ZÁPADOČESKÁ UNIVERZITA V PLZNI
UNIVERSITY OF WEST BOHEMIA IN PILSEN

FAKULTA APLIKOVANÝCH VĚD
KATEDRA KYBERNETIKY
FACULTY OF APPLIED SCIENCES
DEPARTMENT OF CYBERNETICS

REGULACE LABORATORNÍHO MODELU VZDUCHOTECHNICKÉ SOUSTAVY POMOCÍ RASPBERRY PI

CONTROL OF A LABORATORY HVAC SYSTEM MODEL USING THE RASPBERRY PI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB LUŇÁK,
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV SOBOTA, PhD.
SUPERVISOR

PLZEŇ 2015

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub LUŇÁK**
Osobní číslo:	**A12B0624P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Regulace laboratorního modelu vzduchotechnické soustavy pomocí Raspberry Pi**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Seznamte se s minipočítačem Raspberry Pi.
2. Seznamte se s dostupnými SW nástroji pro využití minipočítače Raspberry Pi jako programovatelného PLC (systémy REX, CoDeSys, ProConOS,Resologis IPC, LogiCAD).
3. Alespoň pomocí 2 systémů (REX povinně, druhý systém volitelný) demonstrujte možnosti PID regulace na laboratorním modelu vzduchotechnické soustavy, jestliže jako vstupně-výstupní zařízení poslouží a) průmyslová jednotka Turck BL20, b) Arduino Nano.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30-40 stránek A4**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Bude specifikován vedoucím BP.

Vedoucí bakalářské práce: **Ing. Jaroslav Sobota, PhD.**
Katedra kybernetiky

Datum zadání bakalářské práce: **1. listopadu 2014**

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

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. listopadu 2014

ABSTRAKT

Bakalářská práce se zabývá problematikou automatického řízení modelu vzduchotechnické soustavy za pomocí minipočítače Raspberry Pi. Do této problematiky jednak spadá návrh vhodného regulátoru a jednak i identifikace systému. Obsahem práce je několik identifikačních metod, sloužících k aproximování chování reálného modelu, a i několik metod návrhů systémů zajišťující automatické řízení, tj. regulátorů.

KLÍČOVÁ SLOVA

automatické řízení, identifikace systému, metody identifikace, metody návrhů regulátorů, minipočítač Raspberry Pi

ABSTRACT

This bachelor thesis deals with the automatic control of a model of airconditioning system using minicomputer Raspberry Pi. This issue consists of designing appropriate regulator and also identifying the system. Contents of this thesis are few methods of identification aproximating the behaviour of realistic model as well as few methods of designing the system ensuring automatic control, i.e. regulators.

KEYWORDS

automatic control, system identification, identification methods, methods of designing controllers, minicomputer Raspberry Pi

LUŇÁK, Jakub *Regulace laboratorního modelu vzduchotechnické soustavy pomocí Raspberry Pi*: bakalářská práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky, 2015. 61 s. Vedoucí práce byl Ing. Jaroslav Sobota, PhD.

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Jaroslavu Sobotovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

V Plzni dne ....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Obrovský rozvoj číslicových systémů v posledních několika letech nám umožňuje řešit stále složitější problémy týkající se řízení systémů. V některých případech dokonce mluvíme o tzv. automatickém řízení systému, kdy je snaha o vyloučení lidského faktoru z procesu řízení.

K tomu, aby byl navržen správný řídicí algoritmus je v ideálním případě potřeba znát detailní popis chování systému. To ovšem v praxi u některých případů není možné, a proto je potřeba systém zmapovat na nějakém okolí jeho chování. Toto okolí nelze možno jednoznačně specifikovat, protože záleží za jakým účelem je řídicí algoritmus navrhován, např:

Při návrhu tempomatu (systém udržující stálou rychlost) pro automobil budeme systém zkoumat z hlediska aktuální rychlosti objektu a nebudeme se zajímat o teplotu pneumatik nebo o úhel natočení volantu. Všechny tyto veličiny lze na systému pozorovat, ale pro návrh řídicího algoritmu řízení rychlosti je klíčový pouze jediný aspekt z výše uvedených a to aktuální rychlost vozidla.

Popis systému lze získat nejrůznějšími postupy a metodami od matematicko-fyzikálního modelování, kdy na systém aplikujeme matematicko-fyzikální zákony, kterými se daný systém řídí, a získáme obecně soustavu několika rovnic popisující charakter systému. Tato metoda má velký přínos z hlediska znalosti chování systému, která je přínosná pro predikování onoho systému v budoucnosti, ale je velmi náročná na znalost zákonů fyzikálních, matematických, ekonomických atd. Nejenže je tato metoda v několika případech náročná na hlubší znalost zákonů, ale v mnoha případech i nereálná na realizaci, a proto se touto metodou v práci dále nezabývám. V této práci se především věnuji několika metodám, které bychom mohli všeobecně nazvat metodami experimentálními, kdy systém vybudíme vhodně zvoleným signálem, který přivedeme na vstup systému, a změříme chování na výstupu systému. Takto získaná data různými metodami analyzuji a získám aproximativní model systému, který ovšem není tak obecný jako v předchozí metodě, ale pro mnou zvolený účel je tento model dostačující.

Hlavním cílem práce je navržení vhodného regulátoru pro řízení výchylky klapky modelu vzduchotechnické soustavy, tento model blíže představím v samostatné kapitole. V tomto bodě je opět několik metod zaručující správný výběr struktury regulátoru a výpočet konstant ovlivňující jeho chování. Já si vybral pouze některé z nich,

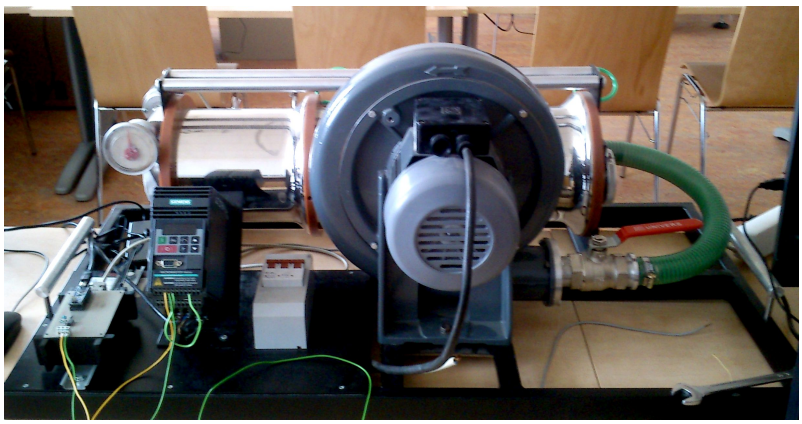
s jejichž pomocí se budu snažit navrhnout několik druhů regulátorů, od pomalého opatrného řízení až po agresivní rychlejší řízení.

Samotný řídicí algoritmus dále implementuji pomocí dvou řídicích systémů a minipočítače Raspberry Pi. Prvním z řídicích systémů je systém od firmy REX Controls s.r.o. s názvem REX (Rapid development, excelent performance) a druhým řídicím systémem je Resologis IPC od firmy Resologis inc.

V závěrečné části práce se zabývám vizualizací modelu vzduchotechnické soustavy za pomocí webové stránky, naprogramované ve standardu HTML5, a technologie WebSocket. Pomocí této webové aplikace je možné systém ovládat, tj. přepínat mezi manuálním a automatickým režimem regulace, nastavovat požadovanou výchylku natočení klapky popřípadě požadované nastavení výkonu motoru vytvářející proud vzduchu, který zajišťuje vychýlení klapky.

1 TEORETICKÁ ČÁST STUDENTSKÉ PRÁCE

1.1 Popis modelu vzduchotechnické soustavy



Obr. 1.1: Reálný model vzduchotechnické soustavy

Reálný model vzduchotechnické soustavy (viz. obr.1.1) se skládá z několika hlavních částí: motoru, potrubí, frekvenčního měniče a klapky usazené na vyústění potrubí vedoucí proud vzduchu.

Motor, tato část modelu má za úkol vytvářet proud vzduchu vychylující klapku z její rovnovážné polohy. Výkon motoru lze cíleně měnit. Jedná se o třífázový asynchronní motor.

Potrubí slouží jako přenosový prvek proudu vzduchu na cílené místo, který je vytvářen motorem. Potrubí je s motorem spojeno vzduchovým ventilem, který lze mechanicky uzavírat a lze tak zavést i chybu do systému (přiškrcením ventilu docílíme menšího průtoku vzduchu do celé soustavy při stejném výkonu motoru). Dále je možno mechanicky měnit charakter průtoku vzduchu skrze potrubí. Při návrhu řídicího algoritmu budu dále pro jednoduchost uvažovat pouze jednu možnost nastavení charakteru průtoku vzduchu a tu během návrhu nebudu měnit.

Klapka je klíčovou částí systému, protože pro návrh řídicího algoritmu budu uvažovat právě její vychýlení z rovnovážné polohy jako výstupní veličinu celé soustavy. Tato výchylka je úměrná průtoku vzduchu procházejícího skrze potrubí, které je generováno motorem. Výchylku lze jednak odečítat ručně z přidaného budíku ukazující aktuální vychýlení klapky v úhlech, ale takto získaná data by byla velmi nepřesná,

a proto je v systému implementované čidlo odesílající údaje o aktuální výchylce klapky. Hodnoty získané z čidla bude zapotřebí analyzovat a přemapovat na nějakou lépe „čitelnější“ veličinu, tj. na úhly. Transformace veličin bude zajištěna až v návrhu samotného řídicího algoritmu v jednotlivých řídicích systémech.

Frekvenční měnič nebo také měnič kmitočtů je zařízení skládající se z několika částí, z usměrňovače, meziobvodu, střídače a řídicího mikropočítače. Kooperací jednotlivých částí získáme celek, který slouží k přeměně elektrického proudu o určité frekvenci na elektrický proud o jiné frekvenci. Tuto vlastnost využijeme při plynulé regulaci motorů.

1.2 Řídicí systémy

1.2.1 Řídicí systém REX

Česká společnost REX Controls s.r.o. se zabývá výzkumem a vývojem pokročilých algoritmů, podpůrných a diagnostických nástrojů i kompletních programových řešení pro přímé řízení strojů a procesů v reálném čase. Firma vyvinula vlastní řídicí systém s názvem „REX“ (Rapid development, excelent performance), který je tzv. otevřený. Lze jej tedy rozšiřovat a přizpůsobovat vlastním potřebám, např: tvorbou nových funkčních bloků.

Tento řídicí systém slouží pro návrh a realizaci komplexních algoritmů automatického řízení. Systém k návrhu algoritmů využívá vestavěnou knihovnu funkčních bloků, pomocí níž lze snadno navrhovat řídicí algoritmy. Knihovna funkčních bloků pokrývá běžné oblasti automatizace, robotiky a regulace.

Mezi základní vlastnosti systému patří podpora:

1. průmyslového standardu OPC
2. programovacího jazyka JAVA v klientské části
3. komunikaci prostřednictvím technologie WebSocket
4. komunikaci prostřednictvím protokolu Modbus

Všechny tyto vlastnosti umožňují vytvářet vizualizační obrazovky (HMI - Human Machine Interface) a virtuální laboratoře.

1.2.2 Řídicí systém Resologis IPC

Společnost Resologis inc. byla založena v roce 2005, Resologis je tým inženýrů a odborných členů, pro který je hlavní odvětví činnosti doprava. Ovšem zabývají se i jinými obory například: ovládací stanoviště, SCADA, distribuované systémy automatizace, Ethernet sítě, vývoj softwaru a simulace systému. Od roku 2012, Resologis vyvíjí produktovou řadu IPC.

IPC platforma je sada obsahující průmyslové automatizační programy, navržena tak, aby byla kompatibilní s více reálnými a virtuálními zařízeními. Tato sada programů je zabalena v software image (softwarový obraz), který je otestován a certifikován pro každé zařízení. Jedna z verzí je určena pro platformu Rasberry Pi, včetně vstupně-výstupních PiFace řadiči. Bezplatná zkušební verze je k dispozici, ovšem s omezením běhu výpočetního jádra po dobu dvou po sobě jdoucích hodin.

I tento řídicí systém opět podporuje standard OPC, pomocí něhož lze vytvářet příznivé grafické uživatelské rozhraní pro obsluhu daného objektu nebo pro vizualizaci běhu objektu. Opět systém podporuje vzájemnou komunikaci prostřednictvím otevřeného protokolu Modbus.

1.3 Metody identifikace systémů

1.3.1 Metoda nejmenších čtverců

Metoda založená na předpokladu, že mezi vstupní posloupností $\{u(kT)\}$ a výstupní posloupností $\{y(kT)\}$ diskrétních hodnot existuje kauzální vztah, který budeme respektovat výběrem vhodné struktury modelu s neznámými parametry. Z této úvahy dále plyne regresní tvar diferenciálních rovnic:

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=1}^{n} b_i u(k-i) + \xi(k) \quad ,$$

kde první dva členy udávají předpokládanou kauzální závislost naměřených dat a ξ pak chybu, která vznikla při měření.

Regresní model lze převést do maticového tvaru, který má pak tvar:

$$y_m = \Phi\Theta + \xi$$

y_m ... vektor měření,Φ ... matice regresorů, Θ ... vektor parametrů, ξ ... vektor chyb

Tvar minimalizačního kritéria:

$$J(\Theta) = \sum_{j=k}^{k+l} \xi^2(j,\Theta) \quad \text{resp. vektorově} \quad J(\Theta) = \xi^T(\Theta)\xi(\Theta)$$

odvozením získám výpočtový tvar pro optimální hodnoty parametrů:

$$\Theta^* = \left(\Theta^T\Theta\right)^{-1}\Theta^T y_m$$

1.3.2 Identifikace pomocí MATLABu

K identifikaci systému, o kterém nemáme žádné bližší informace pouze data, která můžeme na systému naměřit, můžeme také využít aplikaci obsaženou v softwaru MATLAB.

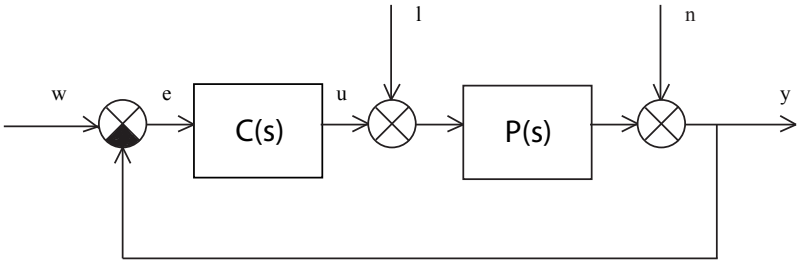
Matrix laboratory (zkráceně MATLAB), je interaktivní programové prostředí a skriptovací programovací jazyk vyvíjený firmou MathWorks. Součástí tohoto prostředí je miniaplikace s názvem **„System Identification Toolbox“**, pomocí které lze jednoduše systém identifikovat. Do prostředí aplikace se načtou naměřená data, nastaví se požadované počáteční podmínky a druh systému spojitý či diskrétní, popřípadě se zvolí perioda vzorkování a další vlastnosti blíže specifikující chování systému, jenž chceme identifikovat.

Výstupem aplikace je popis systému ve tvaru, který si předem specifikujeme (přenosovou funkci, stavový popis systému, ...). Systém tedy umožňuje matematický popis i hodně složitých systémů, jejichž chování není úplně jednoduché popsat.

1.4 Metody návrhů regulátorů

Existuje mnoho metod pro návrhy nejrůznějších regulátorů. V této práci se zabývám pouze jednou metodou návrhu regulátorů a to **metodou robustních regionů**, nebo ji také můžeme nazývat metoda návrhu regulátoru pomocí tvarovacích bodů. Touto metodou lze navrhnout robustní regulátor, tzn. jeden regulátor, který je schopen regulovat několik systémů.

Hlavní myšlenkou této metody je tvarování Nyquistovy křivky otevřené regulační smyčky na základě základních požadavků na bezpečnost v zesílení a bezpečnosti ve fázi.



Obr. 1.2: Regulační smyčka

Uvažujme jednoduchou regulační smyčku, viz obr. č. 1.2, kde $\boldsymbol{w}$ označuje požadovanou hodnotu, $\boldsymbol{e}$ regulační odchylku, $\boldsymbol{u}$ akční zásah a $\boldsymbol{y}$ regulovanou veličinu. Proměnné $\boldsymbol{l}$ a $\boldsymbol{n}$ reprezentují poruchy působící na soustavu.
Pak, pokud budeme uvažovat nulové poruchy působící na systém, přenos otevřené regulační smyčky je dán vztahem

$$L(j\omega) = C(j\omega)P(j\omega) \quad ,$$

kde $s = j\omega$. Kompenzací Nyquistovy křivky právě tohoto přenosu $L(j\omega)$ ovlivníme tvar uzavřené regulační smyčky, jejíž přenos je dán vztahem

$$F(j\omega) = \frac{L(j\omega)}{1 + L(j\omega)} = \frac{C(j\omega)P(j\omega)}{1 + C(j\omega)P(j\omega)}$$

Pokud si dále zvolíme strukturu regulátoru odpovídající PI regulátoru,

$$C(j\omega) = k - j\frac{k_i}{\omega}$$

a přenos systému ve tvaru:

$$P(j\omega) = a(\omega) + jb(\omega)$$

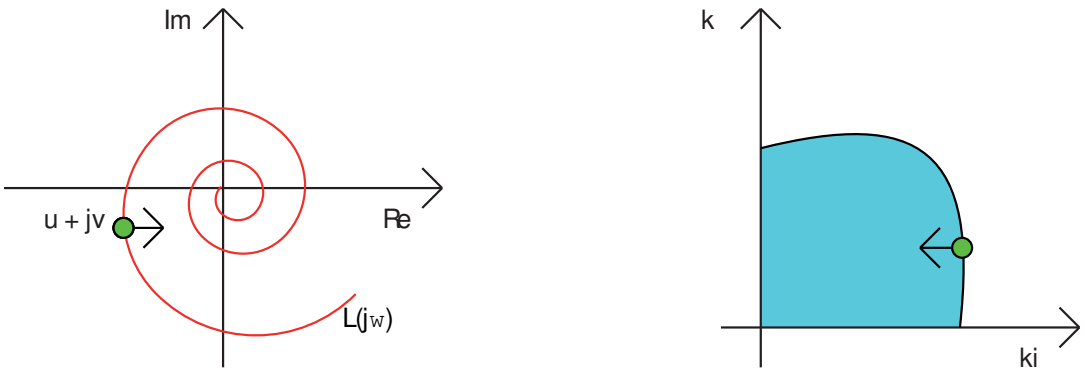
potom tvar otevřené regulační smyčky $L(j\omega)$ je dán vztahem:

$$L(j\omega) = \left(k - j\frac{k_i}{\omega}\right)(a(\omega) + jb(\omega))$$

Tvarovací bod obecně můžeme zapsat $X = u + jv$. Pokud tyto dva vztahy dáme do rovnosti, získáme rovnici jejíž řešením získáme vztahy, které definují parametrickou křivku s parametrem ω v rovině k, k_i parametrů PI regulátoru (připouštíme pouze kladné hodnoty parametrů k a k_i, proto nás zajímají pouze hodnoty v prvním kvadrantu viz. obr. 1.3).

$$L(j\omega) = \left(k - j\frac{k_i}{\omega}\right)(a(\omega) + jb(\omega)) = u + jv$$

$$k_i = \frac{[a(\omega)v - b(\omega)u]\,\omega}{a^2(\omega) + b^2(\omega)} \qquad k = \frac{a(\omega)u + b(\omega)v}{a^2(\omega) + b^2(\omega)}$$



Obr. 1.3: Metoda robustních regionů

1.4.1 Nástroj PIDlab

Účelem webu PIDlab.com je poskytnout virtuální laboratoře (Java aplety) prezentující pokročilé problémy PID regulace. Web přináší nástroje pro interaktivní návrh a ladění PID regulátorů na základě nominálního modelu či experimentálních dat, nástroje prezentující pokročilé PID autotunery a nástroje pro simulaci komplexních PID řídicích struktur používaných v průmyslové praxi. Simulátory jsou celosvětově používány pro výuku i pro komerční a vývojové účely.

Právě pomocí tohoto nástroje, který je veřejně dostupný, v další části práce navrhuji parametry regulátorů. Základem nástroje je právě metoda „Robustních regionů“. Tuto metodu nástroj rozšiřuje a lze pomocí něj navrhovat regulátory nejrůznějších druhů, já se ovšem zaměřuji na návrh dostatečně robustních PID regulátorů. Více informací o tomto nástroji lze nalézt na stránkách výrobce viz. příloha - odkazy.

1.5 Minipočítač Raspberry Pi

Raspberry Pi je výpočetní modul, jednodeskový počítač velikosti zhruba platební karty (viz. obr. A.1). Je vyvíjen britskou firmou Raspberry Pi Foundantion. Základem desky je výpočetní jádro ARM1176JZF-S s taktem 700 MHz, grafický procesor VideoCore IV a 256 MB nebo 512 MB paměti RWM (RAM). Pro trvalejší uchovávání dat je na kartě obsažen slot pro SD kartu. Další rozhraní, jako např. USB a síťový adaptér s konektorem RJ45, obsaženo na desce se liší verzí minipočítače.

Samotným výrobcem je deska dodávána s operačními systémy ARMové verze linuxových distribucí Debian a Arch, popřípadě Rasdroid na bázi operačního systému Android verze 4.0.

Funkce základní desky je možno rozšířit o různé moduly jako například HDMIPi, displej pro Raspberry Pi o rozlišení 1280x800 pixelů při velikosti 9 palců (22.86 cm).

Zařízení lze využít v několika odvětvích od automatizace domácnosti až po ovládání robotů. Raspberry Pi lze také využít jako webový server.

V práci využívám konkrétně model **Raspberry Pi B+** (viz. příloha obrázky: obr. č. A.1). Design je založen na Broadcom BCM2835 SoC, který spojuje ARM1176JZF-S 700 MHz CPU, VideoCore IV GPU, 512 MB paměti RAM. Modul také obsahuje 40pin GPIO, 4 x USB 2.0, 4 pólový stereo výstup a kompozitní video port, HDMI (1080p) 10/100 ethernet port, CSI kamera port pro připojení Raspberry Pi kamery, DSI display port pro připojení Raspberry Pi displeje, micro SD slot pro kartu s OS a pro ukládání dat, micro USB konektor napájení. Návrh neobsahuje vestavěný pevný disk, místo toho spoléhá na microSD kartu pro bootování a ukládání dat. Tato platforma je určena pro operační systém na bázi LINUXu (Raspbian).

1.6 Arduino Nano

Arduino je open-source platforma založená na mikrokontrolérech ATMega od firmy Atmel. Desky arduino obsahují 8-bitové mikrokontrolery, několik I/O pinů, do kterých lze jednoduše připojit další obvody (Shieldy).

Arduino Nano (viz. příloha obrázky: obr. č. A.2) je mikrokontrolérová vývojová deska založená na ATmega328. Je to nejmenší vývojová deska ze všech Arduin. Desku lze napájet 7-12 V stejnosměrným zdrojem napětí. Samotná deska dále pracuje pouze s napětím do 5 V. Deska obsahuje 14 digitálních I/O pinů (z toho může

být 6 použito jako výstupy PWM), 6 analogových vstupů, 16 MHz krystal, připojení pomocí USB, napájecí konektor, ICSP rozhraní a resetovací tlačítko.

Bližší specifikace nalezneme na stránkách výrobce (viz. příloha odkazy).

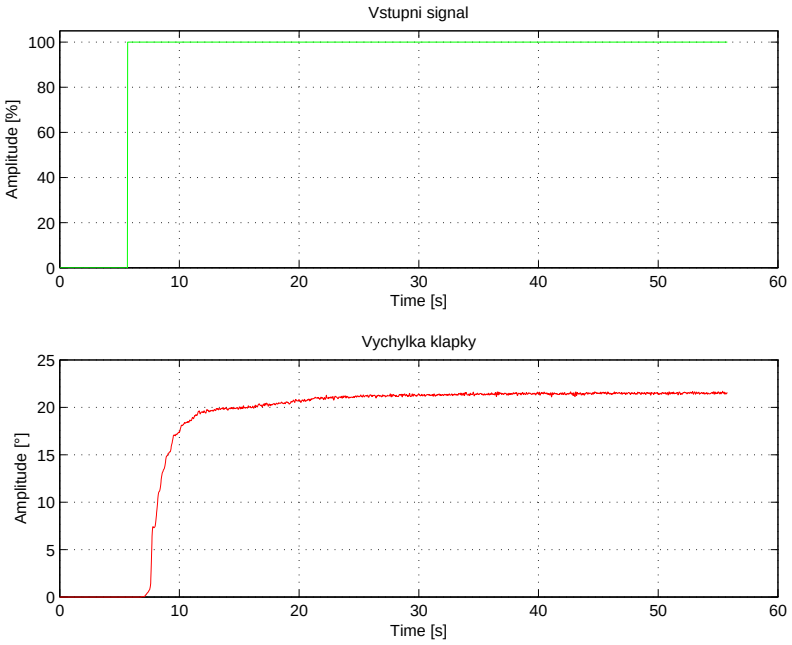
V mém případě arduino využívám jako vstupně-výstupní jednotku komunikující s jádrem řídicího systému REX (RexCore).

Spolupráce řídicího systému REX a Arduino desky je založen na jednoduchém komunikačním protokolu, jehož podřazená část (slave) je implementována v Arduinu, zatímco hlavní část (master) běží v cílovém zařízení REX Controls System (např Raspberry Pi, nebo notebooku nebo desktop PC běží runtime modulu RexCore). Hlavní část je realizována pomocí programovatelného funkčního bloku REXLANG.

2 PRAKTICKÁ ČÁST STUDENTSKÉ PRÁCE

2.1 Analýza chování systému

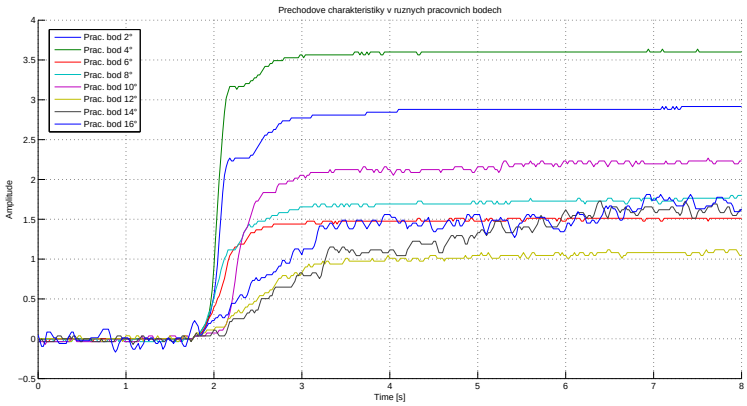
Systém jsem se rozhodl analyzovat v několika pracovních bodech. Nejprve jsem si zjistil do jaké maximální výšky, jakého maximálního úhlu, může klapka dosáhnout. Nastavil jsem nulový výkon motoru a tím jsem zjistil spodní hranici výchylky klapky, kterou jsem si definoval jako nulový úhel. Poté jsem motor pustil na plný výkon a zjistil maximální vychýlení klapky od definovaného počátku. Výsledek experimentu si můžeme prohlédnout na obr. 2.1.



Obr. 2.1: Maximální výchylka klapky

Z experimentu jsem zjistil, že maximální výchylka, v mnou definované soustavě, je přibližně 22°. Takto definované rozpětí 0° až 22° jsem rozdělil do několika pracovních bodů, jedná se o násobky dvou začínající od 2° a končící ve 20°.

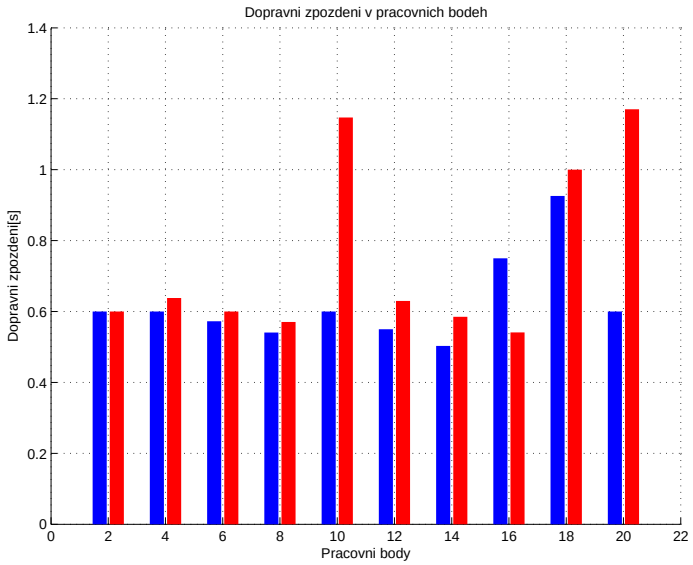
Dále jsem v těchto pracovních bodech provedl experimenty, podle kterých jsem si přiblížil některé vlastnosti systému. V každém pracovním bodě jsem provedl skok o 5% výkonu motoru směrem nahoru (+5% výkonu) i dolů (−5% výkonu). Naměřené přechodové charakteristiky si můžeme prohlédnout na obrázcích č. 2.2 a č. B.1(viz. příloha grafy).



Obr. 2.2: Přechodové děje v jednotlivých pracovních bodech +5%

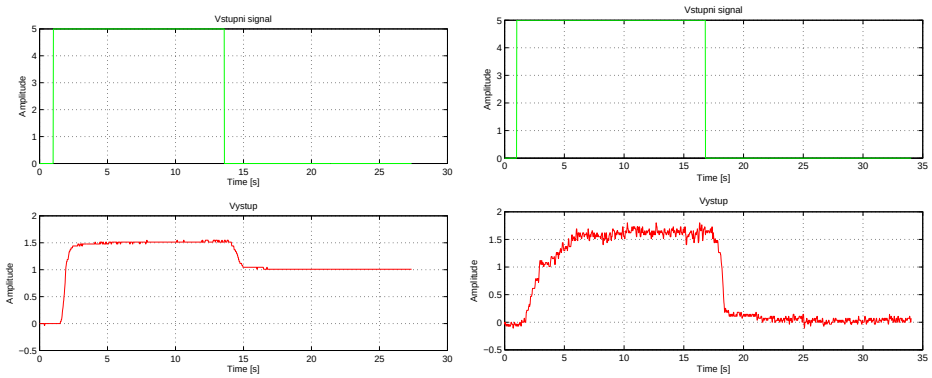
Na obrázku č. 2.2 jsou pro názornost vyobrazeny pouze vybrané přechodové děje v několika pracovních bodech a dále je ignorováno dopravní zpoždění, které je v každém pracovním bodě jiné. Charakteristiky jsou posunuté v časové ose tak, aby děj začínal ze stejného bodu, aby bylo lépe vidět, že systém má v každém pracovním bodě jiné statické zesílení a také jinou dynamiku, která je charakteristická pro každý pracovní bod.

Přehled dopravních zpoždění je znázorněn na obrázku č. 2.3. V grafu jsou znázorněny jednak dopravní zpoždění pro případ, kdy v pracovním bodě zvýšíme výkon o 5% (modrá barva), a jednak i pro případ, kdy naopak v pracovním bodě výkon snížíme o 5% (červená barva).



Obr. 2.3: Dopravní zpoždění v pracovních bodech

Další problém, na který jsem narazil při analýze systému, byl takový, že při návratu klapky do výchozí polohy byla ustálená hodnota jiná než před provedením vychýlení klapky. Tato vlastnost je zapříčiněna zejména třením, které vzniká na hřídelce a jejím uchycení. Proto jsem se rozhodl provést tento experiment, tj. odezvu systému na obdélníkový puls v jednotlivých pracovních bodech. Výsledek experimentu je znázorněn na dvou grafech. První graf demonstruje chování systému pří menším vychýlení a druhý naopak znázorňuje chování systému při vyšším vychýlení klapky.



Obr. 2.4: Odezva na obdélníkový signál

Z obrázku č. 2.4 je patrné, že pro malé výchylky (graf vlevo) se hodnota ustálených stavů při stejném vstupu liší. Kdežto při větších výchylkách (graf vpravo) můžeme konstatovat shodnost ustálených stavů při obdélníkovém vstupu do systému. Neshodnost těchto ustálených stavů při malých výchylkách, jak již bylo řečeno, je zapříčiněno především třením vzniklé v ukotvení hřídelky, na níž je uchycena samotná klapka. Ovšem při větším vychýlení klapky a poté ubrání výkonu motoru se klapka vrátí do výchozí polohy proto, že přitažlivé síly působící na samotnou klapku překonají odpor vzniklý třením. Tento jev při menších výchylkách není možný, neboť přitažlivé síly na malých výchylkách nemají dostatek energie, aby tření překonalo a klapka se tedy ustálí na jiné hodnotě, než byla hodnota výchozí.

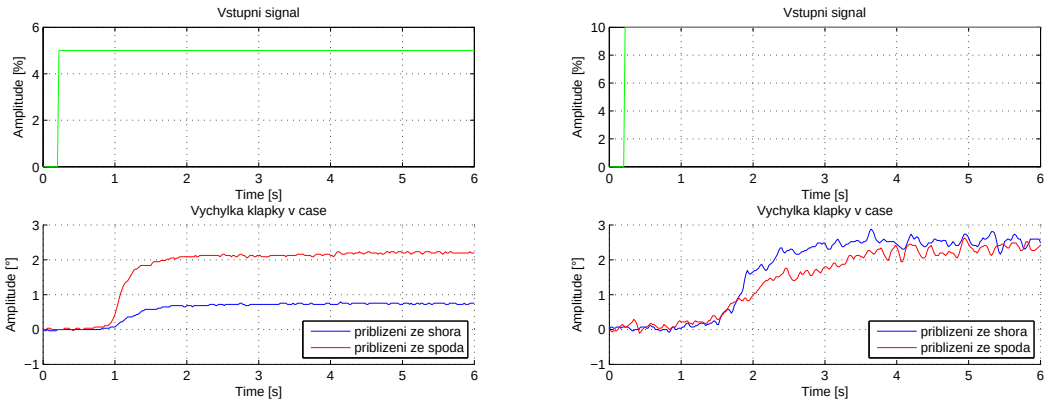
Tento „nedostatek“ by bylo možno kompenzovat zejména lepším uchycením hřídelky, nejlépe takovým, kde nedochází ke tření např: elektromagnetickým uchycením. Nebo by postačilo stávající uchycení promazat a snížit tak tření, které zde vzniká.

Díky tomuto tření vzniká další problém, který bude komplikovat návrh řídícího systému. V některých případech provádění experimentu jsem se setkal s problémem,

kdy jsem si myslel, že výchylka se pomalu ustaluje na své hodnotě a najednou se klapka doslova „utrhne“ a ustálí na hodnotě jiné než se ze začátku jevilo. V přechodové charakteristice tak vznikne nepříjemný skok, který se těžko bude aproximovat. Jev je zapříčiněn opět třením působící na hřídelku klapky. Motor po čase působení na klapku, vlivem proudu vzduchu, vyvine dostatek síly na to, aby tyto třecí síly překonal, což má za následek prudkou změnu polohy klapky.

Další charakteristické chování, které jsem při analýze systému pozoroval, nastává v případě, kdy se blížíme k pracovnímu bodu jednou z vyšších poloh a po druhé z poloh nižších. Pokud v pracovním bodě provedeme tentýž experiment, ale jednou se budeme k pracovnímu bodu blížit z nižších poloh a podruhé z vyšších poloh, výsledek experimentu nebude totožný, ale jeho výsledky se budou lišit.

Výsledek takového experimentu je vyobrazen v následujících grafech, kdy jednou je experiment proveden v pracovním bodě 10° (levý graf) a v druhém je experiment proveden v pracovním bodě 18° (pravý graf):

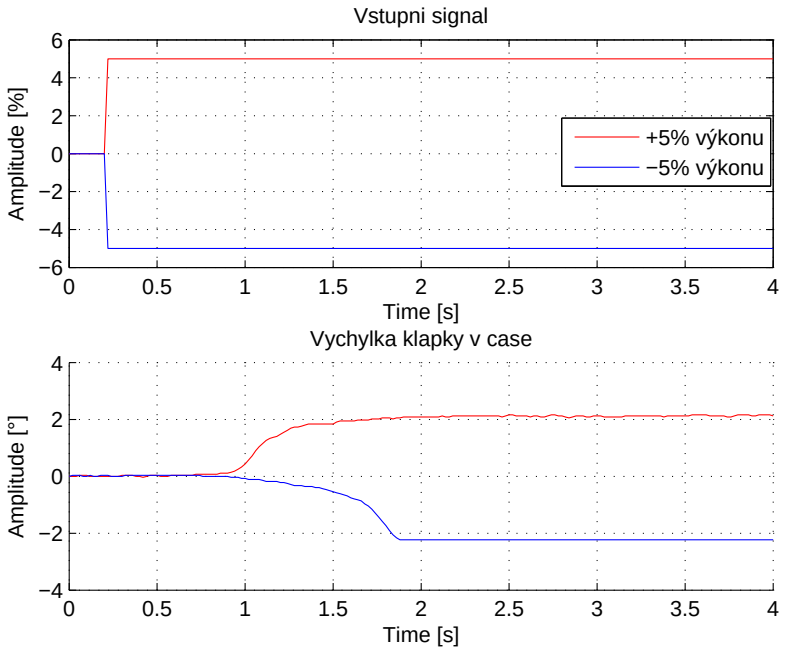


Obr. 2.5: Srovnání přechodových dějů při různém přiblížení k prac. bodu

Z výsledku experimentu lze konstatovat, že systém má odlišné chování při různém přiblížení k pracovnímu bodu při provedení totožného pokusu. Bude tedy zapotřebí rozlišit z jakého směru se k pracovnímu bodu systém blíží a nebo navrhnout dostatečně robustní regulátor tak, aby byl schopen regulovat subsystémy, které se kolem pracovního bodu vyskytují.

Je patrné, že systém na menších výchylkách cca do 10° při odlišném přiblížení se k pracovnímu bodu má jednak jiné statické zesílení a jednak i jinou dynamiku. Pro vyšší výchylku, cca nad 10°, má systém zhruba stejné statické zesílení, ale jinou dynamiku. Dopravní zpoždění se jeví v jednotlivých výsledcích zhruba totožné.

Další co mě při analýze chování systému zajímalo byl průběh sledované veličiny, tedy výchylky klapky v čase, kdy v pracovním bodě provedeme skok o určitou hodnotu směrem k vyšším výchylkám a v druhém případě provedeme v témže pracovním bodě stejný skok, ale v opačném směru. Pokud by systém byl lineární, výsledky jednotlivých experimentů by měli být totožné ovšem překlopené podle časové osy. Tedy v určitém čase t by měl systém nabývat shodných hodnot s opačným znaménkem. Z předešlých experimentů je ovšem patrné, že systém má nelineární charakter a lze tedy předpokládat, že výsledek experimentu se nebude shodovat s předešlou úvahou linearity systému.



Obr. 2.6: Odezva systému na skokový signál v různém směru

Na obr. č. 2.6 je vyobrazen výsledek experimentu, kdy do systému je jednou puštěn skok o velikost +5% výkonu motoru (červené průběhy) a po druhé skok o velikosti -5% výkonu motoru (modré průběhy). Z těchto průběhů je vidět, že systém má opravdu nelineární charakter a průběhy zcela jistě nejsou osově souměrné podle časové osy. Proto opět bude zapotřebí, při navrhování regulace pro tento systém, brát v potaz i tento aspekt, který opět řeším robustní regulací.

V tomto pracovním bodě (10°) má systém alespoň zhruba stejné statické zesílení pro jednotlivé skoky, ale v systému se nalézají i takové odezvy, které nemají shodné ani statické zesílení, ani dynamiku a ani dopravní zpoždění. Příklad takového bodu je pracovní bod 8° (viz. příloha grafy obr. č. B.2).

2.1.1 Shrnutí analýzy chování systému

V každém pracovním bodě můžu provést jednu z kombinací následujících experimentů:

1. Na vstup mohu pustit obdélníkový signál ⇒ dva podsystémy, se kterými musím počítat viz. obr. 2.4 (Odezva na obdélníkový signál), tento experiment mohu provést i v opačném smyslu, než je vyobrazeno na obrázku ⇒ plus další dva podsystémy.

2. K pracovnímu bodu se mohu blížit ze dvou směrů, buďto ze shora nebo ze zdola ⇒ $2 \cdot (2 + 2)$ podsystémů

Celkem tedy v každém pracovním bodě potřebuji aproximovat chování osmi podsystémů, abych mohl provést regulaci alespoň na okolí tohoto pracovního bodu. Pokud jsem si tedy definoval deset pracovních bodů, znamená to aproximovat chování alespoň osmdesáti podsystémů.

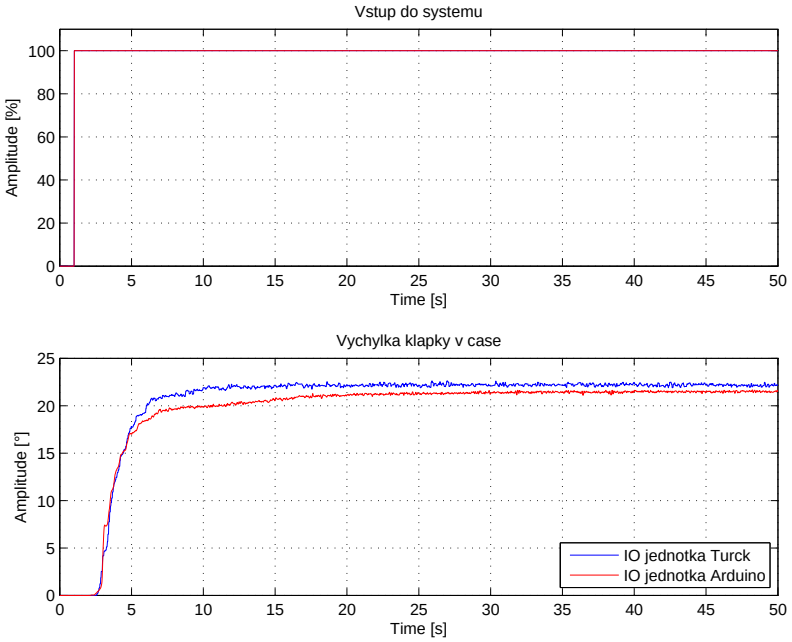
2.2 Vliv vstupně-výstupního zařízení na chování systému

V této části se zabývám porovnáním chování systému, kdy jednou použiji jako vstupně-výstupní zařízení Arduino Nano a v druhém případě pak využiji jako vstupně-výstupní zařízení průmyslovou jednotku Turck BL20.

Cílem této, části je dokázat hypotézu, že vstupně-výstupní zařízení nemá na chování systému žádný vliv a proto lze systém analyzovat s kterýmkoliv zapojením a aplikovat získané poznatky pro oba případy.

2.2.1 Maximální výchylka klapky v čase

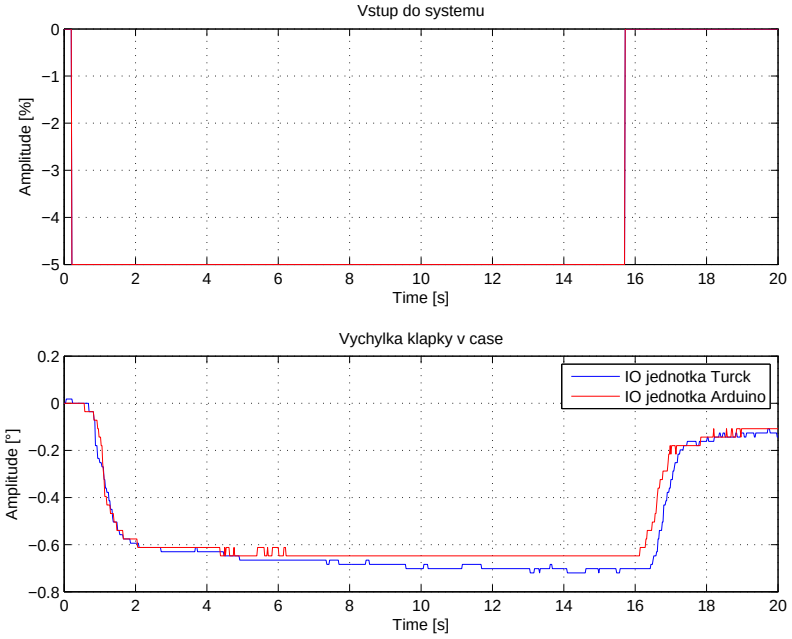
Nejprve jsem otestoval zdali klapka docílí té samé výchylky při maximálním výkonu motoru, tj.maximální výchylku klapky, kterou je možné dosáhnout. Výsledek experimentů a jejich porovnání je vyobrazeno na obr. č. 2.7.



Obr. 2.7: Maximální výchylka srovní IO jednotek

2.2.2 Odezva systému na obdélníkový signál

V tomto experimentu pustím na vstup obdélníkový signál a porovnám výsledky experimentu jednou pro zapojení se vstupně-výstupní jednotkou Arduino a po druhé pak se zapojením IO jednotky Turck.



Obr. 2.8: Odezva systému na obdélníkový signál srovnání IO jednotek

Těmito několika experimenty jsem ukázal, že opravdu platí hypotéza ekvivalentnosti vstupně-výstupních zařízení a úplná neshodnost signálů je spíše charakteristickou vlastností systému, nežli vstupně-výstupního zařízení. Proto lze systém identifikovat, tzn. definovat matematické vztahy popisující reálný systém, z poznatků, které jsem získal v kapitole 2.1 „Analýza chování systému“, kde jsem jako vstupně-výstupní zařízení použil platformu Arduino Nano.

2.3 Identifikace reálného systému

V této části se zabývám stanovením matematických vztahů popisujících chování reálného systému. K definování těchto aproximativních vztahů přistupuji experimentální metodou, kdy na vstup systému pouštím měřitelný signál, tj. výkon motoru, a sleduji výstup systému, tj. výchylka klapky v čase, který je taktéž měřitelný. Následně definuji vztah mezi těmito naměřenými daty pomocí následujících metod a nástrojů.

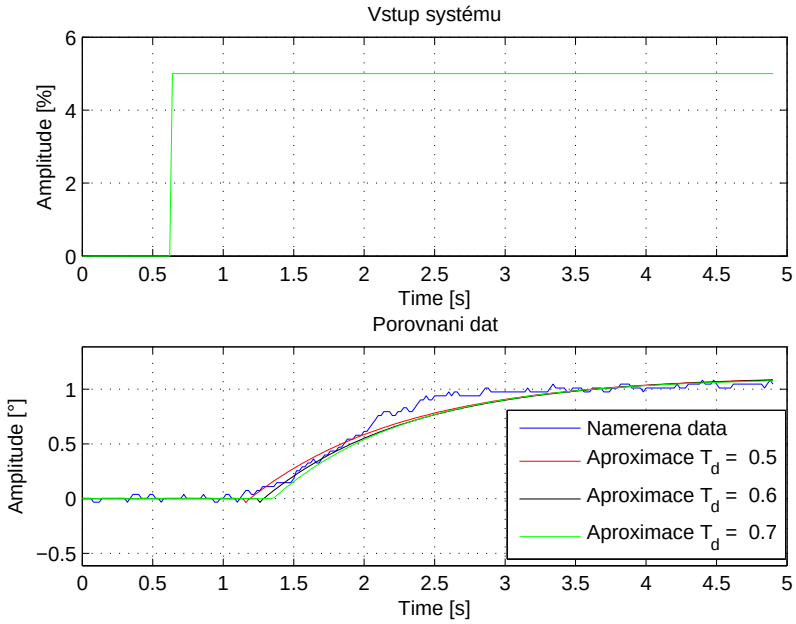
Systém jsem se rozhodl identifikovat v deseti pracovních bodech. Začínám identifikovat od 2° postupně zvyšuji výchylku o 2° a horní mez jsem si stanovil 20°, kolem takto získaných pracovních bodů definuji malé okolí (cca. ±1°), kde považuji chování systému za lineární.

Z předešlé analýzy (viz. sekce: 2.1 Analýza chování systému) vyplývá, že kolem každého pracovního bodu se nachází osm charakteristických rysů v chování systému, které bude zapotřebí identifikovat zvlášť jako několik podsystémů. Jestliže jsem systém rozdělil na deset pracovních bodů a v každém takovém bodě existuje osm podsystému, bude zapotřebí identifikovat osmdesát podsystémů. Nejprve si ovšem stanovím metodu, kterou k této identifikaci využiji.

2.3.1 Metoda nejmenších čtverců

Teorii a definování kauzálních metod pomocí této metody jsem již představil v kapitole 1.3.1 „Metoda nejmenších čtverců“. V této části se dále zabývám využitím a aplikací této metody na identifikaci zadaného modelu vzduchotechnické soustavy.

Pomocí softwaru MATLAB, kde jsem napsal jednoduchý algoritmus pro výpočet regresního modelu z naměřených dat. Simulací jsem ověřil jednak správnost algoritmu a jednak i shodu této metody s naměřenými daty. Simulaci jsem provedl pro několik dopravních zpoždění tak, abych stanovil co nejoptimálnější dopravní zpoždění.



Obr. 2.9: Regresní model v pracovním bodě 12°

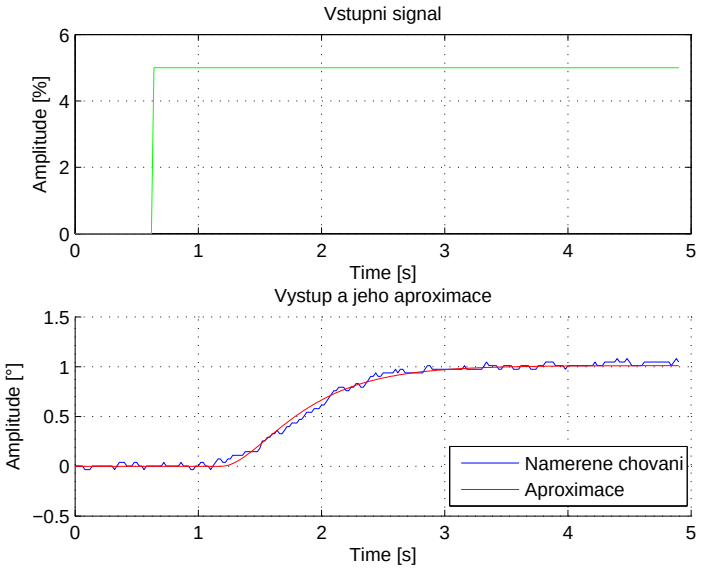
Z obrázku č. 2.9 je vidět několik výsledků aproximace reálného systému pomocí metodou nejmenších čtverců pro několik dopravních zpoždění. Nejlépe dopadla regrese pro případ, kdy dopravní zpoždění T_d bylo rovno $0.6\,[s]$ (černá křivka). Ovšem na první pohled všechny regresní modely mají chování, které je charakteristické pro první řád systému. Ovšem strukturu, kterou jsem si zadefinoval pro regresi byl systém druhého řádu. Při bližším prozkoumání je vidět, že systém je opravdu druhého řádu, ale jedna časová konstanta je velmi blízká nule. Aproximace tímto způsobem tedy nebyla dostatečně blízká naměřenému chování, proto jsem se rozhodl využít i další možnost identifikace a porovnat výsledky.

2.3.2 Nástroj System Identification Toolbox

Nástroj velmi stručně popisuji v teoretické části práce(viz. 1.3.2 Identifikace pomocí MATLABu). Do prostředí nástroje jsem naimportoval vstupní a výstupní data, která jsem naměřil na reálném systému. Využívám stejná data jako v předešlém případě, abych mohl rozhodnout jakou metodu budu dále využívat k další aproximaci zbývajících podsystémů.

Opět zvolím strukturu modelu systému druhého řádu s dopravním zpožděním. Z naměřených dat lze v některých případech velmi snadno určit statické zesílení a ulehčit tak identifikaci tím, že v nástroji stanovíme parametr určující právě toto

statické zesílení na pevně danou hodnotu odečtenou z naměřených dat (poměr ustálených stavů vstupu a výstupu). V některých případech se ovšem ukázalo, jedná se především o vetší výchylky, kdy systém vykazuje určité zašumění výstupní veličiny, aby nástroj sám stanovil nejoptimálnější hodnotu tohoto parametru, většinou se jedná o střední hodnotu signálu v ustáleném stavu.



Obr. 2.10: Aproximace v pracovním bodě 12°

Hned na první pohled je jasné, že tato aproximace (obr. č. 2.10) je daleko přesnější než aproximace metodou nejmenších čtverců a proto ji budu dále využívat pro identifikaci zbývajících podsystémů.

Výchozí tvar přenosu systému, ve kterém si nechám od nástroje pro identifikaci systémů generovat aproximativní modely, je následující:

$$F(s) = \frac{K}{(t_1 s + 1)(t_2 s + 1)} e^{T_d s},$$

kde $\mathbf{K}$ je statické zesílení systému, $\mathbf{t_1}$ a $\mathbf{t_2}$ jsou časové konstanty systému a $\mathbf{T_d}$ udává časové dopravní zpoždění systému.

Pro tento konkrétní pracovní bod a konkrétní podsystém má pak přenos tvar:

$$F(s) = \frac{0.2021}{(0.32s + 1)(0.40s + 1)} e^{0.55s},$$

Tímto způsobem aproximuji zbylé podsystémy, které jsem si v předchozích kapitolách stanovil.

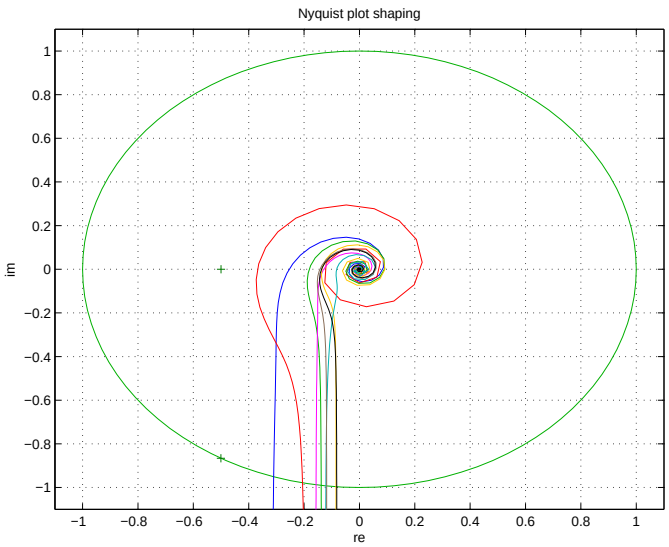
2.4 Návrh parametrů PID regulátorů

Jak bylo nastíněno v kapitole 1.4 (Metody návrhů regulátorů),kde také popisuji jak metoda funguje, využiji k odhadu parametrů PID regulátoru, tj. proporcionální složku, integrační a derivační časovou konstantu, popřípadě konstantu určující filtraci derivační složky, metodu robustních regionů, která je implementována v prostředí PIDlab.

Regulátory postupně navrhuji pro každý pracovní bod tak, aby regulátor obsluhující určitý pracovní bod byl dostatečně robustní, aby zvládl regulovat všechny charakteristické rysy definující chování na okolí tohoto pracovního bodu.

V předchozích kapitolách jsem nadefinoval, že okolí pracovního bodu je charakterizováno osmi podsystémy. Tyto podsystémy vložím do prostředí PIDlabu a tvarováním Nyquistových křivek získám tzv. robustní regiony, které definují skupinu parametrů splňující dané požadavky na bezpečnost v zesílení, jenž jsem stanovil $G_m = 2$, a bezpečnost ve fázi $P_m = 60°$.

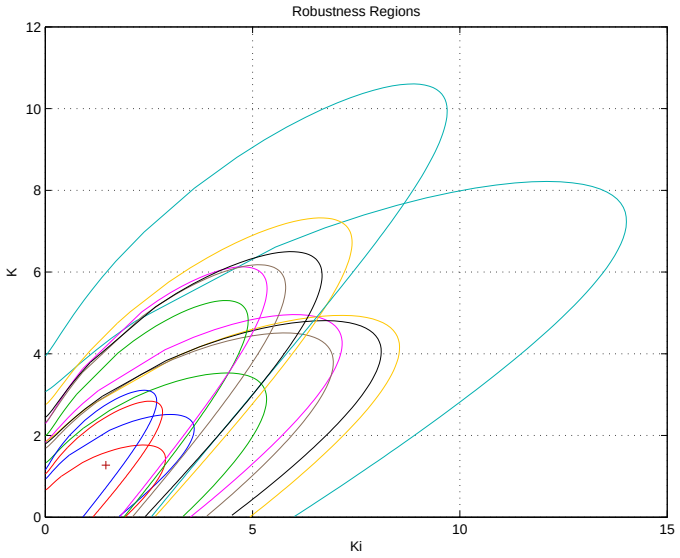
V následujících grafech demonstruji jak takový návrh parametrů regulátoru pomocí tohoto nástroje probíhá pro konkrétní pracovní bod 12°.



Obr. 2.11: Nyquistova křivka otevřené reg. smyčky pro prac. bod 12°

Na obrázku č. 2.11 je vyobrazena Nyquistova křivka (v komplexní rovině) otevřené regulační smyčky pro každý podsystém, který se nachází v tomto pracovním

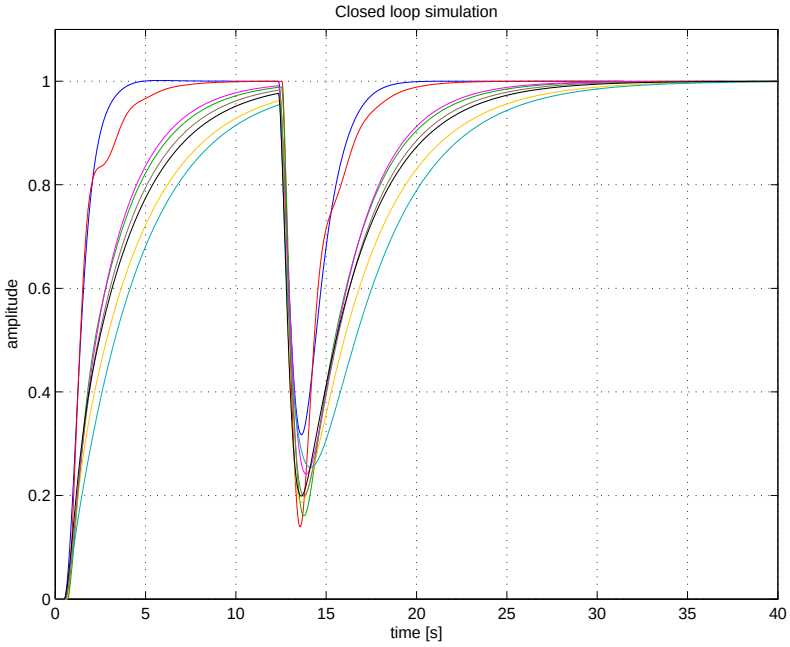
bodě. Dále jsou zvoleny dva tvarovací body určující bezpečnost v zesílení, tomu odpovídá v komplexní rovině bod o souřadnicích $[-0.5 \quad 0]$, a bezpečnost ve fázi se souřadnicemi $\left[-0.5 \quad -\frac{\sqrt{3}}{2}\right]$. K takto definovaným tvarovacím bodům vznikne šestnáct, pro každý tvarovací bod osm robustních regionů definující možné parametry regulátoru pro každý podsystém zvlášť, viz. obr. č. 2.12. Pokud ovšem potřebuji navrhnout pouze jeden regulátor, který bude schopen regulovat všechny podsystémy v daném pracovním bodě, musím zvolit takové parametry, které jsou v průniku všech takto vzniklých regionů. Vetšinou nejlepší vlastnosti mají ty parametry, které mají nejvyšší možnou hodnotu K_i, ovšem v některých případech je vhodnější volit jiné hodnoty, tak aby se dosáhlo požadovaného tvaru přechodové charakteristiky.



Obr. 2.12: Robustní regiony pro prac. bod 12°

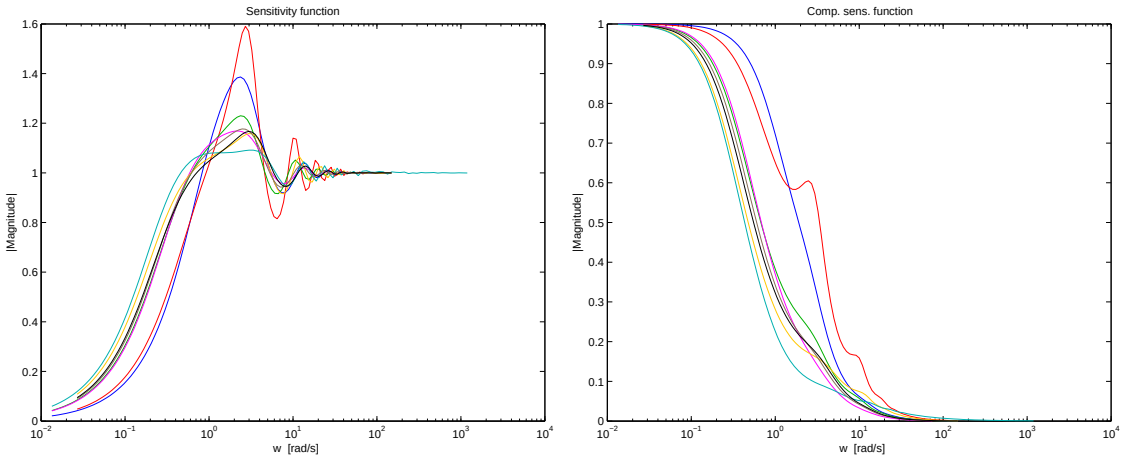
Nástroj hned umožní náhled jednak na průběh uzavřené regulační smyčky a jednak i náhled na čtyři základní citlivostní funkce systému.

Na obr. č. 2.13 je vidět teoretické chování uzavřené regulační smyčky, kdy na vstup je přiveden jednotkový skok a v čase cca 12 s. je vidět odezva na skokovou chybu. Dvě chování se vyznačují agresivnějším průběhem. Ostatní chování se spíše vyznačují pomalejším náběhem k požadované veličině. Ve všech pracovních bodech jsem se snažil navrhnout takové parametry regulátoru, aby přechodová charakteristika uzavřené regulační smyčky neměla velký překmit a sledovaná veličina najela plynule na požadovanou hodnotu.



Obr. 2.13: Chování uzavřené regulační smyčky prac. bod 12°

V následujících dvou grafech je vyobrazen průběh citlivostní funkce uzavřené regulační smyčky a komplementární citlivostní funkci uzavřené regulační smyčky.



Obr. 2.14: Citlivostní funkce regulační smyčky pro prac. bod 12°

Zbývající dvě citlivostní funkce uzavřené regulační smyčky jsou vyobrazeny na obr. č B.3 a obr. č. B.4, viz. příloha-grafy.

Tímto způsobem postupuji i při návrhu zbývajících devíti regulátorů a výchozí parametry jednotlivých PID regulátoru jsou uvedeny v následující tabulce:

Tab. 2.1: Parametry jednotlivých PID regulátorů

	Parametry regulátoru			
Prac. bod	$\mathbf{K_P}$	$\mathbf{T_I}$	$\mathbf{T_D}$	**N**
2	0.8433	0.6709	0.1677	2.0
4	0.5053	0.7148	0.1787	2.0
6	1.1266	0.7176	0.1794	2.0
8	0.8612	0.7427	0.1856	2.0
10	0.7219	0.7798	0.1949	2.0
12	1.2733	0.8709	0.2177	2.0
14	1.0400	1.2561	0.6280	2.0
16	1.0529	1.1285	0.2821	2.0
18	1.3731	1.2315	0.6157	2.0
20	1.6500	1.3562	0.3390	2.0

Výchozí zákon řízení, kterým se regulátor řídí má pak předpis:

$$U(s) = K_P \left\{ bW(s) - Y(s) + \frac{1}{T_I s} \left[W(s) - Y(s) \right] + \frac{T_D s}{\frac{T_D}{N} s + 1} \left[cW(s) - Y(s) \right] \right\}$$

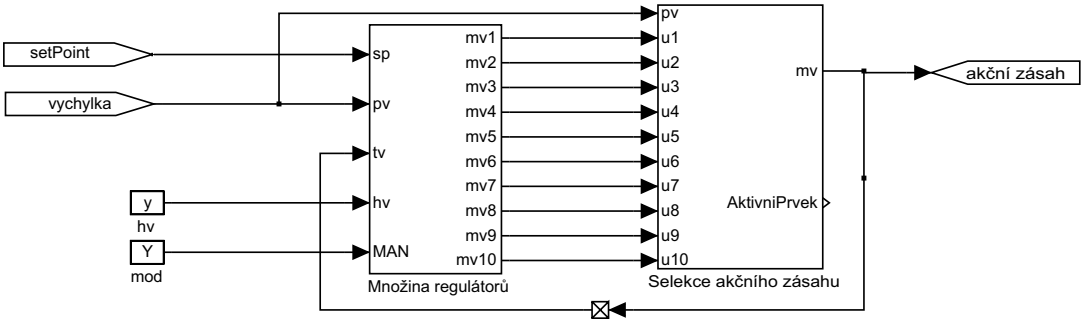
Koeficienty b a c definují zdali se jedná o 1DOF (b,c = 1) regulátor či 2DOF regulátor. V tomto případě necháme tyto koeficienty na výchozích nastaveních a to $b = 1$ a $c = 0$. K_P je pak statické zesílení regulátoru, T_I určuje časovou konstantu integrační složky, T_D určuje časovou konstantu derivační složky a N je filtrace derivační složky.

2.5 Návrh algoritmu řízení

V této části práce se zabývám samotným návrhem řídicího algoritmu, který následně implementuji ve dvou řídicích systémech a to nejprve v řídicím systému od firmy REX Controls s.r.o. a podruhé pak v řídicím systému od firmy Resologis inc.

Už od začátku, kdy jsem začal analyzovat systém a poté jej rozdělil do několika pracovních bodů, kolem kterých uvažuji lineární chování systému, jsem mířil k tomu, abych mohl navrhnout tzv. „Gain scheduling“ řízení. Jedná se o druh řízení, kdy nelineární systém je rozdělen do několika pracovních bodů, kolem nichž je systém řízen lineárními regulátory. Přepínání mezi jednotlivými regulátory je zajištěno sledováním výstupní veličiny, výchylky klapky v čase. Jakmile se výchylka blíží k určitému pracovnímu bodu je vybrán ten akční zásah, který je k tomuto okolí přidružen.

Blokově to může vypadat následovně:



Obr. 2.15: Algoritmus řízení blokové schéma

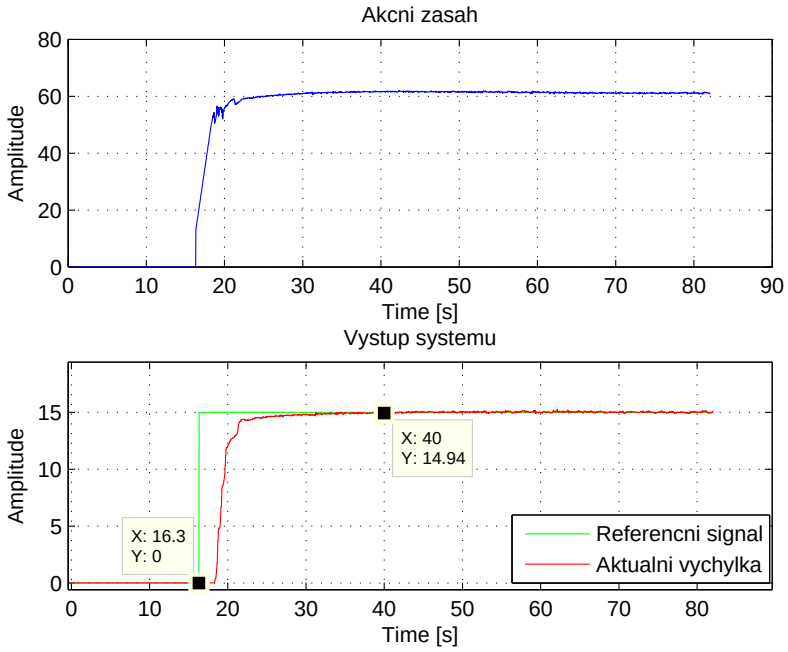
Vybraný akční zásah je pak přiveden jako zpětná vazba ke všem regulátorům. To je z důvodu vysledování akčního zásahu, aby při přepínání mezi jednotlivými regulátory nevznikal v akčním zásahu nechtěný ráz.

Samozřejmě je počítáno i s možností manuálního módu regulace, kdy lze přímo ovládat výkon motoru i v tomto případě by mohl vzniknout ráz v akčním zásahu při přepnutí zpět do automatického módu regulace, ovšem i toto je bezpečně ošetřeno vysledováním akčního zásahu.

2.5.1 Implementace algoritmu řízení v systému REX, IO jednotka Turck BL20

Algoritmus řízení jsem implementoval pomocí standardních bloků systému REX a vytvořil tak blokové schéma podobné jako je na obr. č. 2.15. Komunikace mezi algoritmem řízení, který je implementován na Raspberry Pi, na kterém běží výpočetní jádro systému REX, tzv. RexCore, je zajištěna pomocí protokolu MODBUS.

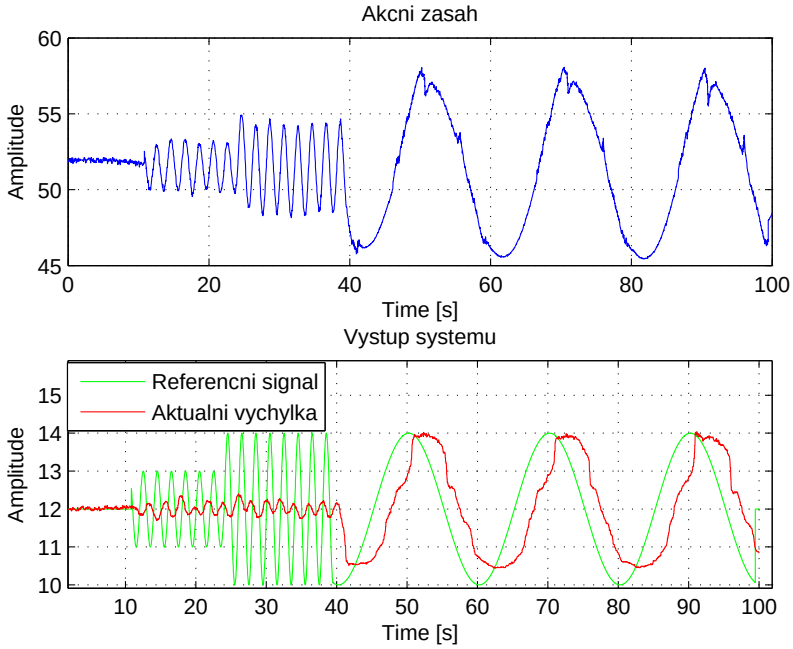
Výsledný algoritmus jsem otestoval na reálném modelu a výsledky jsou vyobrazeny v následujících grafech.



Obr. 2.16: Regulace na konstantní hodnotu 15°

Z tohoto grafu je vidět chování uzavřené regulační smyčky při nastavení požadované hodnoty 15° a počáteční hodnota je 0°. Je vidět, že požadavek na vstup systému je přiveden v čase cca 16 s. a v čase 40 s. je již výchylka ustálena na požadované hodnotě. Celý přechodový děj tedy trvá cca 24 s. Také je vidět, že nedochází k překmitu sledované veličiny, regulátor se spíše jeví jako opatrný zejména v okolí požadované hodnoty. Tento jev je zapříčiněn robustností regulátoru.

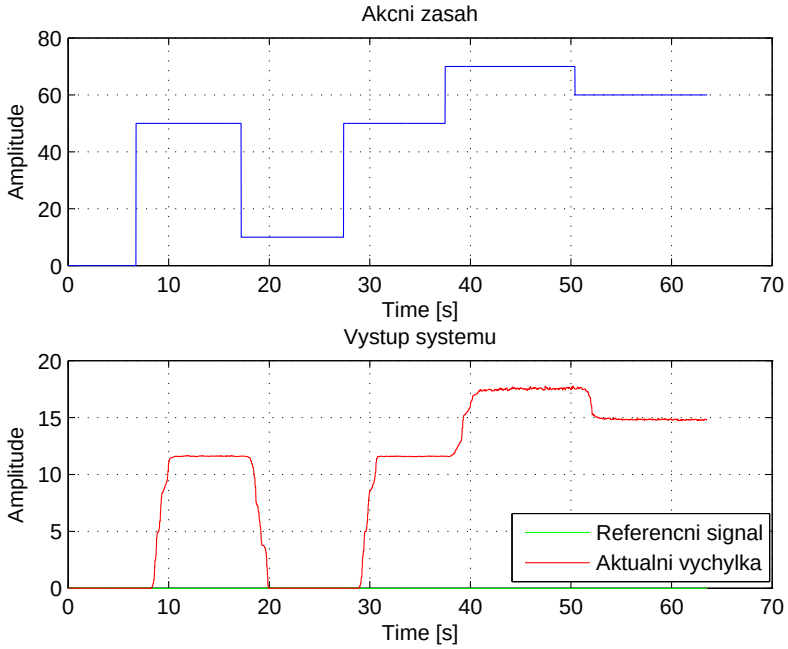
Dále jsem regulační smyčku testoval na sledování referenčního signálu, kdy jsem na vstup přivedl sinusový signál a postupně měnil periodu kmitání. Výsledek je vyobrazen na obr. č. 2.17.



Obr. 2.17: Sledování referenčního signálu

Z těchto průběhů je vidět, že uzavřený systém při vyšších frekvencích změn referenčního signálu sice reaguje, ale nestíhá jej sledovat. Při zmenšení periody kmitání referenčního signálu již výstupní veličina má sinusový charakter, ale k originálnímu signálu, který je přiveden na vstup systému, má velmi daleko. Lepšího výsledku by se dalo docílit ještě větším zmenšením frekvence kmitání referenčního signálu, ale v systému je jednak přítomné dopravní zpoždění, které sledování rychlých změn vstupního signálu komplikuje a jednak je regulace navržena za účelem regulace na konstantní hodnotu.

Posledním požadavkem na uzavřenou regulační smyčku je možnost manuálního ovládání výkonu motoru. I tento požadavek je v algoritmu řízení zahrnut a lze tedy snadno regulaci přepnout do manuálního řízení a přímo ovládat výkon motoru. Opět jsem ověřil funkčnost přímého ovládání výkonu motoru experimentem, jehož výsledek je vyobrazen na obr. č. 2.18.



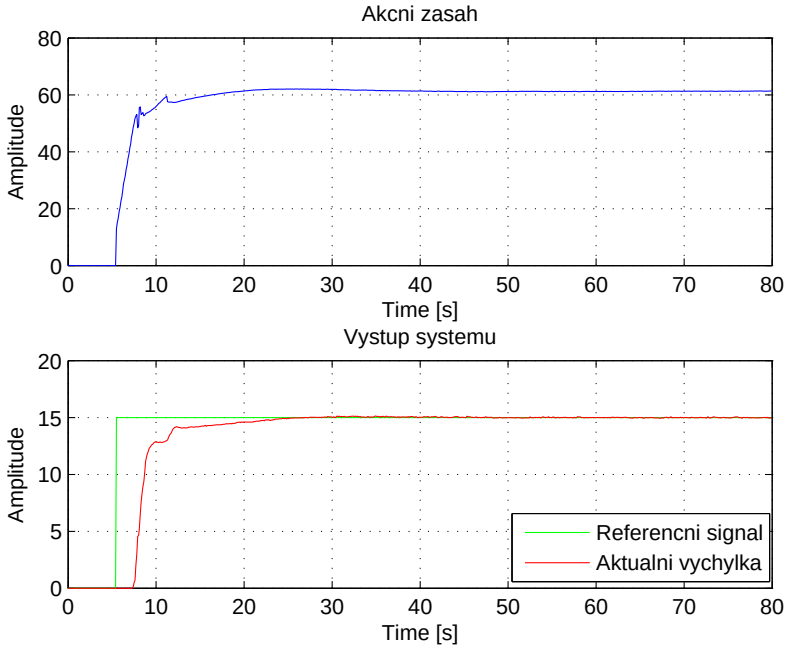
Obr. 2.18: Regulace v manuálním režimu - REX

2.5.2 Implementace algoritmu řízení v systému Resologis IPC, IO jednotka Turck BL20

Řídicí systém Resologis IPC umožňuje návrh řídicích algoritmů několika možnými způsoby například strukturovaným textem, vývojovým diagramem a v neposlední řadě i pomocí blokových schémat. Pro implementaci řídicího algoritmu jsem si zvolil způsob návrhu algoritmu pomocí funkčních bloků. V několika případech mi nestačila klasická knihovna funkčních bloků definována výrobcem a proto jsem využil možnosti návrhu vlastních funkčních bloků. Definování vlastních funkčních bloků se provádí programováním skrze strukturovaný text, kde se definuje chovaní, vstupy a výstupy onoho bloku. Řídicí systém pak sám z tohoto strukturovaného textu vygeneruje funkční blok, který se chová jako nadefinovaná funkce.

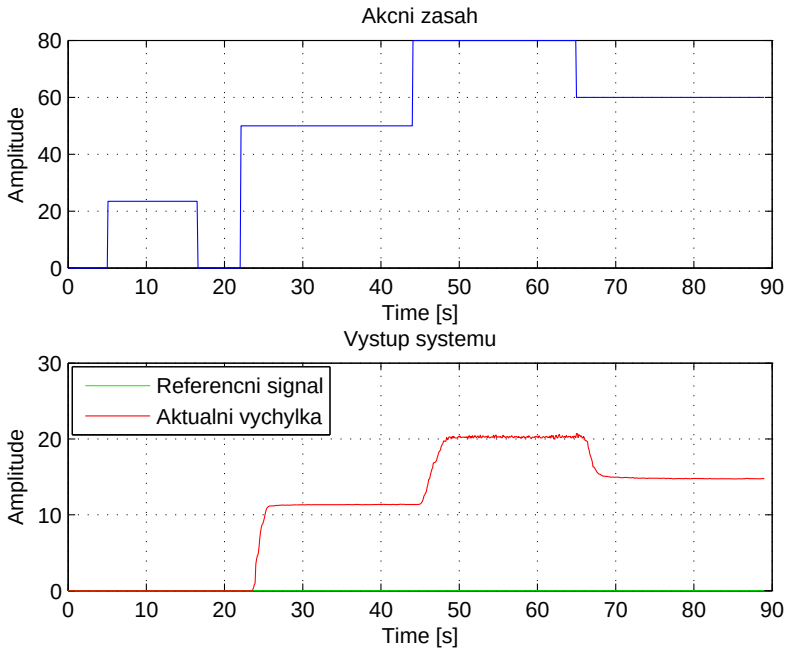
Algoritmus nebylo zapotřebí nikterak měnit a nebylo nutné ani přepočítávat konstanty regulátorů. Implementovaný ovládací algoritmus jsem opět vyzkoušel na reálném systému pomocí několika následujících experimentů.

První experiment, jako v případě implementace za pomocí řídicího systému REX, byl regulace na konstantní hodnotu, kdy požadovaná hodnota vychýlení klapky byla opět nastavena na 15°. Průběh experimentu je vyobrazen v následujícím grafu, viz. obr. č. .



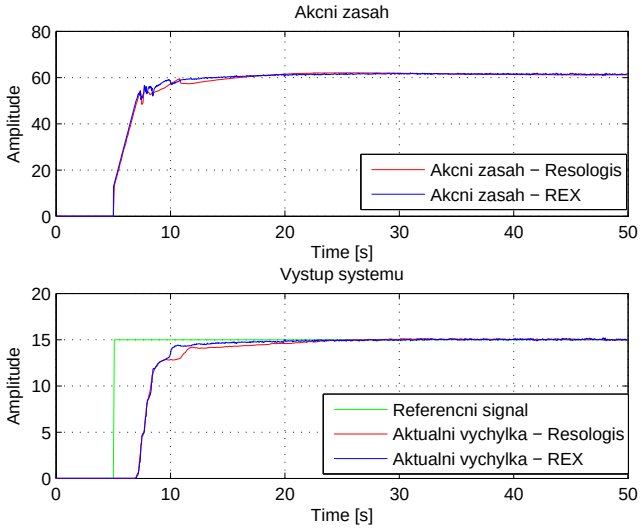
Obr. 2.19: Regulace na konstantní hodnotu 15° - Resologis

V dalším experimentu jsem opět testoval regulaci v manuálním režimu a přímo tak ovládal výkon motoru.



Obr. 2.20: Regulace v manuálním režimu - Resologis

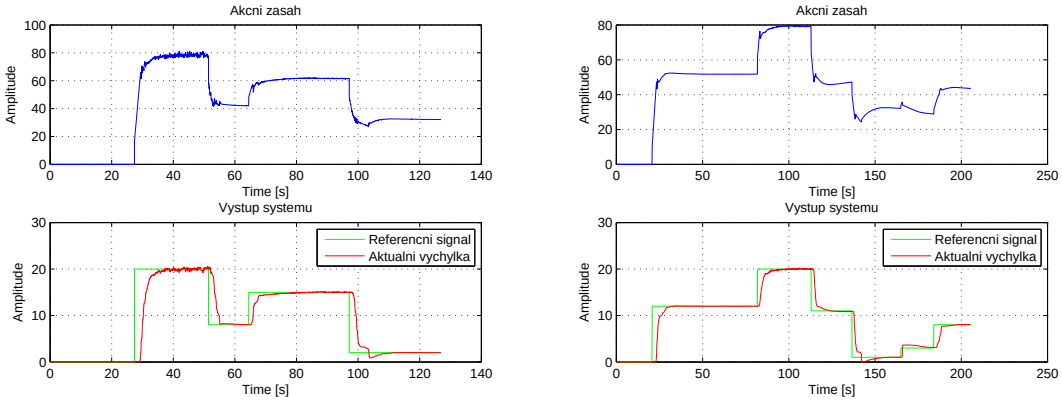
Nyní jsem se rozhodl porovnat implementace mezi sebou a ověřit tak, že algoritmus řízení pracuje stejně nezávisle na řídicím systému, ve kterém je implementován.



Obr. 2.21: Sorvnání implementací řídicího alg.

Na tomto grafu je vyobrazen jednak průběh aktuální výchylky klapky a jednak i akční zásah regulátoru. Průběhy těchto veličin v jednotlivých implementací, jak můžeme vidět, je téměř totožný.

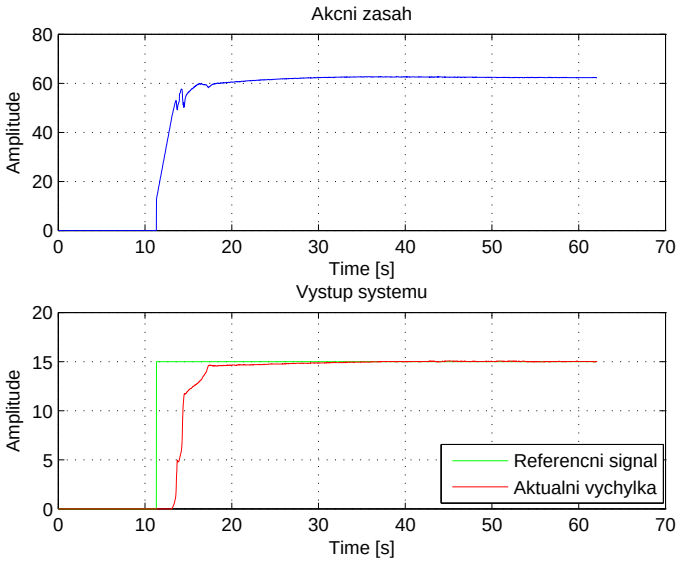
Funkčnost algoritmů v jednotlivých implementacích řídicích systémů jsem odzkoušel ještě několika skoky požadovaných hodnot, kde je vidět, že navržené regulátory dokáží regulovat správně na celé množině hodnot, které výchylka klapky může nabývat.



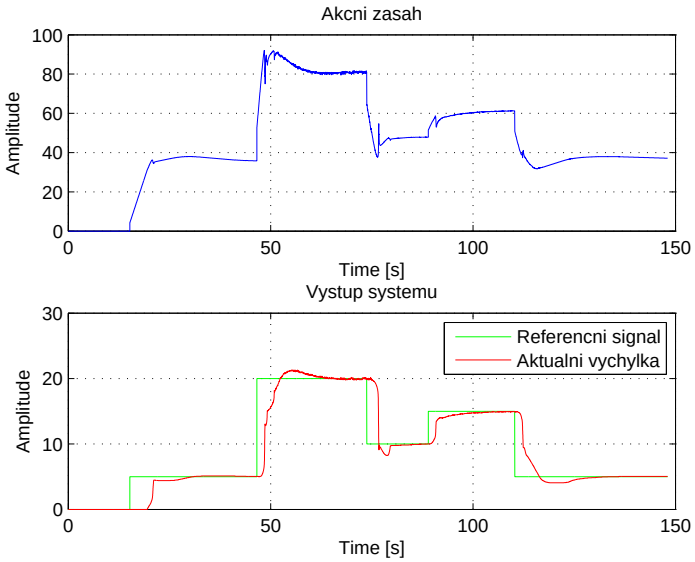
Obr. 2.22: Skoky požadované hodnoty

2.5.3 Implementace algoritmu řízení v systému REX, IO jednotka Arduino Nano

Nyní jsem jako vstupně-výstupní zařízení použil Arduino Nano, do kterého jsem nahrál algoritmus nazvaný „Rexduino“, aby se tato platforma chovala jako vstupně-výstupní zařízení. Opět jsem provedl několik experimentů jejichž výsledek je vyobrazen v následujících grafech.

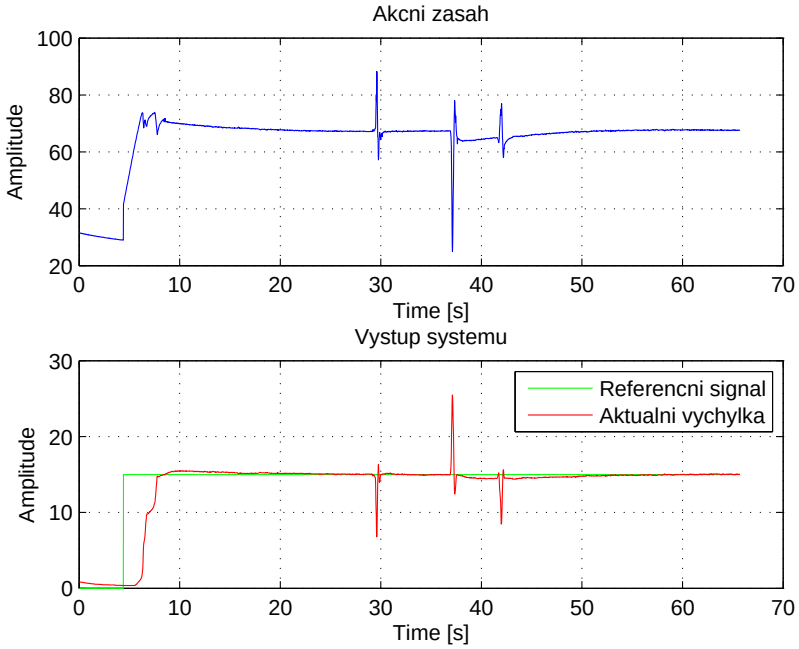


Obr. 2.23: Regulace na konstantní hodnotu 15°



Obr. 2.24: Odezva systému na skokové změny vstupu

Dále jsem regulační smyčku otestoval na odregulování chyby, která je zavedena na výstup systému. Tuto chybu jsem provedl jemným vychýlením klapky z ustáleného stavu. Výsledek je vyobrazen na následujícím grafu č. 2.25.



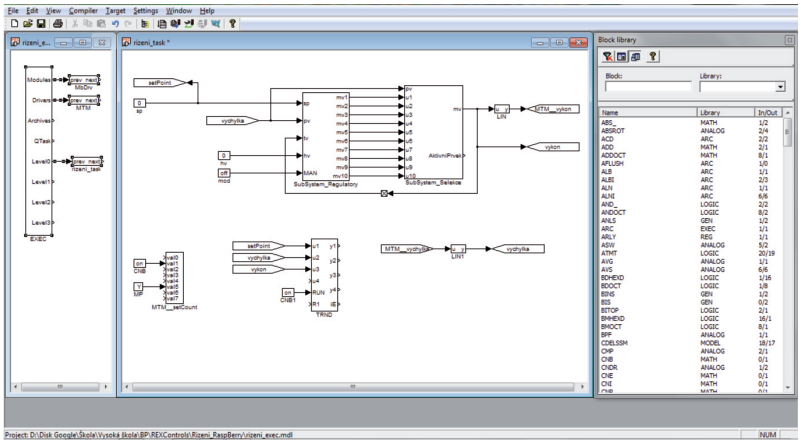
Obr. 2.25: Odezva systému na chybu působící na výstup systému

Z grafu je nejprve vidět odezva systému na skokovou změnu vstupu, kdy požadované vychýlení klapky je 15°. Systém jsem nechal ustálit a po ustálení jsem klapku ručně vychýlil z tohoto ustáleného stavu. Chybu do systému jsem zavedl opakovaně. Odezva na první zavedenou chybu je vidět cca v čase 29 s. Je vidět, že regulátor ihned reaguje a snaží se výchylku dostat zpět do požadované hodnoty. Poté jsem systém opět nechal ustálit a vychýlil klapku znovu opačným směrem a vetší silou než v předešlém případě. Opět je vidět okamžitá reakce a snaha o kompenzaci.

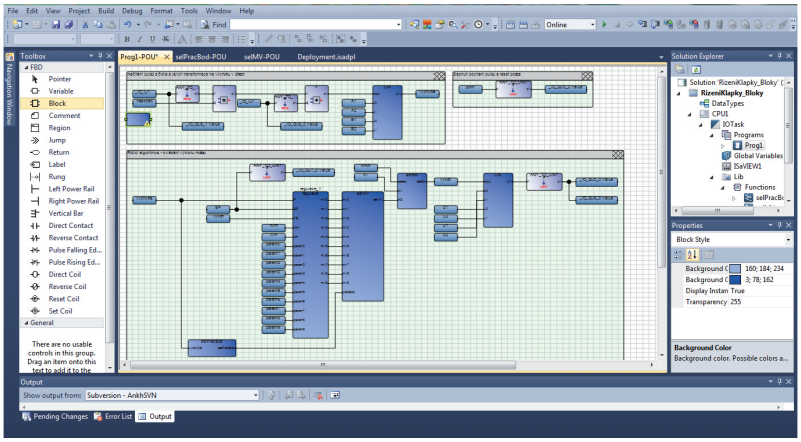
2.6 Porovnání řídicích systémů

V této kapitole mezi sebou porovnávám prostředí použitých řídicích systémů a jelikož jsem v obou případech algoritmus řízení navrhoval pomocí funkčních bloků, tak také subjektivně porovnávám kvalitu a množství funkčních bloků. Jde pouze o základní porovnání, protože ani jeden z řídicích systémů jsem nezkoumal do úplných základů a toto porovnání lze tedy brát spíše jako první dojmy z uživatelského hlediska.

V následujících dvou obrázcích (obr. č. 2.26 a obr. č. 2.27) je vidět grafické rozhraní nástrojů pro návrh řídicích algoritmů pro jednotlivé řídicí systémy (RexDraw pro řídicí systém REX a ISaGRAF pro řídicí systém Resologis IPC).



Obr. 2.26: Prostředí návrhového nástroje RexDraw

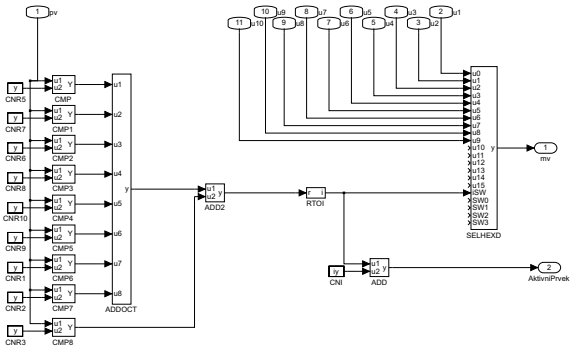


Obr. 2.27: Prostředí návrhového nástroje ISaGRAF

Při porovnání návrhového prostředí je vidět, že firma REX Controls s.r.o. spíše vsadila na jednoduchost a přehlednost. Naopak firma Resologis inc. si dala vetší práci s grafickým provedením a velikým množstvím oken, záložek a panelů, které při návrhu algoritmu využijeme opravdu všechny a musíme mezi nimi mnohokrát přepínat a ne všechna potřebná okna si je možno nechat zobrazená, neboť by pak zbylo malé místo pro vyobrazení hlavní části, kde se vyskytuje navrhovaný algoritmus. Ovšem po delší době si uživatel zvykne na toto chování a první dojem už není tak nepříznivý.

V obou vyobrazení je vidět návrh téhož algoritmu řízení. Z porovnání je vidět, že hlavní část algoritmu je prakticky shodná až na pár maličkostí. V obou případech provedení je proměnná „výchylka“ označující aktuální vychýlení klapky ve stupních, množina regulátorů pro řízení metodou „Gain scheduling“ a pak také blok selekce akčního zásahu. Blok zajišťující výběr akčního zásahu regulátoru jsem v obou případech implementoval sám a v každém návrhovém prostředí je blok implementován rozdílně.

V systému RexDraw jsem blok navrhl pomocí základních bloků tohoto systému a algoritmus výběru pak vypadá následovně (obr. č. 2.28):



Obr. 2.28: Funkční blok selekce provedení v REXu

Pomocí bloku „SELHEXD“, který má na vstup přivedeny signály mezi nimiž se má přepínat a na vstup „iSW“ je přiveden index vybíraného signálu. Zjištění indexu vybíraného prvku je pak provedeno pomocí několika komparátorů, které porovnávají aktuální výchylku klapky s nadefinovanou hranicí, která je pro každý komparátor jiná. Pomocí těchto hranic je nadefinované okolí kolem pracovních bodů. Výsledkem porovnání je logická jedna nebo nula. Takto vzniklé signály jsou pak sečteny pomocí bloku „ADD“ a tím se získá index prvku, který má být vybrán.

V návrhovém prostředí ISaGRAF je tento problém selekce řešen odlišným způsobem. Při návrhu tohoto bloku v tomto prostředí jsem využil možnost, kterou vývojové prostředí umožňuje a to kombinaci programování pomocí funkčních bloků a programování pomocí strukturovaného textu. Porovnání aktuální výchylky s pevně stanovenou hranicí definující okolí kolem pracovního bodu je zde provedeno pomocí podmínky „if“, kde výsledek podmínky je nula nebo jedna podle toho zda-li je aktuální výchylka větší nebo menší než daná hranice. Na konci této funkce jsou výsledky podmínek sečteny a je zjištěno v jakém okolí se aktuálně klapka vyskytuje a podle toho je vybrán právě ten regulátor, který je tomuto okolí přidružen.

Nadefinované funkce jsou pak uloženy a prostředí automaticky vygeneruje funkční blok, který danou funkci realizuje. Takto lze snadno nadefinovat vlastní funkční bloky bez znalostí dalších programovacích jazyků. V prostředí RexDraw lze také nadefinovat vlastní funkční bloky ovšem je zapotřebí znalost dalšího programovacího jazyka a syntaxe pro definování funkčních bloků pro řídicí systém REX. Ovšem v prostředí RexDraw je knihovna funkčních bloků velmi bohatá a obsahuje základní funkční bloky, které lze snadno poskládat tak, aby plnily potřebnou funkci. V prostředí ISaGRAF je knihovna o něco chudší, ale tento nedostatek je pak kompenzován právě snadným nadefinováním vlastních funkčních bloků.

V systému REX lze snadno sledovat, co se děje v jádře řídicího systému při jeho běhu pomocí programu RexView. Program poskytuje detailní hierarchicky uspořádané informace o všech subsystémech jádra. Komunikace pomocí protokolu TCP/IP umožňuje připojit se k běžícímu jádru na lokálním počítači, v lokální síti i ve vzdálené síti (např. přes Internet). Pomocí tohoto programu lze snadno i vykreslit průběhy jednotlivých veličin v čase. V prostředí ISaGRAF ovšem tato možnost není. K tomu abych vykreslil průběhy veličin v čase v tomto prostředí by bylo nutné stáhnout další program (OPC client) nadefinovat v řídicím systému OPC server a pomocí klienta se připojit k serveru a sledovat veličiny tímto způsobem.

V obou provedení řídicího algoritmu jak v systému REX tak i v systému Resologis provádím vykreslení dat pomocí systému REX. V systému REX data vykresluji pomocí funkčního bloku „TRND“, který je za tímto účelem navržen. V prostředí ISaGRAF je vykreslení dat provedeno o něco komplikovaněji. K tomu abych data vykreslil i z tohoto prostředí jsem využil protokolu „MODBUS“ a systému REX. V prostředí ISaGRAF zapisuji potřebná data a signály do jednotlivých registrů, ze kterých pak data zpětně získám z prostředí RexDraw a opět pomocí bloku „TRND“ a programu RexView data vykresluji.

2.7 Vizualizace modelu

Na začátku jsem si stanovil požadavky, které by měla aplikace splňovat. Aplikace by měla znázorňovat jednak daný model a jednak průběh veličin v čase. Dále by měla zahrnovat možnost ovládání modelu skrze vytvořenou aplikaci, tj. musí být možno přepnout regulátor z automatického módu do manuálního, zadat požadovanou hodnotu (tzv. SetPoint) a nastavení výkonu motoru v manuálním módu regulátoru.

Komunikaci mezi jádrem řídicího systému (RexCore) a samotným modelem využívám tzv. **WebSocket**. Jedná se o komunikační spojení, navázané mezi dvěma procesy, pomocí IP technologie. Socket je (zjednodušeně řečeno) popsán dvojicí IP adres, protokolem a portem, který je pro komunikaci použit. Služby na vyšší úrovni tento socket používají k (většinou obousměrnému) přenášení dat.

Pomocí webového prohlížeče se přes WebSocket, připojím k reálnému modelu vzduchotechniky, který je ovládán nástrojem REX Controls. Tento nástroj již obsahuje virtuální server, který nám umožní již zmíněnou komunikaci mezi webovým klientem a algoritmem ovládající náš model vzduchotechniky.

2.7.1 Struktura webového klienta

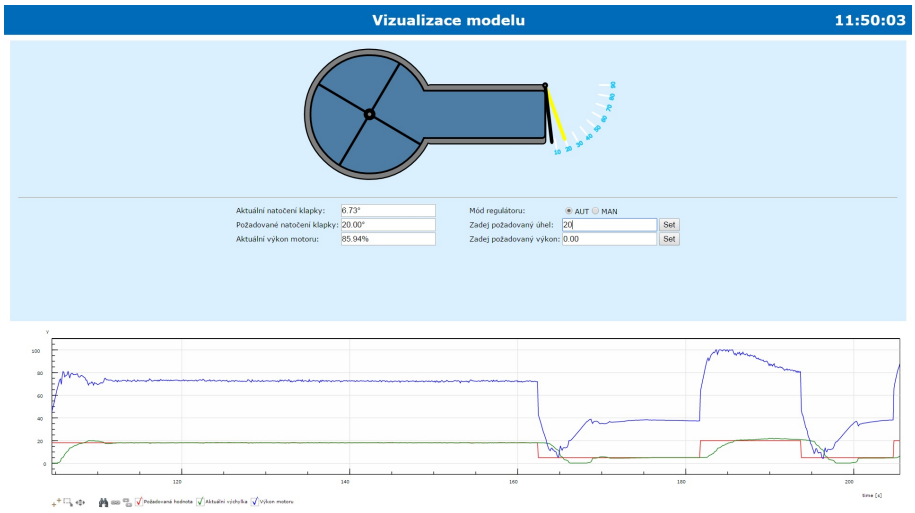
Klientský program je rozdělen do několika částí:

1. **Vizuální část**
2. **Část popisující funkčnost modelu**
3. **Panel zobrazující průběh veličin**

2.7.2 Vizuální část

Tato část je napsána v HTML jazyce a má za úkol vizuálně reprezentovat reálný systém na obrazovce uživatele pomocí webového prohlížeče.

K animaci je využito vektorové grafiky, formát svg. Pomocí několika prvků (circle, rectangle, apod.) je vytvořen jednoduchý schématický obrázek reálného modelu ve 2D (viz. obr. č. 2.29).



Obr. 2.29: Vizualizace modelu - webový klient

Model se skládá ze tří hlavních částí:

1. rotoru, který je realizovaný jako vrtule usazená na otáčející se osičce, kde vzniká proud vzduchu.
2. dopravní potrubí, které přenáší proud vzduch od motoru dále.
3. klapky, jenž je umístěna na konci potrubí tak, aby se proud vzduchu do ní opíral maximální silou.
4. ciferníku znázorňující natočení klapky v úhlech

Takto vytvořený objekt je ovšem statický. Dynamiku mu dodává až další část. Ovšem tato část nemá za úkol pouze vizuální interpretaci modelu, ale také obsahuje jak sdělovací, tak i ovládací prvky.

Do sdělovacích prvků je zahrnuto několik textových polí, ve kterých jsou zobrazeny aktuální hodnoty načtené z reálného modelu(aktuální poloha klapky, požadované natočení klapky a výkon motoru).

V ovládacích prvcích nalezeme „radion button“ AUTO/MAN, díky kterému si přepínáme mezi jednotlivými módy regulátoru a podle toho buďto nastavujeme požadované natočení klapky a regulátor ovládá sám výkon motoru tak, aby byla klapka v požadované poloze, anebo přímo ovládáme výkon motoru sami. Dále v těchto prvcích nalezneme dvojici prvků pole a tlačítko potvrzující námi zadané hodnoty do příslušných polí. Jedno pole slouží k zapsání hodnoty jako požadované natočení klapky a druhé k nastavení výkonu motoru.

2.7.3 Funkční část programu

Tato část má za úkol načíst data z reálného systému a předat je dále ke zpracování a následné vizualizaci. Tato část je psána v jazyce Java Script.

Hned na začátku je zajištěna komunikace mezi Rexem, který zde funguje jako překladač mezi webovým klientem a reálným modelem, a webovou aplikací. Dále je definované připojení ke správným připojovacím bodům, neboli veličinám, které chci interpretovat (natočení klapky v úhlech, výkon motoru a režim regulace).

Dále z této části kódu zajišťuji, pomocí několika jednoduchých metod komunikaci a předávání hodnot jak do informativních polí na webové stránce tak i do polí, pomocí níž můžeme systém ovládat.

Podstatnou funkcí této části kódu je animovat vytvořený schématický model zobrazující se uživateli v prohlížeči. Animované jsou pouze dvě části a to rotor vytvářející proud vzduchu a klapka natáčející se kolem osy svého uchycení.

Algoritmus ovládající dynamiku modelu

Algoritmus načítá data každých 100 ms. Načte jak aktuální výkon motoru tak i aktuální polohu klapky a tyto veličiny dále zpracuje a předá webovému prohlížeči tak, aby se model animoval v závislosti na těchto dvou veličinách.

V obou případech animace jde o transformaci, kterou nazýváme rotací. Tuto transformaci nám umožňuje knihovna obsažena ve vektorovém formátu SVG, jedná se o atribut *„Transform(Rotate(angle))“*, který můžeme uplatnit na objekty, kterými je model vytvořen. Při aplikaci této funkce se objekt otočí o požadovaný úhel (angle). Samotná hodnota výkonu motoru by ovšem nezpůsobila plynulou rotaci objektu a proto do požadovaného úhlu natočení je započten i čas, tzn. každý časový okamžik vynásobíme úhlovou rychlostí a výkonem motoru. Takto vypočtený úhel předaný objektu pomocí funkce transformující jeho souřadnice již způsobí plynulou animaci v závislosti na výkonu motoru i na čase.

S animací klapky to bude daleko jednoduší. Jelikož hodnoty jsou načítány každých 100 ms a není potřeba, aby se klapka točila stále do kola kolem své osy, ale pouze v rozmezí cca 0 - 90°. Hodnota natočení klapky je přímo předaná do funkce zajišťující přepočet souřadnic daného objektu.

2.7.4 Panel zobrazují průběh veličin v čase

Tato část (skript) je převzata z příkladů, které jsou součástí nástroje Rex Controls. Tento skript jsem si stáhl a poupravil pro své potřeby. Na začátku se opět připojuji k modelu pomocí Web Socketu a nástroje REX Controls a opět načítám potřebné veličiny. Tyto veličiny (požadovaná hodnota, aktuální hodnota, výkon motoru) jsou dále předávány do panelu, tzv. „Trendu“, který vykreslí průběh jednotlivých veličin v čase.

3 ZÁVĚR

Primárním cílem této práce byl návrh řídicího systému pro reálný model vzduchotechnické soustavy. K tomu, aby byl navržen tento systém, bylo zapotřebí získat bližší povědomí o tom jak se daná soustava chová a toto chování aproximovat. Při analýze systému jsem zjistil, že soustava má nelineární charakter. Tuto vlastnost jsem ovšem předpokládal a několika experimenty jsem domněnku potvrdil. Nyní nastal problém jak navrhnout pro danou soustavu řídicí systém. Rozhodl jsem se, že pro vzduchotechnický model navrhnu řízení metodou „Gain scheduling“. Jedná se o metodu, kdy je nelineární systém rozdělen do několika pracovních bodů a kolem těchto bodů je systém řízen lineárními regulátory. Předpokládá se, že na malém okolí kolem těchto pracovních bodů má systém lineární charakter a lze tedy přepínáním regulátorů docílit kvalitního řízení na celém systému. Na základě této metody jsem systém rozdělil do deseti pracovních bodů, kolem kterých předpokládám lineární charakter chování. Kolem těchto bodů jsem systém opět analyzoval a přiblížil si chování kolem nich. Okolí každého pracovního bodu bylo charakterizováno čtyřmi vlastnostmi. Každou vlastnost bylo potřeba nějak definovat. Definování vlastností jsem klasifikoval jako problém čtyř podsystémů charakterizující daný pracovní bod a tyto podsystémy jsem popsal přenosovými funkcemi. K identifikaci, tedy získání matematického popisu, jsem využil několik metod, kdy jednu z nich, metodu nejmenších čtverců, jsem zavrhl, jelikož druhou metodou jsem docílil bližšího aproximativního chování. Tím jsem docílil lepšího kvalitnějšího návrhu řízení jednak pro daný pracovní bod, ale také i pro řízení celého systému. Problém řízení jsem se rozhodl řešit následovně. Každý pracovní bod a jeho okolí bude řízeno jedním PID regulátorem. Touto definicí jsem docílil toho, že systém bude řízen deseti PID regulátory mezi nimiž se na základě aktuální polohy klapky bude přepínat. Každý regulátor v pracovním bodě musí být natolik robustní, aby byl schopen uřídit čtyři podsystémy, které se v každém konkrétním bodě nacházejí a definují chování na tomto okolí. K navržení parametrů regulátoru jsem využil nástroje PIDlab, který používá k návrhu parametrů metodu robustních regionů, která je také známa pod názvem metoda návrhu parametrů regulátoru pomocí tvarovacích bodů. Touto metodou jsem navrhl vhodné parametry tak, aby systém nevykazoval veliký překmit popřípadě podkmit a choval se stabilně. Základní požadavky na bezpečnost v zesílení jsem u všech návrhů stanovil rovnou dvěma a stejně tak pro bezpečnost ve fázi, kterou jsem definoval šedesáti stupni. Po získání těchto hodnot parametrů pro jednotlivé regulátory bylo zapotřebí tuto teorii a teoretické hodnoty převést do reálu na reálný model.

Ze zadání plyne, že při návrhu řídicí soustavy mám brát v úvahu případ, kde jednou jako vstupně-výstupní jednotku použiji Arduino Nano a po druhé pak vstupně-

výstupní jednotku Turck BL20. Již při analýze systému jsem dokázal, že vstupně-výstupní zařízení nemá na chování systému žádný vliv a proto lze navrhnout jedno ekvivalentní řešení, které bude vykazovat při nejmenším velmi podobné chování. Jedním z požadavků na tuto práci plynoucí ze zadání je použití dvou řídicích systémů pro implementaci řídicího algoritmu. Musel jsem se tedy seznámit s oběma řídicími systémy. Systémy popisuji v samostatné kapitole v teoretické části a v praktické části jsem tomuto tématu také věnoval jednu kapitolu, kdy porovnávám řídicí systémy mezi sebou a popisuji první dojmy, které ve mě zanechaly. Řídicí algoritmus je v obou případech implementován pomocí funkčních bloků a v obou případech je shodný. Výpočetní jádro a tedy běh celého řídicího systému v obou případech zajišťuje minipočítač Raspbery Pi, které ve spojení s řídicími systémy tvoří levnou variantu řídicího zařízení. Navržené řízení jsem implementoval a otestoval na reálném modelu a opět jsem ověřil ekvivalentnost vstupně-výstupních zařízení.

Posledním cílem, který jsem si stanovil, byl návrh a realizace uživatelského rozhraní. Z rozhraní lze systém jednoduše ovládat, přepínat mezi manuálním módem regulace, kdy přímo uživatel definuje výkon motoru, a automatickým módem, kdy uživatel zadá požadované vychýlení klapky a řídicí systém obstarává výkon motoru. Součástí rozhraní je jednak grafické znázornění modelu v reálném čase, kdy dynamika tohoto znázornění se odvíjí od skutečných hodnot načítaných z reálného modelu. Jednotlivé veličiny měřené na systému, tj. aktuální vychýlení klapky, výkon motoru a požadovaná hodnota, jsou vykresleny do grafu v závislosti na čase. Grafické rozhraní je navrženo pouze pro případ, kdy je model řízen řídicím systémem REX.

LITERATURA

[1] REX Controls s.r.o. *Funkční bloky systému REX - referenční příručka* [online].Poslední aktualizace 7. 4. 2015 [cit. 4. 5. 2015] Dostupné z URL: <http://www.rexcontrols.cz/media/DOC/CZECH/BRef_CZ.pdf>.

[2] REX Controls s.r.o. *Začínáme se systémem REX na platformě Raspberry Pi - uživatelská příručka* [online]. Poslední aktualizace 7. 4. 2015 [cit. 4. 5. 2015] Dostupné z URL: <http://www.rexcontrols.cz/media/DOC/CZECH/REX_Getting_Started_RasPi_CZ.pdf>.

[3] REX Controls s.r.o. *Ovladač systému REX pro komunikaci Modbus (modulMbDrv) - uživatelská příručka* [online]. Poslední aktualizace 7. 4. 2015 [cit. 4. 5. 2015] Dostupné z URL: <http://www.rexcontrols.cz/media/DOC/CZECH/MbDrv_CZ.pdf>.

[4] Čech Martin, Schlegel Miloš *Návrh PID regulátoru přes Internet: www.PIDlab.com* [online]. Poslední aktualizace 9. 5. 2009 [cit. 4. 5. 2015] Dostupné z URL: <http://automatizace.hw.cz/files/images/files/PIDRegions_cz.pdf>.

[5] MathWorks *System Identification Toolbox* [online]. Poslední aktualizace 16. 4. 2015 [cit. 4. 5. 2015] Dostupné z URL: <http://automatizace.hw.cz/files/images/files/PIDRegions_cz.pdf>.

[6] Doc. Ing. Jiří Melichar, CSc. *LINEÁRNÍ SYSTÉMY 1* [online]. Poslední aktualizace 30. 8. 2011 [cit. 4. 5. 2015] Dostupné z URL: <https://courseware.zcu.cz/wps/PA_Courseware/DownloadDokumentu?id=51559>.

[7] Doc. Ing. Jiří Melichar, CSc. *LINEÁRNÍ SYSTÉMY 2* [online]. Poslední aktualizace 30. 8. 2011 [cit. 4. 5. 2015] Dostupné z URL: <https://courseware.zcu.cz/wps/PA_Courseware/DownloadDokumentu?id=51560>.

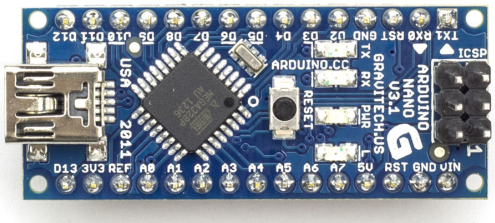
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

HTML5	HyperText Markup Language verze 5
REX	Rapid development, excelent performance
OLE	Object Linking and Embedding
OPC	OLE for Process Control
HMI	Human Machine Interface
Modbus	otevřený protokol pro vzájemnou komunikaci různých zařízení
SCADA	Supervisory Control And Data Acquisition, tedy dispečerské řízení a sběr dat

SEZNAM PŘÍLOH

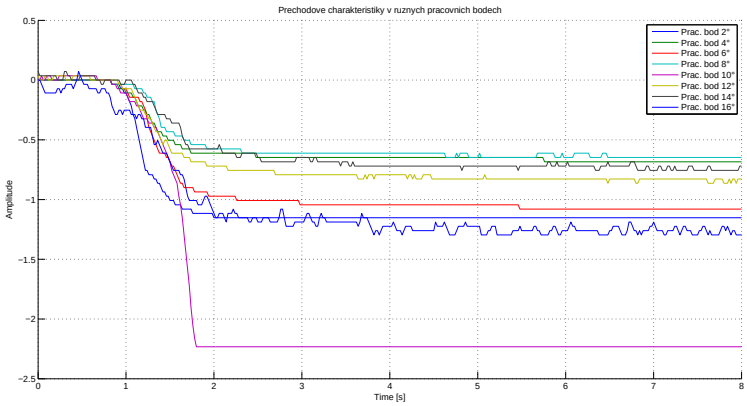
A PŘÍLOHA - OBRÁZKY

Obr. A.1: Minipočítač Raspberry Pi model B

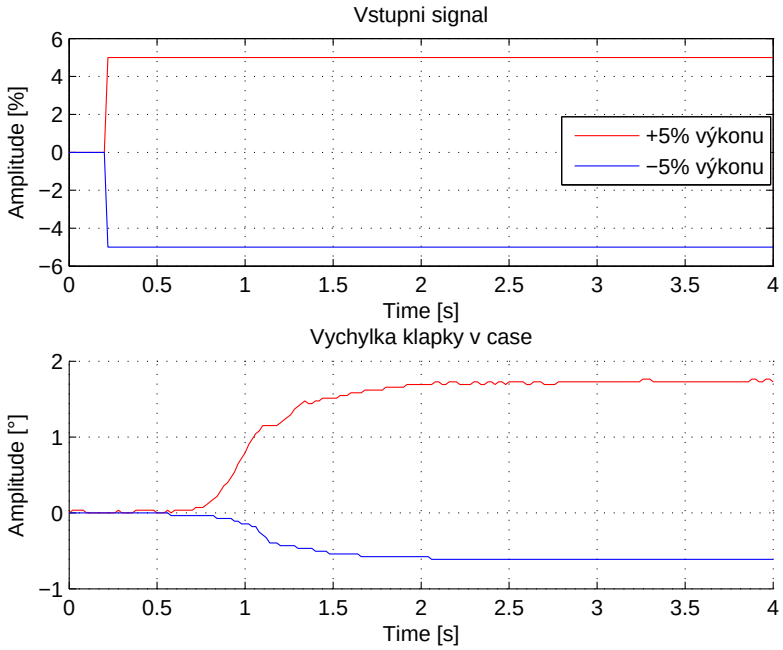


Obr. A.2: Deska Arduino NANO

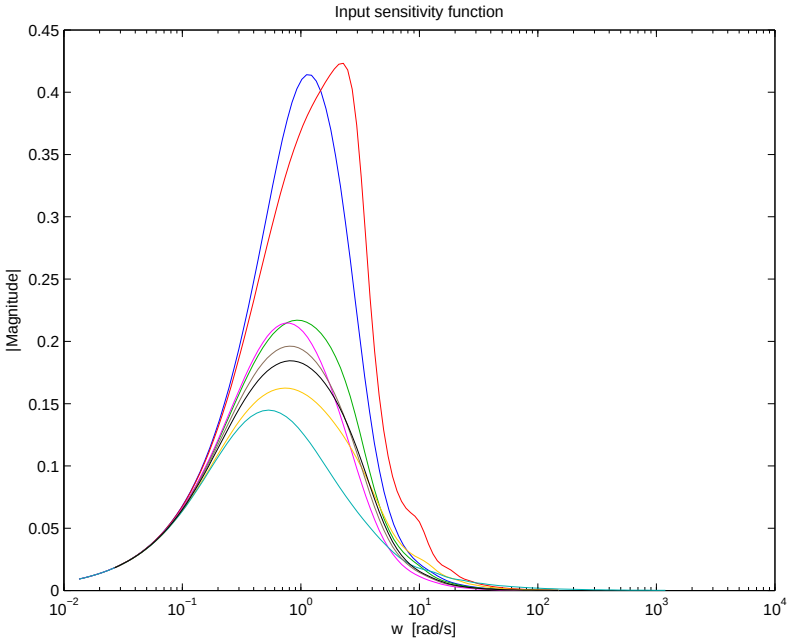
B PŘÍLOHA - GRAFY



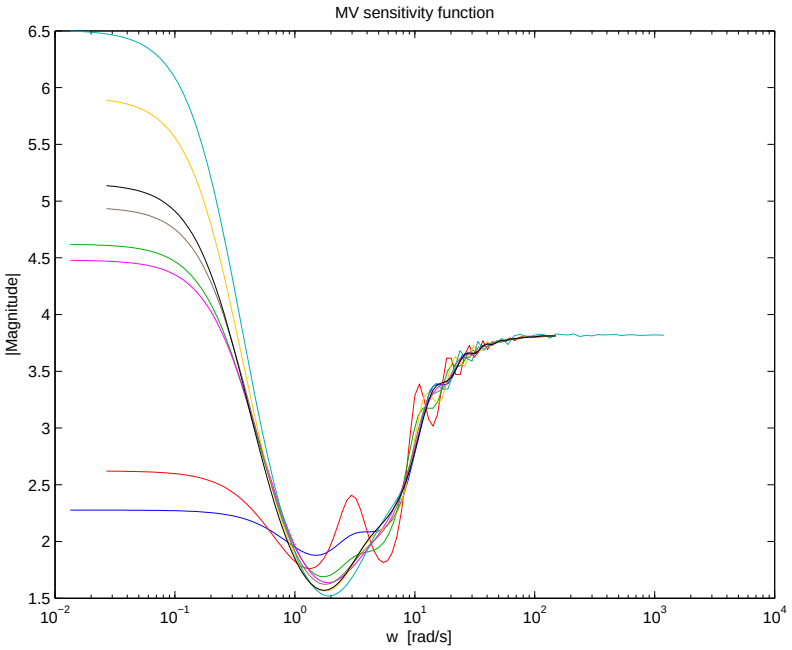
Obr. B.1: Přechodové děje v jednotlivých pracovních bodech -5%



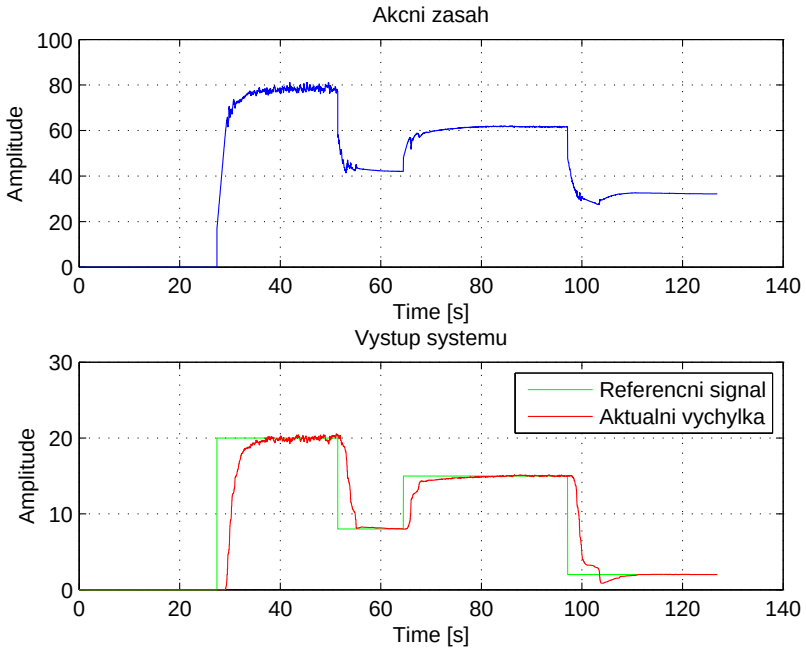
Obr. B.2: Odezva systému na skok v různých směrech



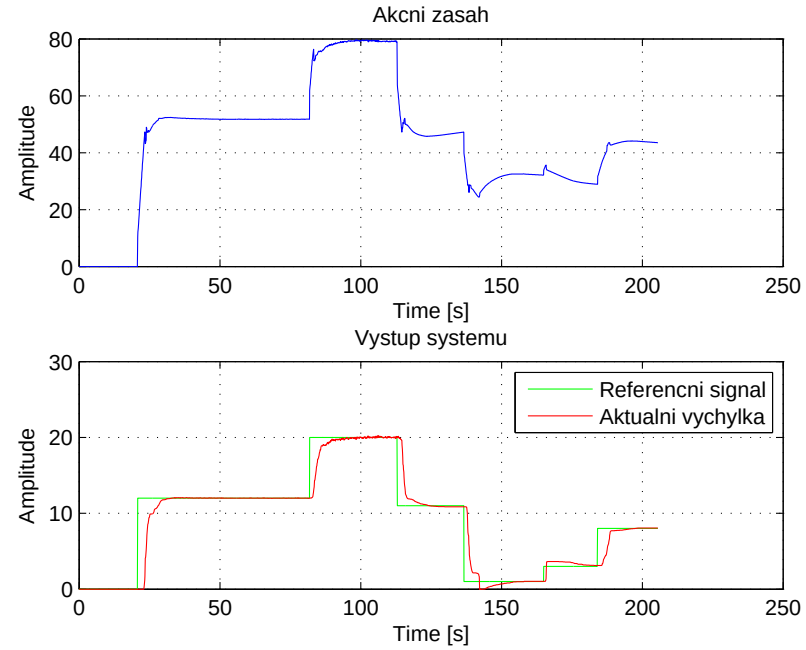
Obr. B.3: Citl. funkce vstupu regulační smyčky pro prac. bod 12°



Obr. B.4: Citl. funkce poruchy na vstupu regulační smyčky pro prac. bod 12°



Obr. B.5: Odezva systému na skokové změny vstupu - IO turck, REX



Obr. B.6: Odezva systému na skokové změny vstupu - IO turck, Resologis

C PŘÍLOHA - ODKAZY

- Oficiální webové stránky projektu Raspberry Pi
  http://www.raspberrypi.org

- Oficiální webové stránky projektu Arduino NANO
  http://arduino.cc/en/Main/arduinoBoardNano

- Aplet PIDLab
  http://www.pidlab.com/cs/

D OBSAH PŘILOŽENÉHO CD

- Dokumentace bakalářské práce ve formátu pdf
- Složka „Matlab“ obsahující naměřená data (jak ve formátu csv, tak ve formátu mat) a skripty, které jsou odzkoušeny ve verzi Matlabu 2013b.
- Složka „PIDLab“, která obsahuje data pro aplikaci PIDlab
- Složka „REXControls“ obsahující soubory pro pro řídicí systém REX a také algoritmus pro Arduino, který zajišťuje komunikaci mezi touto platformou a řídicím systémem REX
- Složka „Resologis“ obsahující data a soubory pro návrhové prostředí ISaGRAF
- Složka „Vizualizace modelu“ obsahující webovou aplikaci pro jednoduché ovládání řídicího algoritmu