data byla obdržena a je nyní možné je zpracovat (v příkladu zobrazení syrových dat ve stránce). Druhá situace nastává, když odpověď od serveru přijde, ale s chybným hlášením a třetí když odpověď vůbec nepřijde. Důsledek je naprosto stejný. Je potřeba se vypořádat s chybovým hlášením, případně se musí informovat uživatel, atd.

V celém skriptu je několik funkcí, které využívají AJAX a volají v závislosti na stránce, která je zrovna otevřená. Při identifikaci se používá jiný dotaz než při PID regulaci. Jedna z funkcí, která se využívá na více místech je getData().

```
function getData(){
  $.ajax({
    type: 'GET',
    url: 'getData',
    success: function(data){
        drawChart(chart_data,data.ouput, data.vstupy);
    },
    error: function(jqXHR, textStatus, errorThrown){
        clearInterval(bezi)}
  });
}
```

Tato funkce posílá AJAX dotaz na http metodu getData.VI (GET). Při úspěšném provedení se přijatá data zobrazí v grafu. Při neúspěchu se vypne opakované volání (vysvětleno níže).

Aby se data v grafu zobrazovala průběžně, využívá se funkce:

```
setInterval(getData, 500);
```

Tento příkaz spouští funkci getData každých 500 ms. Tím je dosažena aktualizace grafu přibližně každou půl sekundu. Opakované volání se spouští tlačítkem Start a vypíná tlačítkem Stop, pokud nenastane výjimečná situace (AJAX vrátí chybu), kdy se opakované volání vypne samo (funkce clearInterval()).

Dále jsou ve skriptu dvě podobné funkce, které posílají data zadaná uživatelem do cRIO (vstupy a paremetry PID). Jsou to funkce sendInputOL() a sendInputPID(). V následujícím kódu je zapsána funkce sendInputOL().

```
function sendInputOL(){
  var x = document.getElementById("vstup").value;
  $.ajax({
    type: 'GET',
    url: 'openLoop/?input='+x,

    success: function(data){
    },
    error: function(jqXHR, textStatus, errorThrown){
        clearInterval(bezi)
    }
  });
}
```

Obě dvě funkce se navzájem liší počtem čtených proměnných ze vstupních polí a adresou, na kterou se požadavek zasílá. Funkce sendInputOL() čte pouze jedno vstupní pole a zasílá jeho hodnotu na adresu openLoop/?input={value}. Funkce sendInputPID() čte 4 vstupní pole a zasílá jejich hodnotu na adresu controlPID/?D={value}&I={value} &P={value}&input={value}.

Poslední funkcí využívají AJAX, je typu POST a zasílá data k identifikaci soustavy. Funkce se jmenuje sendToIdent() a co se týče AJAX požadavku, liší od ostatních typem POST, paramentrem data a samozřejmě jinou adresou (Ident/?DenOrder={value}&Num Order={value}). Další přidaný kód v téro funkci se týká přípravy data do správného tvaru pro odeslání.

```
var response = [];
var stimulus = [];
$.each(parsedData.rows, function(index,element){
    response.push(element.c[1].v);
    stimulus.push(element.c[2].v);
});
response = response.join();
stimulus = stimulus.join();
var postData = response + "&" + stimulus;
```

V kódu se bere datová tabulka grafu a rozdělují se dvou polí response a stimulus. Tyto dvě pole se následně přemění na řetězec funkcí join(). Jednotlivé hodnoty pole jsou v řetězci oddělené čárkou. Poslední úkonem je spojení těchto řetězců oddělovacím znakem & do jednoho velkého řetězce. Ten je pak předaný do vnitřní proměnné požadavku a odeslán na identifikaci. Po přijetí odpovědi se zobrazí přenos a simulovaná data v grafu.

6.3 Podpůrné funkce

Celý skript nepracuje pouze funkce pro posílání AJAX požadavků, ale využívá spoustu dalších funkcí. Které provádějí důležité změny. Důležité je například načítání dat ze souboru a ukládání. Na stránce zpracování je jedna možnost použít zrovna naměřená data a také načíst již uložená data a s nimi poté dále pracovat.

Pro načítání souboru jsou důležité následující dva řádky HTML kódu:

```
<p><input type="file" id="readFile" value=0></p>
<p><button id="load" type="button"
onclick="readSingleFile()">Načíst</button></p>
```

První představuje vstupní pole. Díky typu file, je pro uživatele jednoduché prohledat místní disk počítače a najít potřebný soubor k nahrání. Samotné nahrání a zobrazení v grafu se spustí stiskem tlačítka Načíst. Funkce zodpovědná za nahrání se jmenuje readSingleFile(). Její struktura vypadá takto:

```
function readSingleFile() {
  var file = document.getElementById('readFile').files[0];
  if (!file) {
    return;
  }
  var reader = new FileReader();
  reader.onload = function(e) {
    var loadedData = e.target.result;
    loadChart(JSON.parse(loadedData));
  };
  reader.readAsText(file);
}
```

Nejdříve se načítá soubor ze vstupního pole do proměnné file. Následně se testuje, zda tato proměnná existuje (kontrola proti fatální chybě). Po vytvoření FileReader(), což instance třídy pro čtení souborů, se definuje, co se dělá po přečtení (předání nahraných dat do tabulky a vykreslení grafu). Posledním příkazem je samotné přečtení textu ze souboru. Ve funkci není vůbec řešeno, zda jsou data ve správném tvaru a proto je naprosto v rukou uživatele, v jakém tvaru data nahraje a jestli se správně zobrazí v grafu nebo ne.

Funkce Save() pro ukládání dat se spouští tlačítkem Uložit data. Ve funkci probíhá předání dat z tabulky grafu do nové proměnné data a jsou přeměněna na typ JSON. Poté se tvoří adresa, která obsahuje veškerá data v textové podobě, otevře se nové okno prohlížeče a data se v něm zobrazí. Nyní se ukládá kliknutím pravým tlačítkem myši na výběrem možnosti „Uložit jako…“. V ukládacím dialogovém okně je možné zvolit název souboru a místo uložení. Nastavení přípony souboru je doporučeno na *.txt nebo *.json. Tento přístup byl zvolen kvůli bezpečnostnímu omezení tvořit soubory na serveru z odlišné webové adresy.

Funkce rezat() se používá v části zpracování. Aby mohla být data odeslána na identifikaci, musí mít správný tvar. To znamená jeden skok vstupní veličiny, před změnou musí být několik desítek vzorků a vstup i výstup musí být normován na nulu. Uživatel si data může upravit sám v nějaké jiném nástroji, nebo může využít ořezové funkce. Vstupem do funkce jsou dvě vstupní pole. V první poli se nastavuje počet vzorků, které chce uživatel oříznout od začátku. Ve druhém potom celkovou délku ořezaných dat. Samotný ořez probíhá na datech z tabulky grafu za podpory funkce splice(od, delka), kde atribut od představuje index elementu pole, od kterého proběhne ořez, a atribut delka představuje počet ořezaných vzorků. Cyklus $.each() obstarává normování všech veličin na nulu. Na konci je umístěna funkce na vykreslení ořezaných dat do grafu.

Poslední podpůrnou funkce je pridatCollumn(). Tato funkce se používá také v záložce zpracování při obdržení dat ze samotné identifikace. Cílem je zobrazit v grafu další signál. Při měření se v grafu zobrazuje pouze vstupní a výstupní signál. Po identifikaci je příhodné zobrazit i simulovaná data. Nový signál se do grafu přidává příkazem

chart_data.addColumn(‚number‘, ‚Model‘);

Následně se použije cyklus $.each(), aby každý řádek nového sloupce naplnil hodnotami. Poslední příkaz je vykreslení grafu.

7 Závěr

Bylo představené zařízení CompactRIO od firmy National Instrument. Potvrdilo se, že je to velice vhodné zařízení pro řízení a sběr dat z reálných systémů. Díky real-time procesoru poskytuje velký výpočetní výkon a díky FPGA poskytuje rychlý přístup, ke vstupům a výstupům na přídavných C serices modulech. Pro připojení reálného sytému jsou moduly NI 9381 a NI 9263 vysoce dostačující z hlediska přesnosti a počtu vstupů a výstupů.

Řízená soustava, která je ovládána pomocí cRIO je Wattův regulátor. Z matematického popisu se předpokládalo, že se jedná o nelineární systém, což potvrdilo měření i jednoduchá identifikace. Statická charakteristika vyšla nelineární a po zvolení pracovního bodu [3,5 2,65] a oblasti na vstupu v rozmezí od 3V do 4V bylo zjištěno, že v této relativně lineární části dochází ke značným změnám zesílení systému v rozmezí od 1,1 do 1,5. S takovou nelinearitou je potřeba počítat a je nutné zvolit vhodnou strukturu řízení.

Bylo představeno několik možností vzdáleného připojení ke CompactRIO. Přístup pomocí remote front panelu ve webovém prohlížeči se zdá jako nejjednodušší a nejrychlejší možnost. Poskytuje také výborné grafické uživatelské rozhraní, protože se zobrazuje celý front panel. Z hlediska výkonu a přenosové rychlosti, ale není na komplexní aplikaci dostačující.

Metoda, která využívá samostatnou aplikaci LabView, je také velice zajímavá. Veškeré programování probíhá přímo v LabView. Jediná věc, která se musí řešit navíc, je komunikace mezi programem na cRIO a na uživatelském počítači. Když se využijí sdílené proměnné a network stream, komunikace je rychlá a efektivní. Nevýhodou je nutná instalace LabView RTE a často i samotné aplikace.

Metoda vytvořená pomocí TCP/IP protokolu je náročnější na realizaci, protože je nutné naprogramovat vlastní aplikaci (např. v jazyku java), která by obstarávala TCP/IP komunikaci, zobrazování a ukládání dat. Je to ale flexibilnější metoda, protože se mohou realizovat téměř jakékoliv požadavky, které budou na aplikaci kladeny.

Vzdálené připojení bylo nakonec realizováno webovými službami a webovou aplikací. Na cRIO byl vytvořený program, který umožňuje měření reakce výstupu na vstup a regulaci pochodu s využití PID regulátoru z LabView. Byla vytvořena webová služba, která obsahuje příslušné http metody, které umožňují čtení naměřených dat a předávání řídících parametrů. Byla webová stránka, která umožňuje komunikaci s webovými službami na cRIO a zobrazovaní přenesených dat. K vytvoření grafického vzhledu byl využit Bootstrap, který zjednodušuje programování a poskytuje základní komponenty, které většina stránek potřebuje. Pro zobrazení přenesených dat bylo využito komponenty Google charts, která umožňuje tvorbu interaktivních grafů a jednoduché zobrazování dat v jazyku javascript. Jako hlavní komunikační spoj mezi webovými službami na cRIO a webovou

stránkou byla využita komponenta AJAX. Ta zasílá asynchronní požadavky na http metody na cRIO. Ty obratem vrací žádané data. Tato metoda byla zvolena kvůli zásadní vlastnosti. Na rozdíl od všech předchozích možností tvoří tenkého klienta. Na vytvořené webové stránce jsou čtyři záložky. První dvě umožňují měření na soustavě. Jedna záložka je v módu otevřené smyčky a druhá v módu regulace PID regulátorem. Třetí záložka je určená k úpravě naměřených dat a identifikaci soustavy. V rámci práce byl vytvořený návod, který popisuje grafické rozhraní webové stránky a jednotlivé kroky při identifikaci systému. Tento návod je umístěn ve čtvrté záložce na webové stránce.

Takto vytvořený systém je lehce rozšiřitelný, protože do programu na cRIO stačí přidat další stav a k němu naprogramovat příslušnou http metodu a požadavek. Do budoucna lze tedy systém rozšířit o další možnosti řízení, např.: stavová regulace, prediktivní regulátor, řízení s vnitřním modelem.

Příloha A

Obsah přiloženého CD

2015_vinkler_DP.pdf	Diplomová práce v pdf formátu
LV_program.rar	LabView projekt se zdrojovými kódy programu na cRIO
LV_web.rar	Zdrojové kódy webového rozhraní

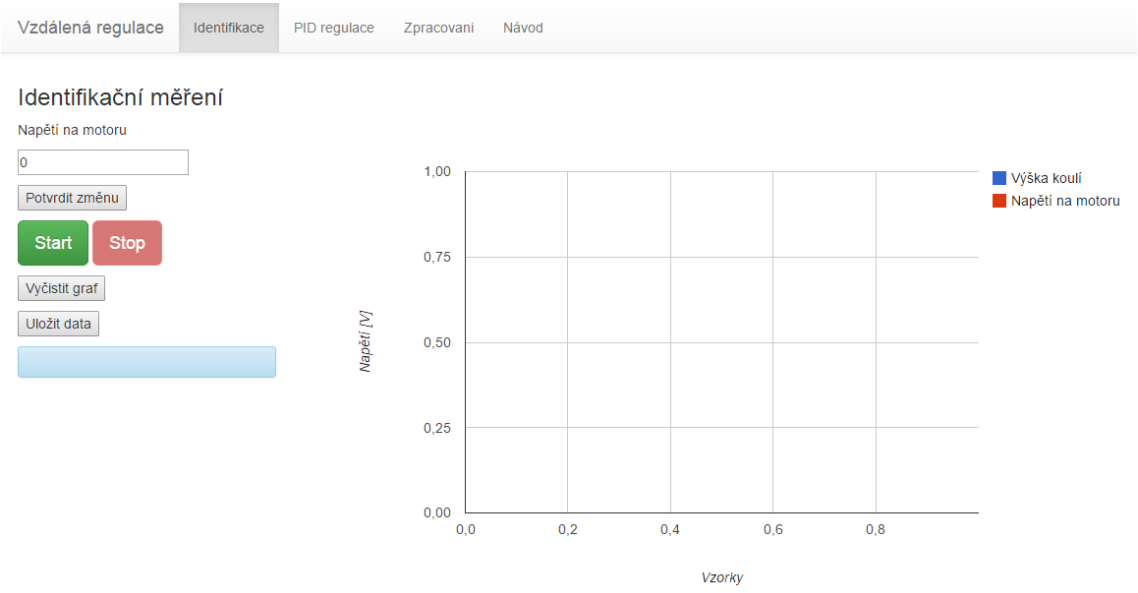
Příloha B

Popis a návod programu

V této příloze je stručně popsáno ovládání programu. Jak se startují a končí měření a jak ukládat naměřená data. Je zde také uvedený návod na identifikační měření, který se musí provádět vždy, když je potřeba řídit soustavu. Zjednodušuje výběr i návrh regulátoru a ulehčuje poté další práci se soustavou. Veškerý následující text je umístěný i na webových stránkách meření pod záložkou Návod.

B.1 Identifikační měření

Na obrázku B.1 je vidět záložka identifikační měření. Napětí na motoru je vstup do soustavy v rozsahu 0-5V. Změna vstupu se musí potvrdit stisknutím tlačítka „Potvrdit změnu“. Tlačítky „Start“ a „Stop“ se spouští a vypíná měření. Před vypnutím měření se doporučuje vrátit vstup do nuly. Naměřená data se zobrazují v grafu. Legenda grafu vysvětluje, že Napětí na motoru je vstup do soustavy a Výška koulí je výstup. Tlačítkem „Vyčistit graf“ se zahodí veškerá naměřená data a měřená dále pokračuje od nuly.



Obrázek B.1 Záložka Identifikační měření

B.2 Ukládání dat

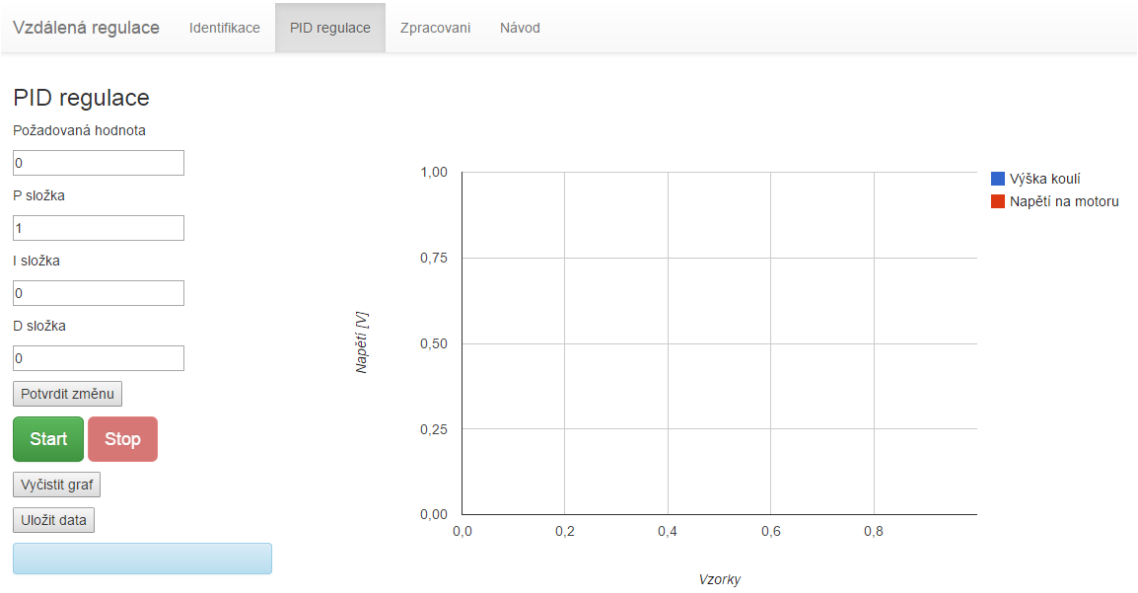
Data, která jsou zobrazená v grafu, je možné uložit do souboru. Po stisknutí tlačítka „Uložit data“ se otevře nová stránka, kde jsou zobrazena data v textové podobě. Nyní se klikne pravým myšítkem a zvolí možnost „Uložit jako...“. Následně se data uloží pod zvoleným názvem. Důležité je zvolit příponu souboru *.txt nebo *.json.

B.3 PID regulace

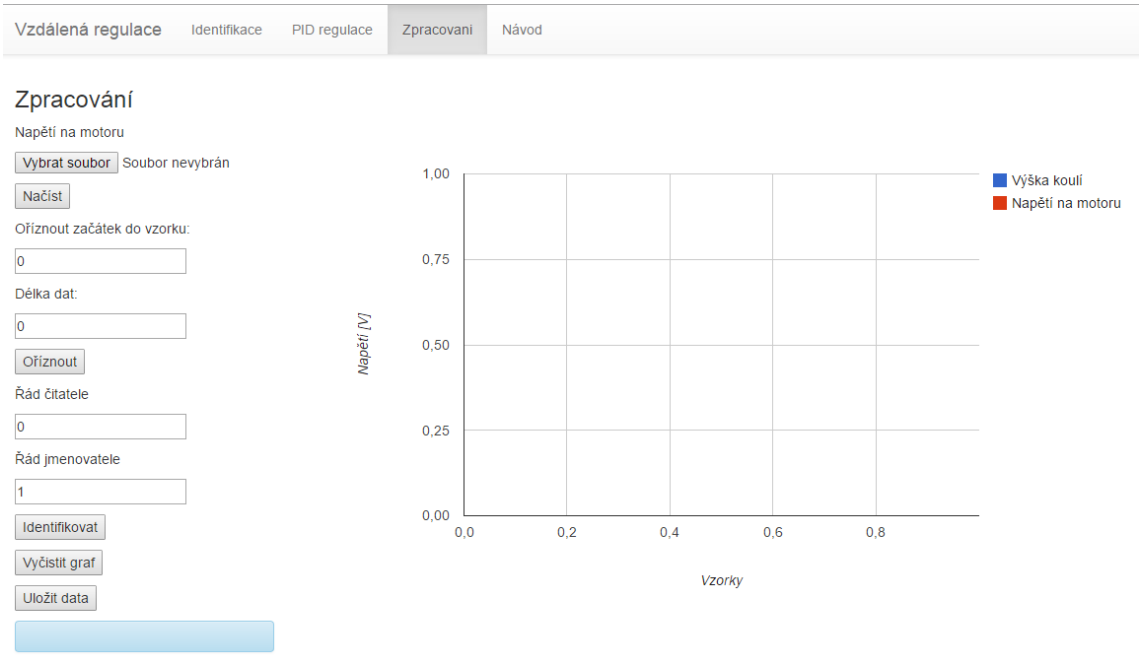
Jedná se o klasickou zpětnovazební regulaci pomocí PID regulátoru (obrázek B.2). Tato záložka se od Identifikačního měření liší pouze v nabízených vstupech. Požadovaná hodnota je vstup do soustavy a zbylé vstupy jsou jednotlivé parametry PID regulátoru. Změna se opět odesílá stisknutím tlačítka „Potvrdit změnu“. Při zadávání Požadované hodnoty je potřeba si uvědomit, jaké maximální hodnoty může soustava dosáhnout při daných vstupech.

B.4 Zpracování

Na této záložce probíhá zpracování dat z identifikačního měření (obrázek B.3). Pokud jsou změřená nějaká data, tak se zde automaticky zobrazí. Zároveň je také možné nahrát data ze souboru. Ta musí být ve speciální tvaru JSON. Ořezání dat se nastavuje parametry „Oříznout začátek do vzorku“ a „Délka dat“. Dále se nastavuje řád čitatele a jmenovatel estimované přenosové funkce. Je nutné dodržovat základní pravidla pro přenosové funkce jako např.: řád jmenovatele musí být stejný nebo větší než čitatele. Po



Obrázek B.2 Záložka PID regulace



Obrázek B.3 Záložka Zpracování

stisknutí tlačítka „Identifikovat“ se zobrazí vypočítaná přenosová funkce. Zbylá tlačítka fungují stejně jako v ostatních záložkách.

B.5 Identifikační měření

Identifikace začíná měřením statické charakteristiky. Toto měření se nejlépe provede sérií skoků od 0 do maximálního napětí 5V. Hodnotu skoku je dobré volit podle doby ustálení, aby měření nebylo moc dlouhé. Z výsledků se určí pracovní bod a pracovní oblast (v ideálním případě lineární).

Pro identifikaci dynamických vlastností soustavy je určeno jiné měření. Nejprve se provede skok do pracovního bodu a po ustálení se provede další skok v pracovní oblasti. Naměřená data je možné uložit do souboru pomocí tlačítka „Uložit“. Zpracování naměřených dat se provádí v záložce „Zpracování“. Naměřená data se vám zde automaticky zobrazí, nebo si můžete nahrát vlastní data ze souboru (musí být ve správném tvaru). Následně se provádí ořezávání dat. Je nutno myslet na to, že do identifikačního výpočtu může vstupovat jen jeden skok. Další důležitá věc, je mít před skokem nějakou dobu ustáleného stavu (doporučení min. 200 vzorků). Při ořezávání vybíráte vzorek, do kterého ořezávat, a délku ořezaných dat. Oříznutím se také data automaticky posunou do nuly. Následně už jen stačí zvolit řád jmenovatele a čitatele výsledné přenosové funkce. Po zmáčknutí tlačítka „Identifikace“ se zobrazí vypočítaný přenos i s případným zpožděním

Příloha C

Spuštění programu

Projekt v LabView se jmenuje RT_html.lvproj. V něm je nejdůležitější soubor DP_zkouska.VI. Je to hlavní VI, ve kterém je realizovaný veškerý program pro cRIO. Spouští se tlačítkem Run. Další důležitá součást projektu je webová služba. Jmenuje se HMIweb. V ní jsou obsaženy všechny potřebné http metody. Spouští se kliknutím pravým tlačítkem myši na název webové služby a výběrem možnosti Start.

Pokud je spuštěná webová služba, tak k webové stránce se připojuje adresou: 147.230.77.36:8001/HMIweb/remoteControl.html.

Číslo portu se může lišit. Port 8001 patří debug módu. Pokud je služba spuštěna samostatně na cRIO, tak číslo portu bude odpovídat nastavení (většinou 8080).

Zdrojový kód stránky je v souboru remoteControl.html. Zdrojový kód příslušného javascript souboru je ve složce js v souboru remoteControl.js. Ostatní soubory ve složce jsou určené pro Bootstrap. Ve složce css jsou kaskádové styly pro Bootstrap. V případě vlastního vzhledu rozhraní je možné sem vložit vlastní soubor s kaskádovými styly.

Reference

[1] CompactRIO Integrated Systems with Real-Time Controller and Reconfigurable Chassis NI cRIO-907x [online]. 2014 [cit. 2015-05-14]. Dostupné z: http://sine.ni.com/ds/app/doc/p/id/ds-354/lang/cs

[2] OPERATING INSTRUCTIONS AND SPECIFICATIONS NI 9381[online]. 2012 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/pdf/manuals/375983a.pdf

[3] OPERATING INSTRUCTIONS AND SPECIFICATIONS NI 9263[online]. 2009 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/pdf/manuals/373781e.pdf

[4] VLACH, Jaroslav, Josef HAVLÍČEK, Martin VLACH. Začínáme s LabView. Ilustrace Viktorie Vlachová. Praha: BEN - technická literatura, 2008, ISBN 978-80-7300-245-9.

[5] MODRLÁK, Osvald, Lukáš HUBKA, Steffen GÄRTNER, Franz WORLITZ. Position Control – Watt Controller. Liberec, 2012.

[6] Remote panels in LabVIEW – Distributed Application Development [online]. 2013 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/tutorial/4791/en/

[7] Remote Front Panels [online]. 2006 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/5154/en/

[8] Developing Remote Front Panel LabView Applications [online]. 2007 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/3277/en/

[9] Configuring Remote Front Panels on a Real-Time Target [online]. 2013 [cit. 2014-05-23]. Dostupné z: http://digital.ni.com/public.nsf/allkb/AB6C6841486E84EA862576C8005A0C26?OpenDocument

[10] Network Variable Technical Overview [online]. 2014 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/5484/en/

[11] Usin the LabView Shared Variable [online]. 2012 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/4679/en/#toc3

[12] Understanding Shared Variable Technology [online]. 2012 [cit. 2015-05-14]. Dostupné z: http://zone.ni.com/reference/en-XX/help/371361J-01/lvconcepts/ni_psp/

[13] Lossless Communivation with NetworkStreams: Components, Architecture and Preformace [online]. 2013 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/12267/en/

[14] Streaming Data and Sending Commands between Applications [online]. 2011 [cit. 2015-05-14]. Dostupné z: http://zone.ni.com/reference/en-XX/help/371361H-01/lvconcepts/networkstreams/

[15] Overview: Web-based Communication with a LabView Application
(Real-Time, Windows) [online]. 2013 [cit. 2015-05-14]. Dostupné z:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvconcepts/webservices/

[16] Developing http Method VIs
(Real-Time, Windows) [online]. 2013 [cit. 2015-05-14]. Dostupné z:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvconcepts/ws_method_vis/

[17] NI LabVIEW Web UI Builder Product Information [online]. 2010 [cit. 2015-05-14].
Dostupné z: http://www.ni.com/uibuilder/

[18] Getting Started with LabView Web UI Builder, Part 1:
Creating and Running a VI [online]. 2010 [cit. 2015-05-14].
Dostupné z: http://zone.ni.com/reference/en-XX/help/373286A-01/uibuilder/wuib_gsg1/

[19] Getting Started with LabView Web UI Builder, Part 2:
Building and Deploying a Thin-Client Application [online]. 2010 [cit. 2015-05-14].
Dostupné z: http://zone.ni.com/reference/en-XX/help/373286A-01/uibuilder/wuib_gsg2/

[20] Getting started – Bootstrap [online]. [cit. 2015-05-14].
Dostupné z: http://getbootstrap.com/getting-started/

[21] Google Charts – Google Developers [online]. 2015 [cit. 2015-05-14]. Dostupné z:
https://google-developers.appspot.com/chart/

[22] Highcharts – Interactive JavaScript charts for your website [online].
2015 [cit. 2015-05-14]. Dostupné z: http://www.highcharts.com/

[23] AJAX Tutorial [online]. 2015 [cit. 2015-05-14].
Dostupné z: http://www.w3schools.com/ajax/default.asp

[24] Same-origin policy [online]. 2015 [cit. 2015-05-14]. Dostupné z: http://en.wikipedia.
org/wiki/Same-origin_policy

[25] TCP and UDP Socket API [online]. 2015 [cit. 2015-05-14]. Dostupné z: http://www.
w3.org/2012/sysapps/tcp-udp-sockets/

[26] Getting Started with CompactRIO and LabVIEW [online]. 2009 [cit. 2014-05-23].
Dostupné z: http://www.ni.com/pdf/manuals/372596b.pdf

[27] NI LabVIEW for CompactRIO Developer's Guide [online]. 2014 [cit. 2014-05-23].
Dostupné z: http://www.ni.com/pdf/products/us/fullcriodevguide.pdf

[28] NI Scan Engine Performance Benchmarks [online]. 2010 [cit. 2015-05-14].
Dostupné z: http://www.ni.com/white-paper/7792/en/

[29] Using NI CompactRIO Scan Mode with NI LabVIEW Software [online].
2013 [cit. 2015-05-14]. Dostupné z: http://www.ni.com/white-paper/7338/en/

[30] Bootstrap 3 Tutorial [online]. 2015 [cit. 2015-05-14].
Dostupné z: http://www.w3schools.com/bootstrap/

[31] Tutorial: Creating and Accessing a LabVIEW Web Service
(Real-Time, Windows) [online]. 2013 [cit. 2015-05-14]. Dostupné z:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/build_web_service/

[32] Components of a Web Service (Real-Time, Windows) [online]. 2013 [cit. 2015-05-14]. Dostupné z: http://zone.ni.com/reference/en-XX/help/371361K-01/lvconcepts/webservices_components/

[33] Using the POST http Method (Real-Time, Windows) [online].
2013 [cit. 2015-05-14]. Dostupné z:
http://zone.ni.com/reference/en-XX/help/371361K-01/lvconcepts/ws_post_method/

[34] Embedded Programming in NI LabVIEW [online]. 2013 [cit. 2014-05-23]. Dostupné z: http://www.ni.com/academic/students/learn-rio/embedded-programming/