ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta strojní – Ústav přístrojové a řídicí techniky

DIPLOMOVÁ PRÁCE

VYUŽITÍ MIKROPOČÍTAČE RASPBERRY PI PRO ADAPTIVNÍ IDENTIFIKACI A ŘÍZENÍ HYDRAULICKO-PNEUMATICKÉ SOUSTAVY S VYUŽITÍM JAZYKA PYTHON

2014 **Matouš Sláma**

Anotační list

Jméno autora:	Matouš Sláma
Název BP:	Využití mikropočítače Raspberry Pi pro adaptivní identifikaci a řízení hydraulicko-pneumatické soustavy s využitím jazyka Python
Rok:	2013
Obor studia:	Přístrojová a řídicí technika
Ústav/odbor:	Přístrojová a řídicí technika
Vedoucí BP:	doc. Ing. Ivo Bukovský, Ph.D.

Bibliografické údaje:

počet stran:	71
počet obrázků	36
počet tabulek	3
počet příloh	4

Klíčová slova: Raspberry Pi, RPi, gradient-descent, LNU, adaptivní neuroregulátor, HPS,

Keywords: Raspberry Pi, RPi, gradient-descent, LNU, adaptiv neuroregulator,HPS

Anotace:

Práce dokumentuje návrh, nízkonákladovou HW a SW realizaci, a testování algoritmu gradient descent pro diskrétní identifikaci a adaptivní řízení laboratorní úlohy „Hydro-pneumatická soustava“ pomocí malého počítače Raspberry Pi a desky Gertboard a opensource skriptovacího jazyka Python. Pro ověření základní funkčnosti HW realizace je uvažován regulační obvod s P-regulátorem, jehož funkci zvolený adaptivní regulátor zdokonaluje jako adaptivní stavový regulátor. Dílčím cílem práce je i vytvoření univerzální platformy s Raspberry Pi pro identifikaci a řízení laboratorních úloh podobného typu s možností snadné implementace a testování vlastních algoritmů uživatelem.

Abstract:

This work documents the design, low-cost HW and SW implementaton, and testing of the gradient descent adaptation algorithm for identification and control of the two-tank laboratory system using a small computer Raspberry Pi , Gertboard DAQ, and opensource scripting language Python. The P-control loop is implemented to validate the basic functionality of the HW implementation. Then, the adaptive P-control loop is further improved by the adaptive state-feedback-type controller. The partial objective of this thesis is also a universal platform with Raspberry Pi for simple implementation and testing custom algorithms for similar laboratory systems.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze dne

_____

Bc. Matouš Sláma

Poděkování

Děkuji vedoucímu své diplomové práce doc. Ing. Ivo Bukovskému za podporu a trpělivost při vypracovávání této práce. Byl mi oporou a vzorem pro akademický život.

Děkuji rodině za jejich trpělivost a oběti, které pokládají za mé nelineární studium.

Děkuji Míše za podporu, bez Tebe bych tohle nikdy nepsal.

Práce byla částečně podpořena grantem SGS12/177/OHK2/3T/12.

Obsah

Seznam obrázků

Seznam tabulek

Použité zkratky a symboly

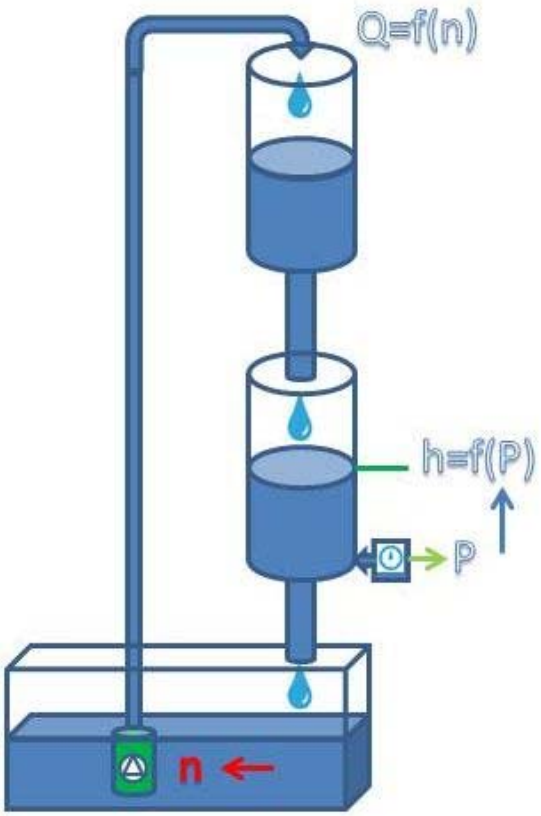
ARM	Architektura procesorů, vyvinutá firmou ARM Limited
BP	Backpropagation
DHCP	Dynamic Host Configuration Protocol
GBd	Gertboard
GD	Gradient Descent
GPIO	General Purpose Input/Output, obecný vstup/výstup
HPS	Hydro-pneumatická soustava
IP	Internet Protocol
I^2C	(I2C), Inter-Integrated Circuit
LNU	Lineární neuronová jednotka (Linear Neural Unit)
NR	Neuroregulátor
OS	Operační systém
PWM	Pulse Width Modulation, Pulzně šířková modulace
QNU	Kvadratická neuronová jednotka (Quadratic Neural Unit)
RM	Referenční model
RPi	Raspberry Pi
SPI	Serial Peripheral Interface, sériové periferní rozhraní
sudo	super user do
UART	(USART), Synchronní / asynchronní sériové rozhraní
d	žádaná veličina (desired) (okamžitá, tj. vstup do referenčního modelu)
d_{ref}	výstup referenčního modelu
e	regulační odchylka
e_{ref}	*regulační odchylka jako rozdíl výstupu referenčního modelu a regulované veličiny*
k	diskrétní index času (konstantní vzorkování)
$\mathbf{w}$	vektor adaptovatelných parametrů (vah) neuronového modelu identifikace
$\mathbf{wi}$	vektor adaptovatelných parametrů (vah) neuronového modelu regulace
$\mathbf{x}$	vektor vstupů identifikačního LNU
y_n	regulovaná veličina, (model neuronem)
y_r	regulovaná veličina měřená (reálná)
μ	koeficient rychlosti učení algoritmu GD
$\boldsymbol{\xi}$	vektor vstupů regulačního LNU

1. Úvod

Cílem této DP je navrhnout a vyzkoušet nízkonákladovou realizaci algoritmu Gradient Descent (GD) [2][3][7] s normalizovaným koeficientem rychlosti učení [8] pro diskrétní identifikaci a adaptivní řízení laboratorní úlohy Hydro-pneumatická soustava (HPS). [1] pomocí malého počítače Rasbperry Pi a desky Gertboard a opensource programovacího jazyka Python. Pro ověření základní funkčnosti HW realizace je uvažován regulační obvod s P-regulátorem, jehož funkci bude zvolený adaptivní regulátor zdokonalovat jako adaptivní stavový regulátor. Valnou částí realizace této DP je hardwarová a softwarová část, kdy je dosaženo zadaných cílů využitím požadovaných prostředků. Dílčím cílem práce je i vytvoření univerzální platformy s Raspberry Pi pro identifikaci a řízení laboratorních úloh podobného typu s možností snadné implementace a testování vlastních algoritmů uživatelem.

1.1. Cíl realizace

Řízení je realizováno na pravé části laboratorní úlohy (detail viz. Obr. 1) sestávající se ze dvou seriově zapojených nádrží kdy regulovanou veličinou je výška hladiny spodní nádrže. Jedná se tedy o soustavu s dvěma kapacitami, tedy druhého řádu. Adaptivní řízení musí postihnout výchozí stav neznámého modelu soustavy a případně dlouhodobě se měnící parametry, které neadaptivním řízením lze jen těžko postihnout, jako je reálná nelinearita osazené soustavy, rozdílná počáteční hladina zásobní nádrže, měnící se citlivost a kalibrace čidel, měnící se parametry se stárnutím aktuátoru (čerpadla) a měnící se průtok soustavou s jejím zanášením atp.



Obr. 1 Hydropneumatická soustava kde vstupem do soustavy jsou otáčky čerpadla n a měřenou veličinou je tlak P ve spodní nádobě, který je přímo úměrný regulované výšce hladiny viz vztah.

1.2. Prostředky pro realizaci

Pro řešení této úlohy je použit mikropočítač Raspberry Pi (RPi), jedná se o levné univerzální zařízení hardwarově propojující a využívající mobilní technologie, rozhraní známé z klasických PC a sběrnice, které jsme zvyklí vídat u průmyslových mikropočítačů. Toto zařízení má dnes čím dál širší komunitu uživatelů a vývojářů z řad studentů pedagogů i odborné a laické veřejnosti, která kromě nalézání nových a nových implementací, spolupracuje na odladění a usnadnění každé další. Jelikož jsou nativním operačním systémem (OS) pro RPi linuxové distribuce, je nasnadě použití programovacího jazyku Python pro řídící algoritmy. Tento jazyk je dostatečně vysoko od strojového kódu, aby bylo psaní algoritmů snadné podobně jako v Matlabu, avšak má stále dostatečně těsnou vazbu s připojovaným hardwarem a programátor si může být jist tím, co dělá. Python je zpravidla nativní součástí všech linuxových distribucí, je zde tedy nativní a jeho použití neskýtá problémy s kompatibilitou. Navíc použití přímo programovacího jazyku (místo software výrobce, nebo programového prostředí typu Matlab) neomezuje uživatele ničím jiným než vlastním časem a chutí zkoumat. Co totiž zatím naprogramované není, může brzy být a pokud to je, ale ne dost dobré, může být lepší a pokud je výpočet v pythonu příliš pomalý, vždy je cesta danou aplikaci počítat v C++ a v pythonu ji pouze pouštět (většina výpočetně náročných funkcí je takto stejně realizována).
Pro realizaci je dále použito vývojového kitu Gertboard (GBd), ze kterého je v této DP využito 10-ti bitového AD převodníku a PWM modulu pro řízení otáček čerpadla.

Obr. 2 Raspberry Pi (model B)

1.2.1. Raspberry Pi(model B)

Raspberry Pi (RPi) je jednodeskový počítač, jehož velikost je často přirovnávána k velikosti platební karty. Je dostupný ve dvou verzích dle výbavy (modela A a B). V této DP se budu dále zmiňovat o vybavenější verzi modelu B. Vyvinula jej britská nadace Raspberry Pi Foundantion s cílem podpořit výuku informatiky ve školách. Jeho základem je SoC BCM2835 firmy Broadcom, který obsahuje procesor ARM1176JZF-S s taktem 700 MHz, grafický procesor VideoCore IV a 512 megabajtů paměti RAM (model B). RPi neobsahuje žádné pevné úložiště. K zavedení systému a uložení dat slouží vyjímatelná SD karta. Mimo univerzální rozhraní, které lze k RPi připojit přes porty USB a GPIO je jistě zajímavá i cena, za kterou lze toto zařízení zakoupit. Cena je stanovena na 35 amerických dolarů (model B), přičemž dražší verze (právě model B) má navíc síťový adaptér s konektorem RJ45 a druhý USB port.

1.2.1.1. Hardware

- procesor ARM1176JZF-S z rodiny ARM11 taktovaný na 700 Mhz
- grafický procesor VideoCore IV, podporující OpenGL ES 2.0, 1080p, MPEG-4
- 512 MB RAM sdílených s grafickou kartou

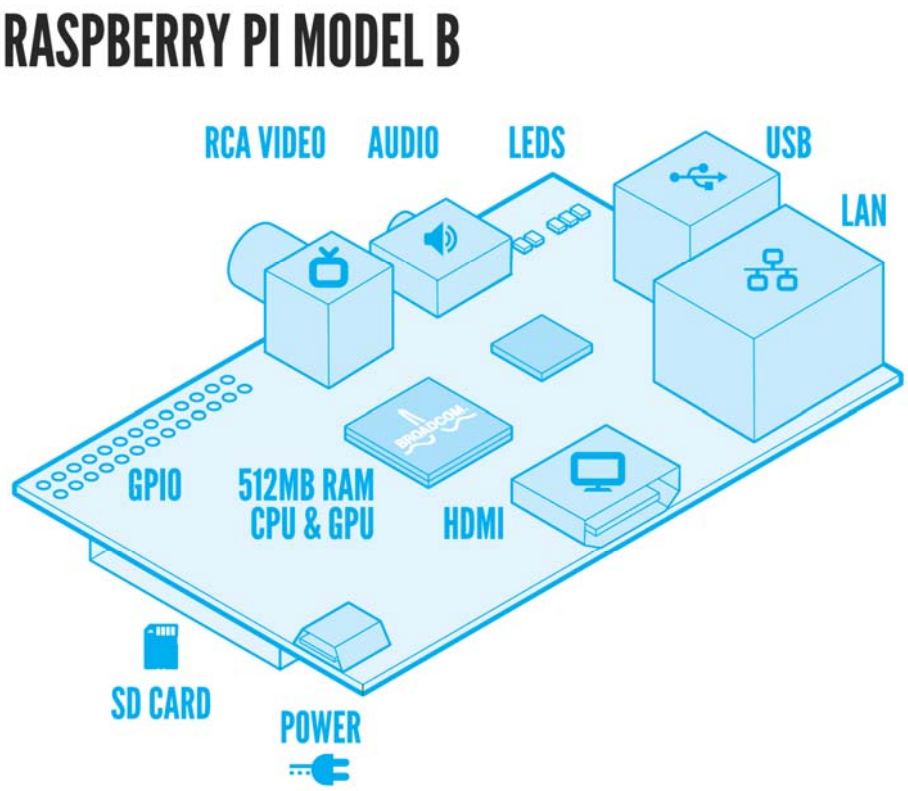
- dva USB 2.0 porty
- Obrazový výstup Composite RCA, HDMI, DSI
- Zvukový výstup přes 3,5 mm konektor, HDMI
- slot pro SD nebo MMC kartu
- ethernetový adaptér 10/100 s konektorem RJ45
- 26×GPIO, UART, I²C, sběrnici SPI

1.2.1.2. OS

Zatím všechny operační systémy pro RPi jsou linuxové distribuce upravené pro chod na ARM procesoru. Lze očekávat, že přibude OS založený na Androidu, který je dnes na ARM technologii nativní.

Ke dnešku jsou oficiálně k dispozici tyto distribuce:

- Raspbian - ARM verze Debianu , přívětivé grafické prostředí i rychlost
- Pidora - ARM verze Fedory pro uživatele zvyklé na toto prostředí
- RISC OS - Lehký nenáročný systém
- RaspBMC - ARM verze XBMC, multimediální systém
- Arch - ARM verze ARCH linuxu, jednoduchá, uživatelsky náročnější
- OpenELEC - ARM verze XBMC, jednoduchý multimediální systém



Obr. 3 Popis a umístění veškerého rozhraní, které lze na RPi využít

1.2.1.3. Raspberry Pi Foundantion (autoři - oficiální text, individuální překlad):

Nápad na malý a levný počítač pro děti/studenty, přišel v roce 2006, kdy Eben Upton, Rob Mullins , Jack Lang a Alan Mycroft, zaměstnaní na University of Cambridge's Computer Laboratory, byli znepokojeni poklesem dovedností studentů na pro porozumění informatice. Ze situace v roce 1990, kdy většina dětí přicházeli na pohovor jako zkušený fanda programátor, tak po roce 2000 byla situace velmi odlišná, typický žadatel umí jen vytvořit jednoduchý web design .

Něco se změnilo způsob, jakým děti komunikují s počítačem. Byla zjištěna řada problémů : kolonizaci osnov informačních a komunikačních technologií s lekcí o používání aplikace Word a Excel , nebo psaní webových stránek, konec dot-com boomu (internetová horečka) a vzestup domácích PC a herních konzolí , které nahradily Amigy , BBC Micros , Spectrum ZX a Commodore 64 na kterých se starší generace naučila programovat.

Není příliš velká skupina lidí, která může řešit problémy, jako je nedostatečný vzdělávací program. Ale cítili jsme, že bychom se mohli pokusit udělat něco se situací, kdy se počítače staly tak drahé a tajemné, že pokusy o programování na nich muselo být zakázáno rodiči a najít platformu, která stejně jako u starých domácích počítačů, by mohla zavést do programovací prostředí. Od roku 2006 do roku 2008 jsme navrhli několik verzí toho, čím se nyní stalo Raspberry Pi.

Před rokem 2008, procesory určené pro mobilní zařízení jsou stále více cenově dostupné a dostatečně silné, aby poskytli vynikající multimediální funkce. Snažili jsme se, aby deska byla žádaná, aby děti, které by měly zájem o zařízení orientované čistě na programování měly takové zařízení k dispozici. Projekt začal vypadat velmi realizovatelně. Eben (nyní chip architect v Broadcom), Rob, Jack a Alan, se spojil s Pete Lomasem , MD of hardware design and manufacture company Norcott Technologies a Davidem Brabenem , spoluautor seminal BBC Micro game Elite, tvoří Raspberry Pi Foundation. O tři roky později nadace zadala hromadnou výrobu prostřednictvím licencované společnosti element 14/Premier Farnell and RS Electronics a během roku se prodalo přes jeden milion kusů .

Měli jsme obrovský zájem, podporu a pomoc ze vzdělávací komunity, za kterou jsme rádi a trochu pokořeni počtem dotazů od agentur a lidí směřujících daleko od našich původních cílů pro toto zařízení. Rozvojové země mají zájem o Raspberry Pi jako zařízení v oblastech kde si nemohou dovolit takový odběr energie pro hardware potřebný ke spuštění tradičního stolního počítače. Nemocnice a muzea nás kontaktují, aby zjistili jak pomocí Raspberry Pi řídit zobrazovací zařízení. Rodiče těžce postižených děti, s námi projednávají o monitoringu a aplikace pro usnadnění. A zdá se, že venku je milion lidí, s rozpálenými pájkami, kteří chtějí stavět roboty.

Nechceme tvrdit, že máme všechny odpovědi. Nemyslíme si , že Raspberry Pi je záchrana všech problémů v světových počítačových otázkách , jsme ale přesvědčeni, že můžeme být katalyzátorem . Chceme vidět levné , dostupné , programovatelné počítače všude, jsme aktivní v podpoře dalších společností . Chceme zlomit paradigma, ve kterém aniž by utráceli stovky liber za PC , nemohou rodiny používat internet. Chceme aby vlastnictví osobního počítače bylo skutečně normální pro děti, a těšíme se na to, co nám budoucnost přinese.

The Raspberry Pi Foundation is a UK registered charity (Registration Number 1129409)

1.2.1.4. Pěkné projekty na RPi u nás a ve světě:

Robotická ruka ovládaná slovními příkazy http://aonsquared.co.uk/

Bezdrátový přenos videa/audia z mobilu (Android) do TV http://www.raspberrypi.org/archives/1512

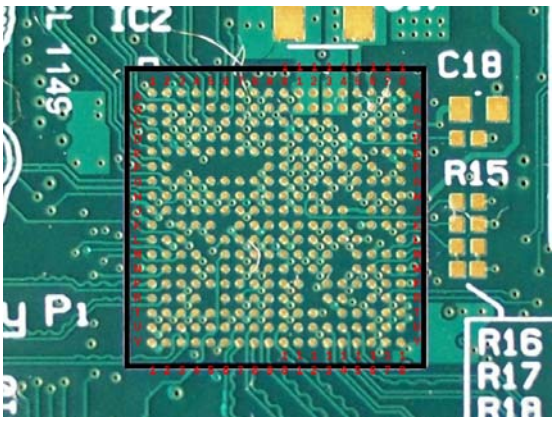
Větší množství menších projektů s čidly a aktuátory pro pozorování hvězd http://www.astromik.org/raspi/

Pin procesoru jako FM transmitter
http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter

Nebo například v Technické ve 3. patře, ve vitrýně velmi pěkná realizace Ing. Vladimíra Hlaváče a Ing. Cyrila Oswalda, ovládání robota RPi přes http. A mnoho dalších.

1.2.2. General Purpose Input/Output (GPIO)

Obecný vstup / výstup (GPIO) je obecný pin na integrovaném obvodu (obyčejně je jím čip), jehož chování (včetně toho, zda se jedná o vstupní nebo výstupní pin) může být řízen (naprogramován), uživatelem za běhu procesoru.
GPIO piny nemají žádnou zvláštní definici, a ve výchozím stavu jsou nevyužité. Myšlenka těchto pinů tkví v tom, že může být vhodné, při návrhu integrovaného obvodu přidat i několik dalších digitálních řídicích linek. Jsou pak k dispozici a lze tím předejít potížím s případným přidáváním obvodů. Například, čipy Realtek ALC260 (audio kodeky) mají 8 GPIO pinů, které jsou nevyužité ve výchozím nastavení. Některé systémové integrátory (Acer notebooky), které používají chipset ALC260 mají využit první GPIO (GPIO0) pro zapnutí zesilovače používaného pro interní reproduktory notebooku a externího konektor pro sluchátka.
Kromě vstupů/výstupů které má i standardní PC, jako je USB, ethernet, RCA video, HDMI a audio jack, je RPi vybaveno i nízkoúrovňovými GPIO. Nízkoúrovňovými je myšlen způsob zpracování signálu. Jde o piny připojené víceméně přímo na ARM procesor a zpracování dat pak probíhá na nejnižší možné úrovni. Toto vyžaduje i jistou disciplínu v zacházení s připojením k GPIO, procesor běží na napětí 3,3V, nesmíme tedy na jeho piny přivést 5V! V lepším případě dojde k vypnutí RPi v horším případě k odpálení procesoru.GPIO porty na RPi nejsou tolerantní k připojení 5V!



Obr. 4 pole pro připojení GPIO pinů procesoru RPi

Procesor RPi má v aktuální revizi 53 GPIO pinů (ze všech pinů procesoru na obr.), z toho 17 je vyvedeno do 26 pinového konektoru P1 zobrazeném naObr. 5. Zbytek pinů konektoru je využit pro napájení a uzemnění. Ze známých sběrnic na konektoru P1 nalezneme UART, SPI a I^2C

Obr. 5 Popis GPIO konektoru P1 RPi

Legenda barev
+5 V
+3.3 V
Ground, 0V
UART
GPIO
SPI
I^2C

Tab. 1 Barevné rozlišení pro přehlednost Tab. 2, Tab. 3 a Obr. 5

1.2.2.1. P1: Popis funkcí GPIO horní řada pinů:

Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-02	5V0		Supply through input poly fuse		
P1-04	5V0		Supply through input poly fuse		
P1-06	GND				
P1-08	GPIO 14		Boot to Alt 0 ->	UART0_TXD	ALT5 = UART1_TXD
P1-10	GPIO 15		Boot to Alt 0 ->	UART0_RXD	ALT5 = UART1_RXD
P1-12	GPIO 18			PCM_CLK	ALT4 = SPI1_CE0_N ALT5 = PWM0
P1-14	GND				
P1-16	GPIO23				ALT3 = SD1_CMD ALT4 = ARM_RTCK
P1-18	GPIO24				ALT3 = SD1_DAT0 ALT4 = ARM_TDO
P1-20	GND				
P1-22	GPIO25				ALT3 = SD1_DAT1 ALT4 = ARM_TCK
P1-24	GPIO08			SPI0_CE0_N	
P1-26	GPIO07			SPI0_CE1_N	

Tab. 2 Konektor P1 popis pinů horní řada

1.2.2.2. P1: Popis funkcí GPIO spodní řada pinů:

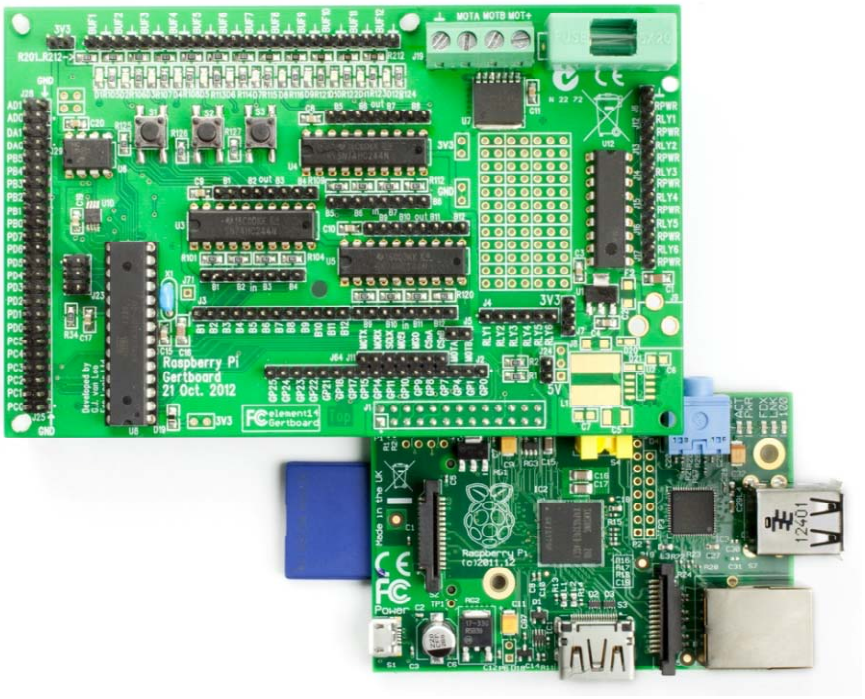
Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-01	3.3 V		50 mA max (01 & 17)		
P1-03	GPIO 0	**GPIO 2**	1K8 pull up resistor	I2C0_SDA / **I2C1_SDA**	
P1-05	GPIO 1	**GPIO 3**	1K8 pull up resistor	I2C0_SCL /	

			I2C1_SCL	
P1-07	GPIO 4		GPCLK0	ALT5 = ARM_TDI
P1-09	GND			
P1-11	GPIO17			ALT3 = UART0_RTS ALT4 = SPI1_CE1_N ALT5 = UART1_RTS
P1-13	GPIO21 **GPIO27**		PCM_DOUT / **reserved**	ALT4 = SPI1_SCLK ALT5 = GPCLK1 / **ALT3 = SD1_DAT3 ALT4 = ARM_TMS**
P1-15	GPIO22			ALT3 = SD1_CLK ALT4 = ARM_TRST
P1-17	3.3 V	50 mA max (01 & 17)		
P1-19	GPIO10		SPI0_MOSI	
P1-21	GPIO9		SPI0_MISO	
P1-23	GPIO11		SPI0_SCLK	
P1-25	GND			

Tab. 3 Konektor P1 popis pinů spodní řada

1.2.3. Gertboard

Gertboard (GBd) je vstupně-výstupní (I / O), rozšiřující deska pro počítač RPi. GBd není oficiálním produktem nadace Raspberry Pi Foundantion, ale je vyvinut Gertem Van Loo (proto název desky Gertboard), který se podílel i na vývoji alfa verze RPi.

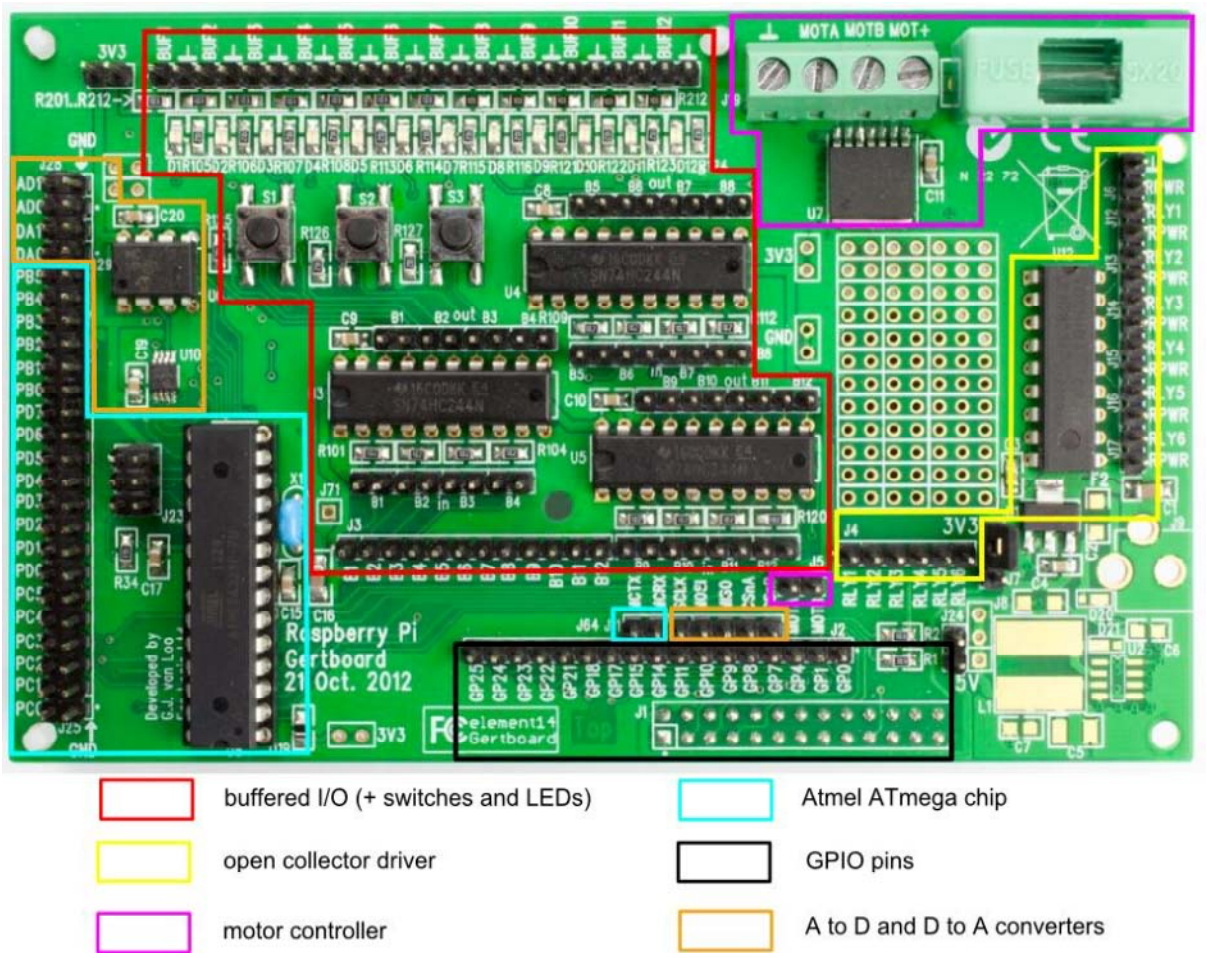


Obr. 6 GBd napojen přímo na RPi (při použití samičího konektoru na GBd)

Na rozšiřující desce nalezneme:

- připojeno přes SPI:
    - MCP4802 - 8-bitový 2-kanálový D/A převodník
    - MCP3002 - 10-bitový 2-kanálový A/D převodník
- předprogramovaný Atmel AVR ATmega 328P, 28-pin dual in-line, připojený přes UART
- obvod pro řízení motoru L2603-MW, 18V, 2A připojený k PWM
- 3 tlačítka
- 12 IO pinů připojeno přes obvod 74xx244, pro signalizaci je použito 12 LED diod
- 6 výstupů s otevřeným kolektorem 50V, 0.5A (ULN2803A)

GBd se připojuje k RPi na piny GPIO přes 26-žilový plochý kabel, nebo při použití vhodného konektoru lze napojit přímo jako na Obr. 6. Původně byl dodáván jako stavebnice, avšak momentálně byla většina součástek minimalizována na SMD a deska se dodává již smontovaná a otestovaná. Její cena je zřejmě z důvodu výroby v menším nákladu než RPi vyšší (než RPi) dnes 50 dolarů, avšak stále se nejedná o likvidační částku.
GBd je napájen prostřednictvím GPIO pinů na RPi, potřebujeme tedy napájení pro RPi, který je schopno dodávat proud alespoň 1A lépe však např. 2A.
GBd se sestává ze sbírky funkčních bloků, které mohou být spojeny a kombinovány mnoha způsoby
Umístění těchto bloků na Gertboard je znázorněno na Obr. 7



Obr. 7 Umístění jednotlivých bloků na desce GBd

1.2.4. Python

Python je interpretovaný, dynamický, objektově orientovaný programovací jazyk, který je určen pro efektivní vývoj aplikací. Často je označován za jazyk spojující výhody programování v jazyce C++ a Java. Obecně lze tento jazyk považovat za vhodný pro začátečníky. Díky jednoduché syntaxi je snadné se s ním naučit tvořit primitivní programy, avšak díky rozsáhlé databázi modulů je možné s ním vytvářet i rozsáhlé a komplikované aplikace. Jazyk python je vyšší než jazyk C++, avšak většina výpočetních funkcí je v modulech python přesto realizováno právě v jazyce C++, což umožňuje využití vyššího výpočetního výkonu. Pro použití jazyka python tak jak jsme zvyklí používat prostředí Matlab je potřeba minimálně doinstalovat tyto moduly:

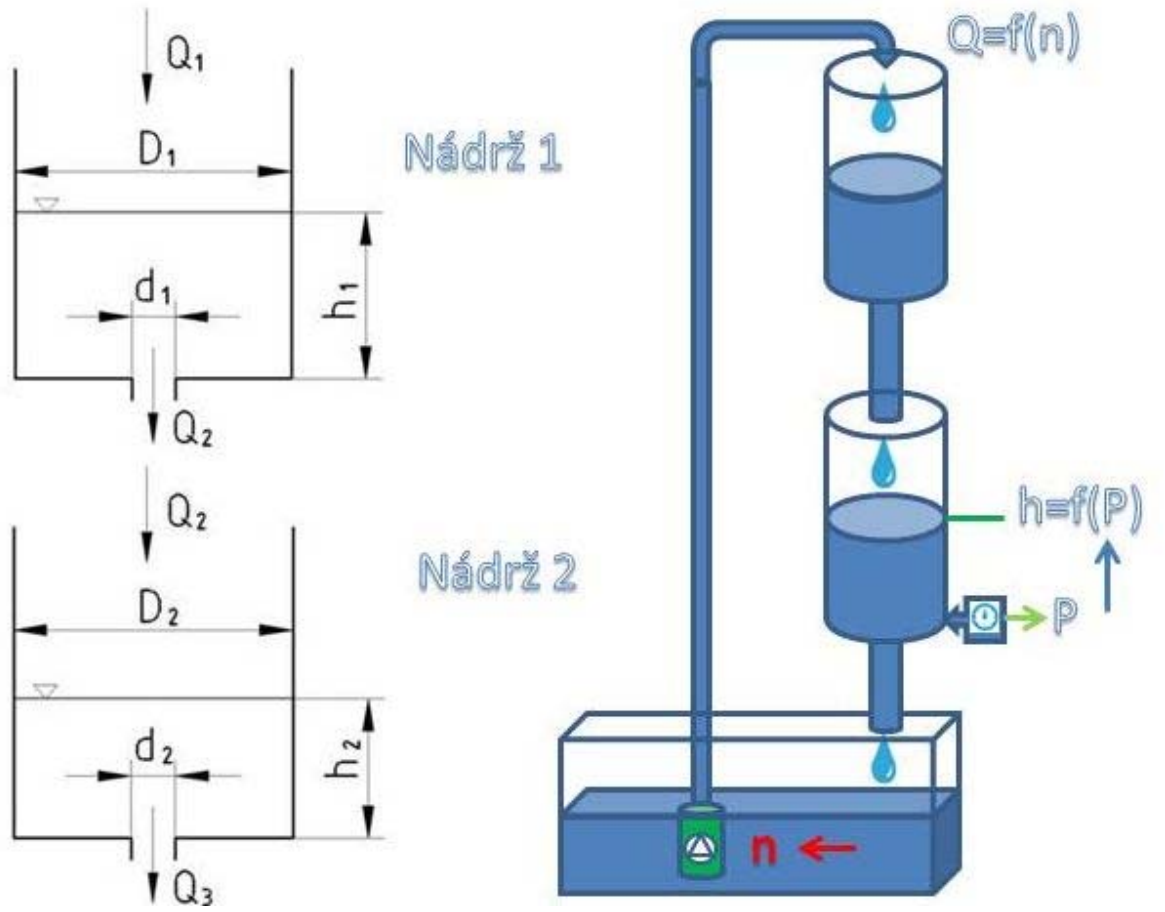
- Numpy (Numerical python)pro realizaci rychlých matematických operací,
- matplotlib pro vykreslování výsledků do grafů
- scipy (Scientific python), některé další matematické funkce které neobsahuje numpy
- spidev, pro komunikaci přes SPI sběrnici
- wiringpi, pro emulaci a nastavení motorkontroléru

1.3. Hydropneumatická soustava

1.3.1. Fyzikální model

Fyzikální model jsem vypracoval v rámci předmětu IDS [11]proto část z něj v této DP pouze cituji.

1.3.1.1. Schéma a značení



Obr. 8 schéma úlohy – dvě nádrže v sérii

Přehled značení:

Q_1, Q_2, Q_3 - objemový tok [m^3/s]
V_1, V_2 - objem nádrže 1 a 2 [m^3]
D_1, D_2 - průměr nádrže [m]
d_1, d_2 - průměr výpusti [m]
h_1, h_2 - výška hladiny [m]
h_{10}, h_{20} - počáteční hodnoty výšky hladiny [m]
S_1, S_3 - průřez nádrže 1 a 2 [m^2]
S_2, S_4 - průřez výpustí u nádrže 1 a 2 [m^2]
v_1, v_2 - rychlost výtoku vody z nádrže 1 a 2 [m/s]
μ_1, μ_2 - výtokový součinitel pro výpusť nádrže 1 a 2 [1]
g - gravitační zrychlení [m/s^{-2}]
k - opravný koeficient pro akční veličinu [1]
u - akční veličina

1.3.1.2. Pomocné výpočty:

Přibližně změřeno: $D_1 = D_2 = 55mm$ a $d_1 = d_2 = 4mm$

$$\Rightarrow S_1 = S_3 = \frac{\pi D_1^2}{4} = \frac{\pi \cdot 0{,}055^2}{4} = 2{,}38 \cdot 10^{-3}\, m^2 \quad (1)$$

$$\Rightarrow S_2 = S_4 = \frac{\pi d_1^2}{4} = \frac{\pi \cdot 0{,}004^2}{4} = 1{,}26 \cdot 10^{-5}\, m^2 \quad (2)$$

1.3.1.3. Matematicko-fyzikální model

Pro změnu objemu v nádrži 1 platí:

$$\frac{dV_1}{dt} = Q_1 - Q_2 \quad (3)$$

Pro změnu objemu v nádrži 2 platí:

$$\frac{dV_2}{dt} = Q_2 - Q_3 \quad (4)$$

Z toho Q_1 je funkcí akční veličiny a lze ji zjednodušeně vyjádřit jako:

$$Q_1 = k \cdot u \quad (5)$$

Velikost objemového toku Q_2 se mění dle Torricelliho vztahu:

$$Q_2 = S_2 \mu_1 \sqrt{2gh_1} \quad (6)$$

A stejně se mění objemový tok Q_3:

$$Q_3 = S_4 \mu_2 \sqrt{2gh_2} \quad (7)$$

Změnu objemu V_1 a V_2 lze rozepsat takto:

$$\frac{dV}{dt} = S\frac{dh}{dt} \quad (8)$$

Dosazením rovnic 3 až 6 do rovnic 1 a 2 dostávám:

$$S_1 \frac{dh_1}{dt} = k \cdot u - S_2 \mu_1 \sqrt{2gh_1} \tag{9}$$

$$S_3 \frac{dh_2}{dt} = S_2 \mu_1 \sqrt{2gh_1} - S_4 \mu_2 \sqrt{2gh_2} \tag{10}$$

Úpravou získám stavový popis soustavy:

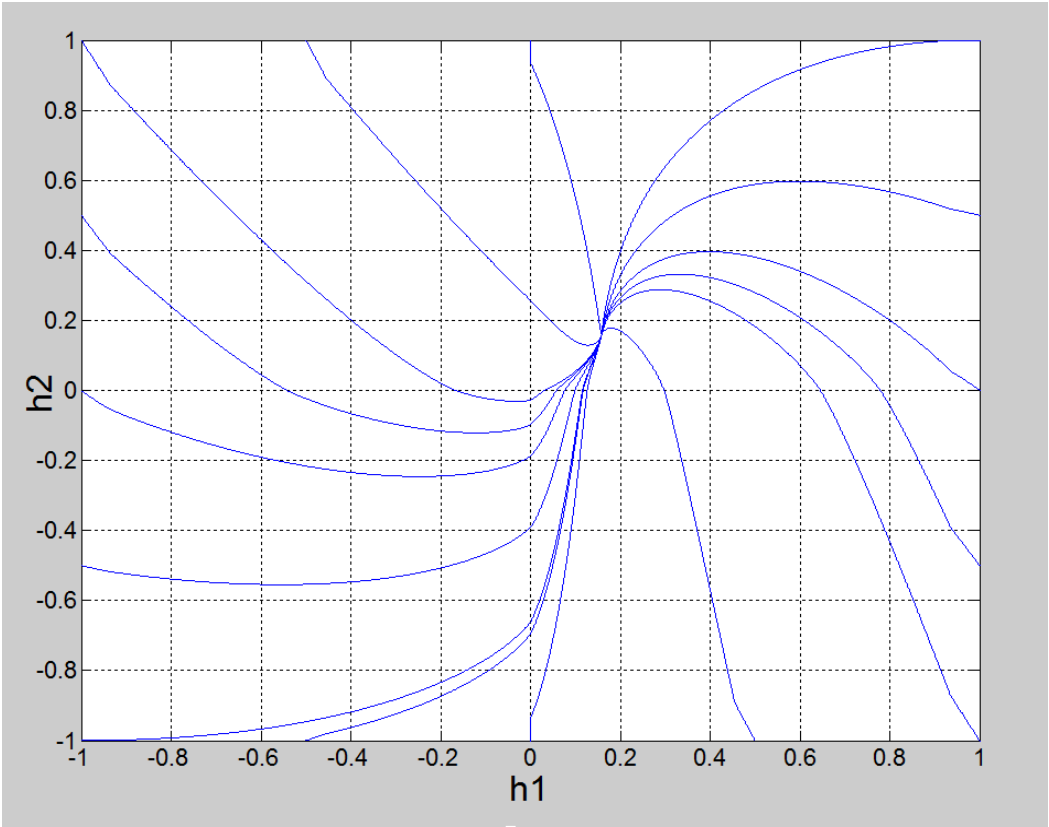
$$\frac{dh_1}{dt} = \frac{k \cdot u - S_2 \mu_1 \sqrt{2gh_1}}{S_1} \tag{11}$$

$$\frac{dh_2}{dt} = \frac{S_2 \mu_1 \sqrt{2gh_1} - S_4 \mu_2 \sqrt{2gh_2}}{S_3} \tag{12}$$

Pokud budu uvažovat zjednodušení $S_1 = S_3$, $S_2 = S_4$, $\mu_1 = \mu_2$ zjednoduší se soustava na:

$$\frac{dh_1}{dt} = \frac{k \cdot u - S_2 \mu_1 \sqrt{2gh_1}}{S_1} \tag{13}$$

$$\frac{dh_2}{dt} = \frac{S_2 \mu_1 \sqrt{2gh_1} - S_2 \mu_1 \sqrt{2gh_2}}{S_1} = \frac{S_2 \mu_1 \sqrt{2g}}{S_1}\left(\sqrt{h_1} - \sqrt{h_2}\right) \tag{14}$$



Obr. 9 Stavový diagram pro jedno nastavení u a pro různé počáteční podmínky h_1 a h_2 [m]

Hodnotu konstanty μ_1 jsem stanovil na 0,8 (v rámci předmětu IDS [11]), což by mělo odpovídat otvoru v nádobě s ostrými hranami. Jak je vidět ze stavového diagramu (Obr. 9) nelineární soustava má ideálně jedno stabilní řešení typu stok.

1.3.1.4. Lineární model

Pokud chci linearizovat model, stačí linearizovat členy $\sqrt{h_1}$ a $\sqrt{h_2}$. Linearizovaný tvar získám z prvních dvou členů Taylorova rozvoje pro pracovní bod h_{10} a h_{20}:

$$\sqrt{h_1} \approx \sqrt{h_{10}} + \frac{1}{2\sqrt{h_{10}}}\left(h_1 - h_{10}\right) \tag{15}$$

$$\sqrt{h_2} \approx \sqrt{h_{20}} + \frac{1}{2\sqrt{h_{20}}}\left(h_2 - h_{20}\right) \tag{16}$$

Nahrazením $\sqrt{h_1}$ a $\sqrt{h_2}$ v rovnicích [13] a [14] dostaneme lineární soustavu:

$$\frac{dh_1}{dt} = \frac{k \cdot u}{S_1} - \frac{S_2 \mu_1 \sqrt{2g}}{S_1}\left(\sqrt{h_{10}} + \frac{1}{2\sqrt{h_{10}}}\left(h_1 - h_{10}\right)\right) \tag{17}$$

$$\frac{dh_2}{dt} = \frac{S_2 \mu_1 \sqrt{2g}}{S_1}\left[\left(\sqrt{h_{10}} + \frac{1}{2\sqrt{h_{10}}}\left(h_1 - h_{10}\right)\right) - \left(\sqrt{h_{20}} + \frac{1}{2\sqrt{h_{20}}}\left(h_2 - h_{20}\right)\right)\right] \tag{18}$$

1.3.1.5. Lineární matematický model

Protože matematicko-fyzikální analýzou soustavy jsme se dopracovali k soustavě druhého řádu, budeme uvažovat hledaný lineární diferenciální model také druhého řádu ve tvaru:

$$G(S) = \frac{S}{(\tau_1 s + 1)(\tau_2 s + 1)} \qquad \text{Vztah 19}$$

Pro získání hodnot τ_1 a τ_2 potřebujeme znát hodnoty $t_{0,6}$ a $t_{0,2}$ z přechodové charakteristiky. Na reálném modelu lze změřit odezvu soustavy na různé skokové změny akční veličiny a převést je na přechodovou charakteristiku. Jelikož jsem toto řešil v rámci semestrální práce předmětu IDS [11] nebudu se zde dále citovat.

2. Instalace OS a prostředků pro chod NR na RPi

Z v úvodu zmiňovaných OS jsem pro práci NR zvolil Raspbian, jedná se o ARM linuxovou distribuci debianu. Toto prostředí má příjemné grafické prostředí ale lze jej spouštět i pouze v shellu. Samozřejmostí systému je automatický DHCP client, který po připojení RPi k ethernet síti s automatickým přiřazením IP adresy, připojí RPi k síti. Ve výchozí konfiguraci je povoleno i SSH připojení čímž se zařízení stává okamžitě použitelné i bez monitoru klávesnice atd. Stačí zařízení pouze napájet a připojit jej k ethernet síti.

2.1. Instalace OS a první spuštění

Distribuce Raspbianu je volně ke stažení odkazem ze stránek RPI:

http://www.raspberrypi.org/downloads

Zde je možné stáhnout obraz OS tento je možné s použitím software (např. doporučeným na http://elinux.org/RPi_Easy_SD_Card_Setup) nakopírovat na SD kartu. Ta by měla mít velikost alespoň 4GB a vzhledem k jejímu využití jako systémového disku je vhodné třída 10 a vyšší. Ve Windows je možné použít program Win32DiskImager, v linuxových distribucích to může být dd.

RPi je možné spustit přímo jako PC s monitorem, klávesnicí a myší, nebo jej pouze připojit k síti a zapnout. Zapínání a vypínání RPi se provádí připojením/odpojením napájení do microUSB konektoru. Ačkoliv výrobce uvádí, že lze takto zařízení bez obav vypínat, je lepší nejdříve ukončit veškeré procesy, což se provádí v terminálu (v GUI je vypínací tlačítko) příkazem

```
sudo halt
```

Při připojení RPi přes DHCP server je pouze nutné zjistit jeho IP adresu. Pak se k němu lze připojit přes SSH jako uživatel pi s heslem raspberry, toto je pouze defaultní, je možné vytvářet a mazat nové uživatele a samozřejmě jim i měnit hesla. V linuxu se připojení k RPi provádí např. v terminálu a příkazem:

```
ssh pi@192.168.x.x
```

kde 192.168.x.x obsahuje skutečnou IP adresu a po povolení komunikace zadám heslo raspberry. Ve Windows lze použít např. jednoduchou aplikaci PuTTY, kde jako hostname uvedeme opět pi@192.168.x.x

První spuštění Raspbianu umožní rozšířit alokovanou paměť SD karty na její skutečnou velikost. Je to dáno mechanismem, kdy je OS dodáván jako obraz, proto se na SD kartě vytvoří jako 2GB prostor. Po spuštění je ale možné toto změnit a využít celý prostor na kartě. Déle je možné si vybrat, zda při startu RPi dojde automaticky ke spuštění grafického prostředí. Toto jde vždy spustit z konzole příkazem

```
startx
```

V grafickém prostředí se příkazy zadávají v terminálu (konzoli), V shellu je lze zadávat přímo.

2.2. Instalace modulů a součástí pro Python na RPi

Vždy před instalací dalších součástí je dobré zkontrolovat aktualizace k těm stávajícím. To se provede následující dvojicí příkazů:

sudo apt-get update
sudo apt-get upgrade

Interpreter Pythonu je nyní součástí raspbianu ale neuškodí zkontrolovat, zda je skutečně nainstalován se všemi potřebnými součástmi. Společně s tím lze nainstalovat i několik dalších šikovných součástí pro Python:

sudo apt-get install python
sudo apt-get install python-dev
sudo apt-get install libjpeg-dev
sudo apt-get install libfreetype6-dev
sudo apt-get install python-setuptools
sudo apt-get install python-imaging python-imaging-tk
sudo apt-get install python-pip

Poslední instalovaná součást (pip) je velmi šikovná, jedná se o instalátor modulů do interpreteru pythonu. Odpadá pak mnohdy pro začátečníka neprůhledná a odrazující instalace. Součásti pomocí pip se nainstalují takto:

sudo pip install RPi.GPIO
sudo pip install pySerial
sudo pip install nose
sudo pip install cmd2
sudo pip install wiringpi
sudo pip install spidev

Moduly, s kterými je python výrazně přátelštější v matematických výpočtech a zobrazení výsledků podobně jako jsme zvyklí v Matlabu nainstalujeme takto:

sudo apt-get install python-matplotlib
sudo apt-get install python-mpltoolkits.basemap
sudo apt-get install python-numpy
sudo apt-get install python-scipy
sudo apt-get install python-pandas

Na modulu spidev bych rád ukázal alternativní způsoby instalace, jelikož s jeho instalací přes pip jsem měl v předchozí verzi potíže. Jedná se o modul, který zprostředkovává komunikaci s AD/DA převodníky a jiným dalším rozhraním komunikujícím přes SPI sběrnici. Instaluje se například z gitu, tento je nejdříve třeba nainstalovat:

sudo apt-get install git

A pak z něho nainstalovat spidev:

git clone git://github.com/doceme/py-spidev
cd py-spidev/
sudo python setup.py install

Druhý způsob jak spidev nainstalovat je stáhnout instalátory a spustit:

mkdir python-spi
cd python-spi
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install

Tady bych rád poznamenal, že SPI musí být nejen nainstalováno, ale je třeba také kontrolovat zda není na „blacklistu“. Bohužel takovou změnu dokáží i některé aktualizace. Změna tohoto se provádí zakomentováním v souboru dostupném z konzole takto:

sudo nano /etc/modprobe.d/raspi-blacklist.conf

Kde musí být mřížka před komunikací, kterou chceme mít povolenu:

#blacklist spi-bcm2708

Pokud nejste připojeni přes ssh, ale pracujete přímo na RPi, lze soubor upravit i například v geditu atp.

Ze všech součástí v předešlých je pro základní chod NR na HPS nutné mýt nainstalovány tyto:

wiringpi
spidev
numpy
scipy
matplotlib

Ostatní jsou hlavně pro další práci s GPIO porty a knihovny, jež nezaberou příliš místa a pro programování v Pythonu na RPi jsou velmi účelné ale pro tuto DP ne nezbytné.

2.3. spouštění Python skriptů na RPi

Skripty napsané v jazyce Python jsou zpravidla textové soubory s koncovkou .py. Jako takové se jak v linuxových distribucích, tak v příkazové řádce Windows spouštějí následujícím příkazem, který pomocí interpreteru Python přeloží zvolený skript a spustí:

python /cesta k souboru pokud nejsem přímo v adresáři/můj_skript.py

Pokud jsem tedy v adresáři kde se skript nachází, píšu pouze:

python můj_skript.py

U Python skriptů spouštěných na RPi, které mají přístup ke komunikaci například přes SPI sběrnici je třeba si uvědomit, že jde o operace vyžadující speciální oprávnění. Proto musí být takovýto skript spouštěn jako super user:

sudo python můj_skript_který_pracuje_s_SPI.py

Z toho vyplívá i omezení pro IDLE (Python GUI), v kterém lze skripty spouštět a ladit. Tento je třeba také spouštět jako sudo:

sudo idle.py

Nebo lze vytvořit spouštěcí skript jako textový soubor s příkazem a koncovkou sh (podobně jako ve windows bat)

Za zmínku stojí možnost použití interaktivního režimu při spouštění intepreteru Pythonu. Jedná se o vlastnost dobře známou z Matlabu, nebo z některých vývojových prostředí pro programování v pythonu (např. idlex), po té co skript dokončí úlohu nedojde k ukončení programu, ale je možné stále pracovat s vytvořenými proměnnými, vykreslovat je, ukládat je nebo program ručně ukončit. Interpreter Pythonu se pak musí pouštět s příznakem (parametrem) -i např:

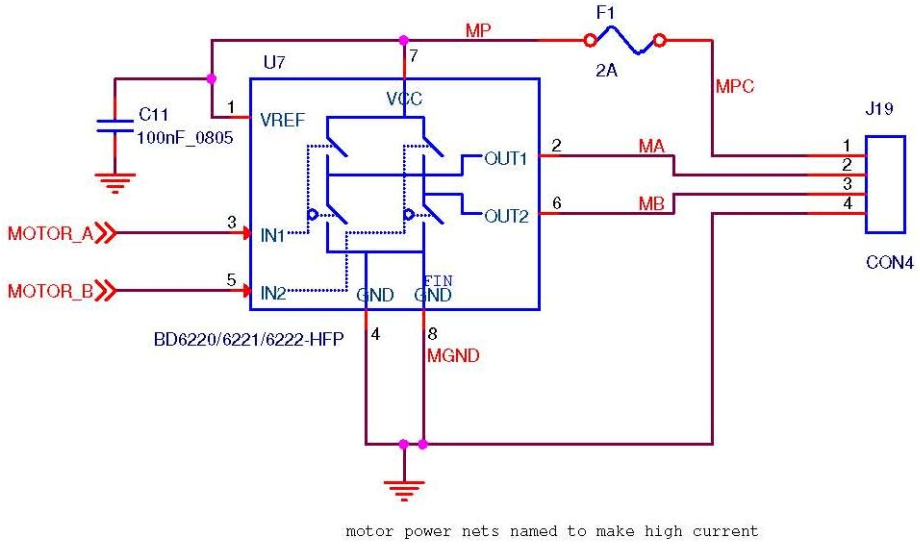
sudo python -i můj_skript_který_chci_jako_su_spustit_v_interaktivním_režimu.py

3. Zapojení GBd s RPi k HPS

Pro řízení HPS budou z GBd využity dva obvody. Jsou to: BD6222 a MCP3002 první zmiňovaný je obvod pro PWM řízení elektromotoru. Druhý obvod je AD převodník, který poslouží k získání hodnoty výšky hladiny ve spodní nádobě.

3.1. BD6222 H-bridge driver

Jedná se o spínaný H-můstek pro ovládání kartáčových motorů. Každý integrovaný obvod může pracovat v širokém rozsahu výkonu. Přímo tento obvod může být napájen až 18V a umožňuje výstupní proudy až 2A. MOS tranzistory v koncovém stupni umožňují generovat PWM signál, zatímco integrované ovládání VREF referenčního napětí jako u předchozích modelů umožňuje přímé nahrazení starších obvodů pro ovládání motoru. Tyto vysoce výkonné obvody se vyznačují efektivním a úsporným provozem. Použití těchto obvodů je běžné v VCR, CD/DVD optických mechanikách atp.
V ostatních variantách tohoto obvodu je možné najít dvoukanálovou variantu nebo obvody pro řízení napětí 3-36V a s nižšími proudy 0,5 a 1A. Na GBd je umístěn přímo obvod BD6222FP což je jednokanálová varianta s pracovním napětím až 18V a proudem až 2A. Obvod umístěný na GBd je rozšířen o tepelnou ochranu a pojistku proti nadměrnému proudovému zatížení viditelné i na schéma Obr. 10.



Obr. 10 schéma obvodu BD6222 umístěného na GBd pro řízení elektromotoru čerpadla.

3.2. Diferenční tlakové čidlo TMDG 338 Z3H

Toto čidlo je na soustavě HPS použito k měření tlaku na dně spodní nádoby. Tlak je přímo úměrný výšce hladiny. Tlakový převodník je určen k měření úrovní tlaku v plynných a kapalných médiích. Jde o pasivní součást regulačního obvodu. Jedná se o převodník tlaku s elektrickým výstupním signálem a parametry:

- Výstupní proud: 0,2 ÷ 1mA
- Výstupní napětí: 0 ÷ 10V
- Napájení : 12 ÷ 36V
- Tlak: 0 ÷ 3kPa

Vzhledem k tlakovému rozsahu použitého měřidla je maximální možná hloubka měření rovna:

$$h_{\max} = \frac{P_{\max}}{\rho \cdot g} = \frac{3000}{1000 \cdot 10} = 0,\overline{33}m \qquad (20)$$

Čidlo je výrobkem firmy Cressto , je osazeno křemíkovou membránou a v současnosti již v této konfiguraci není vyráběno. Jako náhrada jsou dostupné menší čidla typu m, která se vyznačují vyšší tlakovou zatížitelností, napájecím napětím 5V a napěťovým výstupem 0 ÷ 4V nebo 0,5 ÷ 4,5V.

3.3. MCP3002 2.7V Dual Channel 10-Bit A/D Converter with SPI™ Serial Interface

AD převodník MCP3002 umístěný na GBd je dvoukanálový,pro účely připojení k HPS využívám pouze jeden kanál. Připojení k RPi je realizováno přes SPI sběrnici a zpracování dat v pythonu pomocí modulu spidev. AD převodník je napájen napětím z GPIO sběrnice o hodnotě 3,3V. Jelikož je toto napětí i referenční, je třeba napětí z diferenčního tlakového čidla, které může dosahovat hodnoty až 10V redukovat na cca třetinu. Toto lze například napěťovým děličem, nebo pouze trimrem. AD převodník je 10-ti bitový, což znamená, že jeho rozlišitelnost je 2^{10} tedy 1024. při referenčním napětí 3,3V je pak rozlišitelnost (chyba):

$$\Delta V = \frac{3,3}{1024} = 3,223 \cdot 10^{-3} V \tag{21}$$

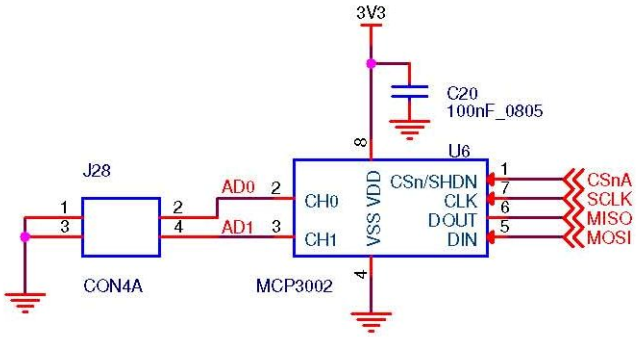
Pro případ kdy je napětí děleno třemi pro snížení rozsahu 0-10V na 0-3,3V je pak rozlišitelnost třikrát větší (horší):

$$\Delta V = \frac{10}{1024} = 9,766 \cdot 10^{-3} V \tag{22}$$

Vzhledem k tomu, že 1V na výstupu diferenčního čidla odpovídá 3/10kPa, je tlaková rozlišitelnost:

$$\Delta P = \frac{3000}{1024} = 2,93 Pa \Rightarrow \Delta h = \frac{3000}{1024 \cdot 1000 \cdot 10} = 0,293 \cdot 10^{-3} \mathrm{m} \tag{23}$$

Nebudeme-li brát v úvahu přesnost tlakového diferenčního čidla závisejícího na provozních stavech (např. množství vzduch v kapalině a tím různá hustota) je použití 10-ti bitového převodníku dostačující při přesnosti měření výšky hladiny ve spodní nádrži do jednoho milimetru.



Obr. 11 Schéma obvodu MCP3002 AD převodníku umístěného na GBd

3.4. Připojení motoru a čidla k vývojovému kitu GBd

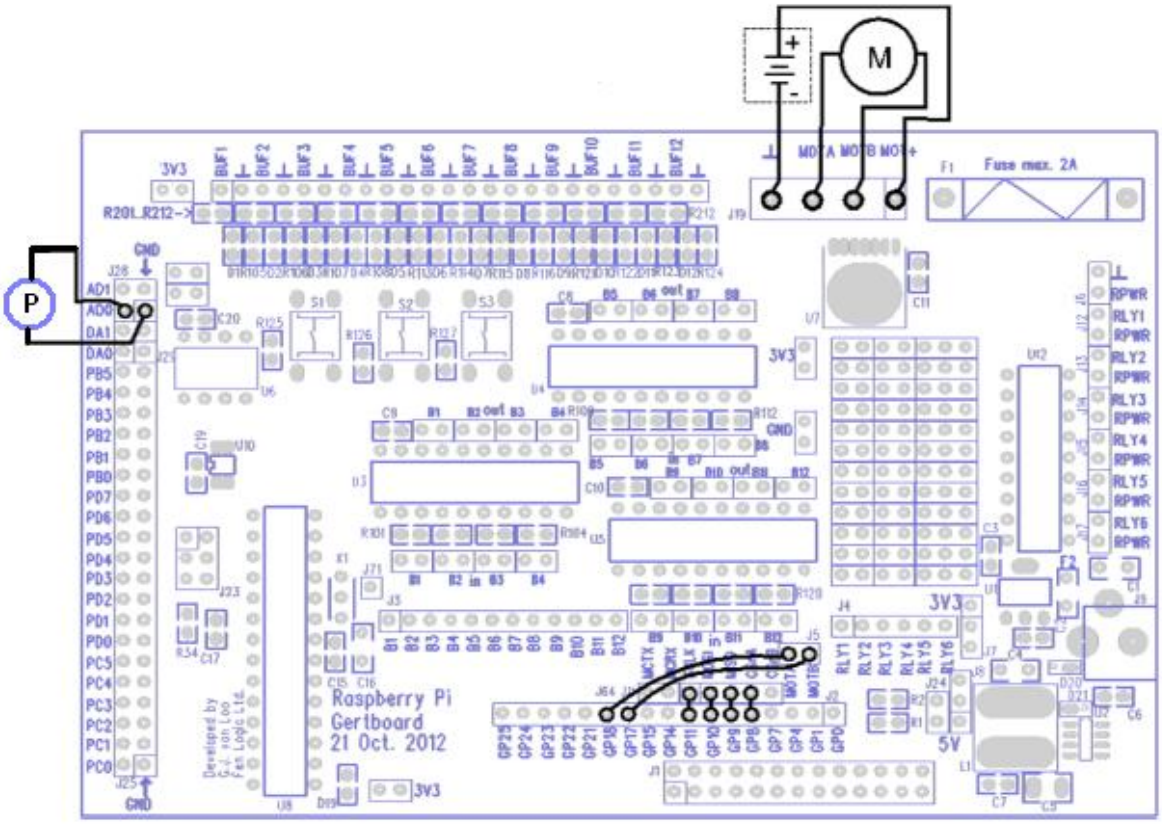
Motor je nutné napájet napětím vyšším než 7V. V případě této DP je použit spínaný zdroj ss napětí o hodnotě 12V a max proudu 2A. Motor je zapojen modrým vodičem do zdířky MOTA konektoru J19 a hnědý vodič do zdířky MOTB. Tlakové diferenční čidlo je napájeno z původní napájecí jednotky ss napětím 12V a výstup čidla je přes napěťový dělič připojen ke kanálu 0 AD převodníku přes kontakt AD0 konektoru J26, je zde provedeno i sekundární pospojování čidla z důvodu rozdílného zdroje napájení než je pro RPi, které GBd napájí.

Pro připojení rozhraní použitého na GBd k RPi je potřeba propojit použité GPIO porty a to následovně:

- Připojení AD převodníku:
  - jumperem propojit GP11 k SCLK na konektoru J64
  - jumperem propojit GP10 k MOSI na konektoru J64
  - jumperem propojit GP9 k MISO na konektoru J64
  - jumperem propojit GP8 k CSnA na konektoru J64

- Připojení motorkontroléru:
  - drátovou spojkou propojit GP17 s MOTB na konektoru J5
  - drátovou spojkou propojit GP18 s MOTA na konektoru J5

Připojení je patrné z Obr. 12, připojení je provedeno přímo na vstupy do jednotlivých obvodů na Obr. 10 a Obr. 11 Je tedy dobře patrné, že Využití desky GBd je zde opravdu vývojové. Oba obvody lze snadno s malými náklady a minimálním zastavěným prostorem realizovat samostatně.



Obr. 12 Připojení motoru čerpadla a tlakového čidla ke GBd

4. Řízení prostředků HPS pomocí jazyka Python

Jak již bylo uvedeno dříve, pro řízení soustavy HPS je použito obvodů MCP3002 a BD6222. Řízení je velmi usnadněno existujícími moduly spidev a wiringpi vytvořenými právě pro podobné aplikace. Je tedy třeba pouze ověřit, zda jejich vlastnosti vyhovují a jak je správně konfigurovat. Pro názornost jsem vytvořil ukázkový skript viz příloha [1] pro P-regulaci PWM výstupního signálu (např. motoru)

4.1. Import modulů:

Rozepíši-li skript po částech tak nejdříve uvolním komunikační kanál a načtu potřebné moduly:

```
import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708',
shell=True, stdout=subprocess.PIPE)
sleep(3)
import spidev
```

```
import wiringpi
```

Je zde vidět i způsob jak v Pythonu spouštět shellové příkazy. Vyčkání 3 vteřiny je zde nezbytné. Někdy i tak (hlavně krátce po startu zařízení) je třeba akci zopakovat. Případ kdy kanál není uvolněn, se projevuje tak, že není získána hodnota z AD převodníku nebo naopak není odeslána hodnota do motorkontroléru. V takové situaci lze skript ukončit (ctrl+c) a znovu spustit. Podruhé již bude kanál jistě vytvořen.

4.2. Nastavení parametrů pro komunikaci s obvodem BD6222

Zde lze nastavit PWM mód řízení, který z pinů bude výstup, směr otáčení motoru (nastavení bitu na zvoleném výstupním pinu) a iniciální hodnotu PWM vstupu (0 aby se motor nezačal samovolně otáčet)

```
wiringpi.wiringPiSetupGpio()                # Initialise wiringpi
 GPIO
wiringpi.pinMode(18,2)                      # Set up GPIO 18 to PWM
 mode
wiringpi.pinMode(17,1)                      # GPIO 17 to output
wiringpi.digitalWrite(17, 0)                # port 17 off for
 rotation one way
wiringpi.pwmWrite(18,0)                     # set pwm to zero
 initially
```

4.3. Zpracování dat z AD převodníku

Tato problematika je u RPi velmi často řešena a vyřešena, proto je její realizace až uspokojivě jednoduchá. Elegantně lze vyřešit použitím modulu spidev, otevřením kanálu na SPI sběrnici kde je obvod umístěn. A vytvořit funkci, která v požadovaný okamžik získá hodnotu ze sledovaného kanálu AD převodníku:

```
spi = spidev.SpiDev()
spi.open(0,0)         # The Gertboard ADC is on SPI channel 0 (CE0 -
                                  aka GPIO8)
channel = 0                     # set ADC channel to 0

def get_adc(channel):                       # read SPI data from
                                  MCP3002 chip
    if ((channel > 1) or (channel < 0)):    # Only channels 0 and 1
                                  else return -1
        return -1
    r = spi.xfer2([1,(2+channel)<<6,0])
    ret = ((r[1]&31) << 6) + (r[2] >> 2)
    return ret
```

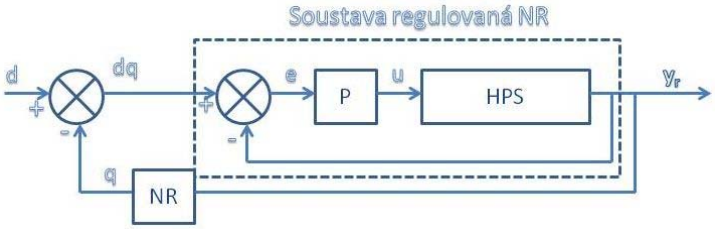
4.4. Zápis hodnoty pro řízení PWM signálu

Zápis je opět prováděn pomocí modulu wiringpi. Zde lze opět zkontrolovat, zda směr otáčení je vyhovující a následně na požadovaný GPIO port (18) zapsat hodnotu PWM signálu do proměnné pwm. Jde o zápis střídy (score) v 10-ti bitové hodnotě tedy 1-1024 a musí se jednat o celé číslo (integer). Pro upřesnění hodnota 1024 odpovídá střídě signálu 100%

```
wiringpi.digitalWrite(17, 0)        # motor spins one way (1 instead 0
for other way)
wiringpi.pwmWrite(18, pwm)          # SET: send PWM value to port 18
```

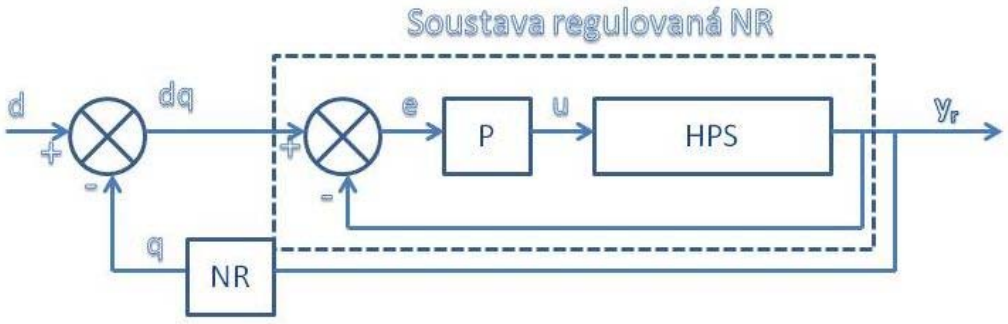
5. Neuroregulátor (NR), vnitřní uspořádání

Neuroregulace HPS je realizována s rozšířením samotné HPS do URO s P-regulátorem Obr. 13



Obr. 13 URO HPS s P-regulátorem

Takováto soustava reprezentuje stav, kdy je již zavedena existující regulace, která však není dostačující. NR pak je implementován jako další zpětnovazební člen připojen ke stávající soustavě Obr. 14 Toto uspořádání má reprezentovat případné nasazení takto vyvinutého NR jako členu, jenž se paralelně připojí k existující soustavě a po adaptaci reguluje vstup do soustavy tak aby výstup ze soustavy odpovídal žádané veličině.



Obr. 14 Umístění NR k existujícímu regulačnímu obvodu

Jelikož NR je nutné adaptovat, bude třeba existující soustavu s P-regulací nejdříve identifikovat. Identifikovaná soustava potom poslouží k adaptaci NR, kdy průběh chyby bude počítán průběhem právě přes identifikovanou soustavu. K identifikaci bude sloužit lineární jednotka, se vstupy žádané a výstupní veličiny. Další mi součástmi NR potom bude referenční model (RM) a vlastní jednotka s algoritmem NR.

5.1. identifikace

Identifikace existující soustavy je realizována metodou gradient descent. Soustavu pak bude popisovat následující polynom:

$$y_{n(k)} = w_0 + w_1 \cdot y_{r(k-1)} + w_2 \cdot y_{r(k-2)} + w_3 \cdot y_{r(k-3)} + w_4 \cdot d_{(k-1)} + w_5 \cdot d_{(k-2)} + w_6 \cdot d_{(k-3)} \quad (24)$$

Samotná metoda Gradient-Descent (někdy také označovaná jako Delta rule) pak pracuje s odchylkou (chybou), tedy rozdílem reálného výstupu soustavy a výstupem neuronové jednotky

Princip metody Gradient-Descent spočívá v hledání směru růstu této odchylky (gradientu). Změna vah je pak provedena přesně opačným směrem, tedy proti směru největšího gradientu funkce odchylky. Tímto dojde ke snížení chyby.

Chyba pro adaptaci vah je v tomto případě definována:

$$e_{(k)} = y_{r(k)} - y_{n(k)} \quad (25)$$

Vlastní úprava jednotlivých vah je potom metodou Gradient-Descent definována:

$$w_{i(k+1)} = w_{i(k)} - \frac{1}{2}\mu \cdot \frac{\partial e_{(k)}^2}{\partial w_i} \quad (26)$$

Přičemž parametr μ se nazývá *learning rate*, neboli rychlost učení a volí se zpravidla menší než 2.

Úpravou dostáváme tvar:

$$\begin{aligned} w_{i(k+1)} &= w_{i(k)} - \frac{1}{2}\mu \cdot 2e_{(k)} \cdot \frac{\partial e_{(k)}}{\partial w_i} \\ w_{i(k+1)} &= w_{i(k)} - \mu \cdot e_{(k)} \cdot \left(\frac{\partial y_{r(k)}}{\partial w_i} - \frac{\partial y_{n(k)}}{\partial w_i} \right) \\ w_{i(k+1)} &= w_{i(k)} + \mu \cdot e_{(k)} \cdot \frac{\partial y_{n(k)}}{\partial w_i} \end{aligned} \quad (27)$$

$y_{r(k)}$ je reálný výstup a jde o měřenou veličinu, která není nijak závislá na vahách $w_{i(k)}$. Její derivace podle těchto vah je tedy nulová.

Pro výše zvolenou funkci ((24) je pak vyjádření úpravy jednotlivých vah následující:

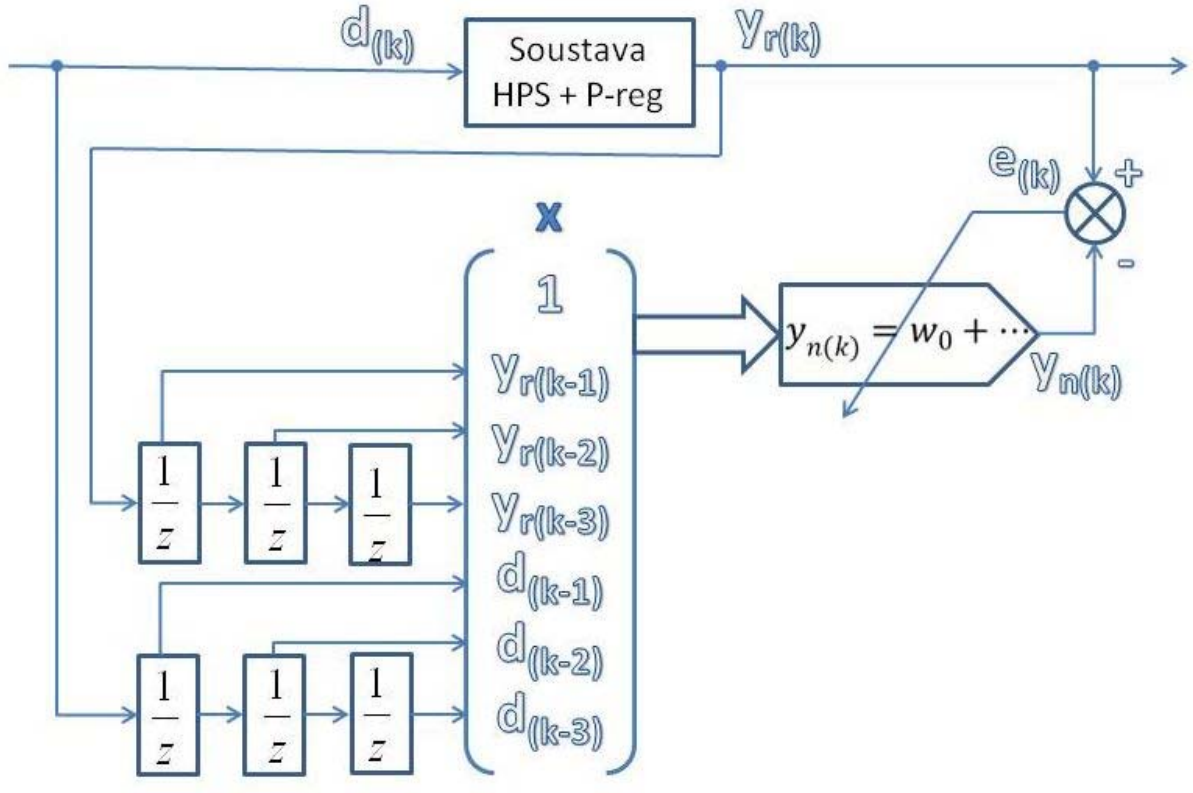
$$\begin{aligned} w_{0(k+1)} &= w_{0(k)} + \mu \cdot e_{(k)} \\ w_{1(k+1)} &= w_{1(k)} + \mu \cdot e_{(k)} \cdot y_{r(k-1)} \\ w_{2(k+1)} &= w_{2(k)} + \mu \cdot e_{(k)} \cdot y_{r(k-2)} \\ &\vdots \\ w_{5(k+1)} &= w_{5(k)} + \mu \cdot e_{(k)} \cdot d_{(k-2)} \\ w_{6(k+1)} &= w_{6(k)} + \mu \cdot e_{(k)} \cdot d_{(k-3)} \end{aligned} \quad (28)$$

Takto vyjádřené vztahy pro adapataci jednotlivých vah lze aplikovat na naměřenou přechodovou charakteristiku. Za $y_{r(k)}$a $d_{(k)}$ dosadím postupně všechny naměřené vzorky. Tuto akci opakuji

v několika epochách. Omezení počtu epoch může být různé. Např. přímo na počet epoch, na čas výpočtu nebo na požadovanou chybu $e_{(k)}$

Pro zjednodušení výpočtu lze vstupy a zpožděné vstupy polynomu sestavit do sloupcového vektoru **x**, stejně tak váhy w_0 ...w_6 sestavit také do sloupcového vektoru **w**. Sestavení vektoru **x** a adaptace hodnot ve vektoru **w** je patrná z Obr. 15

$$\mathbf{x_{(k)}} = \left[1, y_{r(k-1)}, y_{r(k-2)}, y_{r(k-3)}, d_{(k-1)}, d_{(k-2)}, d_{(k-3)}\right]^{\mathbf{T}}$$
$$\mathbf{w_{(k)}} = \left[w_0, w_1, w_2, w_3, w_4, w_5, w_6\right]^{\mathbf{T}} \tag{29}$$



Obr. 15 schéma uspořádání identifikace soustavy

Funkce průběhu lineárním neuronem pak vypadá takto:

$$y_{n(k)} = \mathbf{w}_{(k)} \cdot \mathbf{x}_{(k)} \tag{30}$$

Odchylka je pak vypočtena stejně :

$$e_{(k)} = y_{r(k)} - y_{n(k)} \tag{31}$$

Vlastní adaptace vah lze zapsat jako:

$$\mathbf{w}_{(k+1)} = \mathbf{w}_{(k)} - \mathbf{dw}_{(k)} \tag{32}$$

Přičemž ***dw*** počítám metodou Gradient-Descent takto:

$$\mathbf{dw}_{(k)} = \mu \cdot e_{(k)} \cdot \mathbf{x}'_{(k)} \tag{33}$$

Parametr μ (learning rate) lze pro každý krok (cyklus nebo epochu) adaptovat viz. [7] a [8]. Lze použít například tyto úpravy:

$$\mu_{(k)} = \frac{\mu_{(k-1)}}{\mathbf{x}'_{(k)} \cdot \mathbf{x}_{(k)} + \varepsilon} \tag{34}$$

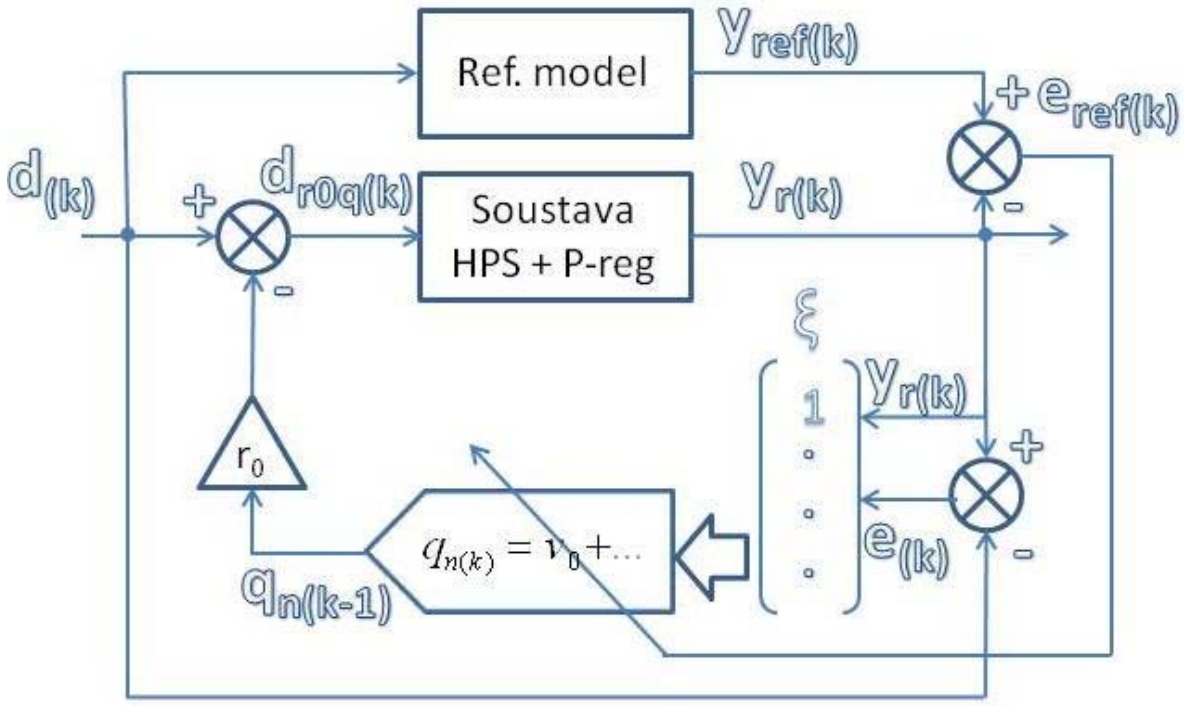
Nebo:

$$\mu_{(k)} = \frac{\mu_{(k-1)}}{\sqrt{\mathbf{x}'_{(k)} \cdot \mathbf{x}_{(k)} + \varepsilon}} \tag{35}$$

Přičemž parametr ε je opět volen v rozmezí 0-1.

5.2. Referenční model

Vzhledem k fyzikálním omezením soustavy HPS nelze očekávat například dokonalé sledování pulzující žádané veličiny. Při adaptaci NR na takovýto vstup by docházelo k výrazné nestabilitě. Proto je vhodné definovat referenční model, jenž bude obrazem toho, jak chceme aby soustava na žádanou veličinu reagovala. A naopak neumožňovala chování, které nechceme např. takové, které odporuje fyzikálním zákonům. Neuronová jednotka, která se bude na soustavu HPS adaptovat pak bude adaptována odchylkou reálného výstupu od výstupu referenční soustavy $e_{ref(k)}$, zatímco regulační odchylka $e_{(k)}$ bude vstupem do jednotky dle Obr. 16.



Obr. 16 Umístění referenčního modelu v soustavě s NR

odchylku referenčního modelu pro adaptaci neuronové jednotky pak počítáme z následujícího vztahu:

$$e_{ref(k)} = y_{ref(k)} - y_{r(k)} \tag{36}$$

Jako vhodný referenční model pro soustavu HPS se ukázal například systém popsaný takto:

$$G(S) = \frac{S}{(\tau_1 s + 1)(\tau_2 s + 1)} \tag{37}$$

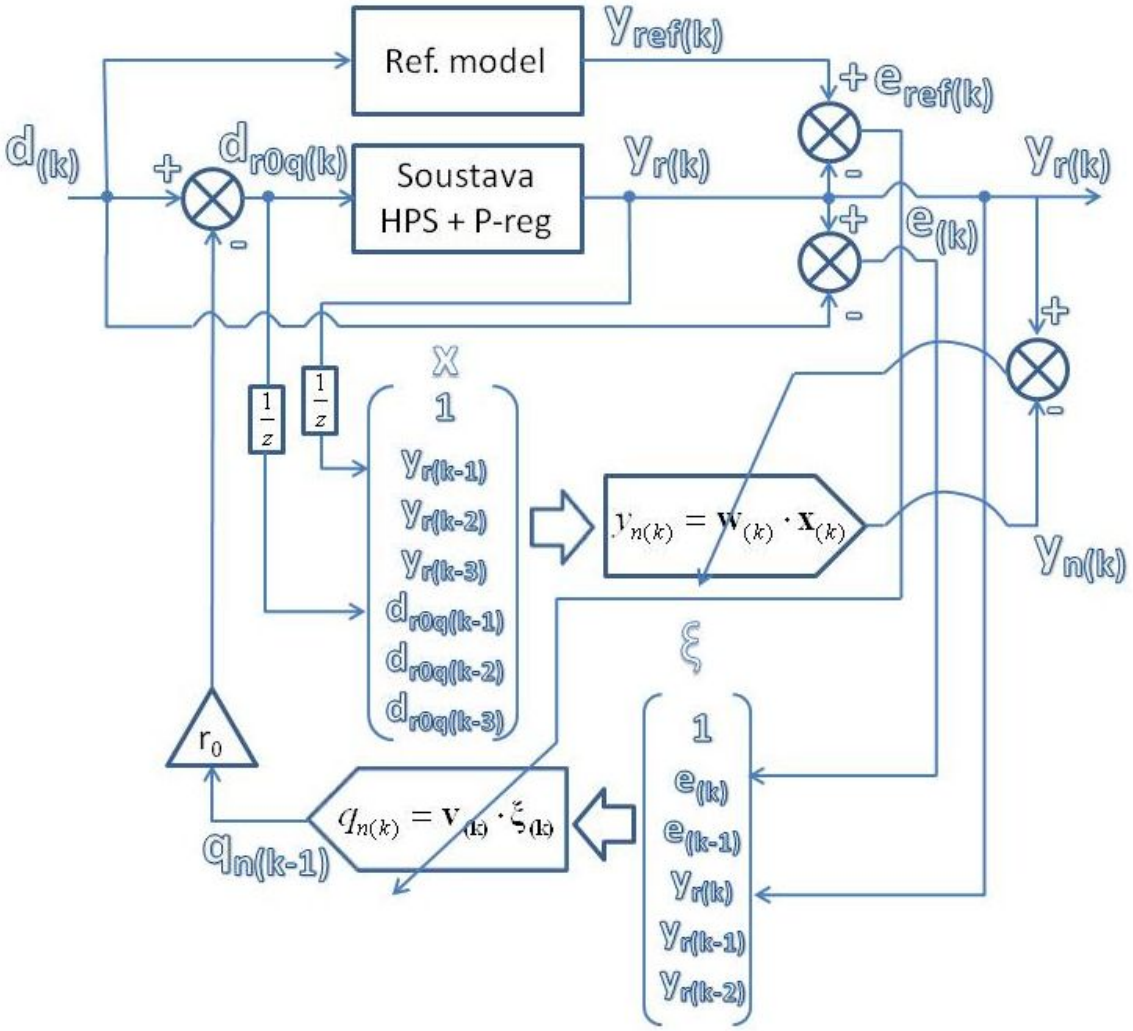
5.3. Neuronová jednotka pro NR

Vhodnou neuronovou jednotkou pro otestování řízení HPS pomocí NR je lineární neuron, další možnou jednotkou může být například kvadratická neuronová jednotka QNU.

Jako vhodný sytém pro regulaci se ukázala lineární jednotka na

Obr. 17 kde je navržen i způsob adaptace a současné identifikace, jejímž výsledkem bude počítán růst chyby. Rovnice neuronu je:

$$q_{n(k)} = v_0 + v_1 \cdot e_{(k)} + v_2 \cdot e_{(k-1)} + v_3 \cdot y_{r(k)} + v_4 \cdot y_{r(k-1)} + v_5 \cdot y_{r(k-2)} \tag{38}$$



Obr. 17 Mechanizmus výpočtu lineárního neuronu, současná identifikace a adaptace

Vstupy jsou potom reálný výstup y_r celé soustavy a regulační odchylka *e.* Metoda adaptace bude opět gradient-descent a chyba proti jejímuž růstu algoritmus půjde, se bude počítat jako rozdíl reálného výstupu a referenčního

Vstupní vektor je **ξ**. Vektor pak je sestaven takto:

$$\boldsymbol{\xi}_{(\mathbf{k})} = \left[1, e_{(k)}, e_{(k-1)}, y_{r(k)}, y_{r(k-1)}, y_{r(k-2)}\right]^{\mathbf{T}} \tag{39}$$

Vektor vah je pak tento:

$$\mathbf{v}_{(\mathbf{k})} = \left[v_0, v_1, v_2, v_3, v_4, v_5\right]^{\mathbf{T}} \tag{40}$$

Chyba pro adaptaci vah je v tomto případě definována:

$$e_{ref(k)} = y_{ref(k)} - y_{r(k)} \tag{41}$$

Vlastní úprava jednotlivých vah je potom metodou Gradient-Descent definována:

$$v_{i(k+1)} = v_{i(k)} - \frac{1}{2}\mu \cdot \frac{\partial e^2_{ref(k)}}{\partial v_i} \tag{42}$$

Přičemž parametr μ je opět rychlost učení a může být rozdílný od rychlosti učení identifikační jednotky.

Úpravou dostáváme tvar:

$$\begin{aligned} v_{i(k+1)} &= v_{i(k)} - \frac{1}{2}\mu \cdot 2e_{ref(k)} \cdot \frac{\partial e_{ref(k)}}{\partial v_i} \\ v_{i(k+1)} &= v_{i(k)} - \mu \cdot e_{ref(k)} \cdot \left(\frac{\partial y_{ref(k)}}{\partial v_i} - \frac{\partial y_{r(k)}}{\partial v_i}\right) \\ v_{i(k+1)} &= v_{i(k)} + \mu \cdot e_{ref(k)} \cdot \frac{\partial y_{r(k)}}{\partial v_i} \end{aligned} \tag{43}$$

$y_{ref(k)}$ je výstup z referenčního modelu a jde o veličinu, která není nijak závislá na vahách $v_{i(k)}$. Její derivace podle těchto vah je tedy nulová.

Závislost $y_{r(k)}$ na vahách $v_{i(k)}$ vyjádříme formální substitucí identifikované soustavy s reálnou:

$$y_{r(k)} \approx y_{n(k)} = w_0 + w_1 \cdot y_{r(k-1)} + w_2 \cdot y_{r(k-2)} + w_3 \cdot y_{r(k-3)} + w_4 \cdot d_{r0q(k-1)} + w_5 \cdot d_{r0q(k-2)} + w_6 \cdot d_{r0q(k-3)} \tag{44}$$

Kde:

$$d_{r0q(k)} = d_{(k)} - r_0 \cdot q_{n(k)} = d_{(k)} - r_0 \cdot \left(v_0 + v_1 \cdot e_{(k)} + v_2 \cdot e_{(k-1)} + v_3 \cdot y_{r(k)} + v_4 \cdot y_{r(k-1)} + v_5 \cdot y_{r(k-2)} \right) \tag{45}$$

Obecně lze zapsat jako :

$$v_{i(k+1)} = v_{i(k)} + \mu \cdot e_{ref(k)} \cdot \frac{\partial y_{n(k)}}{\partial v_i} = v_{i(k)} + \mu \cdot e_{ref(k)} \cdot \frac{\partial (\mathbf{w} \cdot \mathbf{x_{(k)}})}{\partial v_i} \tag{46}$$

$$v_{i(k+1)} = v_{i(k)} + \mu \cdot e_{ref(k)} \cdot \mathbf{w} \cdot \frac{\partial \mathbf{x_{(k)}}}{\partial v_i} \tag{47}$$

$$\frac{\partial \mathbf{x_{(k)}}}{\partial v_i} = \frac{\partial}{\partial v_i} \begin{bmatrix} y_{r(k-1)} \\ y_{r(k-2)} \\ y_{r(k-3)} \\ d_{r0q(k-1)} \\ d_{r0q(k-2)} \\ d_{r0q(k-3)} \end{bmatrix} = \frac{\partial}{\partial v_i} \begin{bmatrix} 0 \\ 0 \\ 0 \\ d_{(k-1)} - r_0 \cdot q_{(k-1)} \\ d_{(k-2)} - r_0 \cdot q_{(k-2)} \\ d_{(k-3)} - r_0 \cdot q_{(k-3)} \end{bmatrix} = -r_0 \cdot \frac{\partial}{\partial v_i} \begin{bmatrix} 0 \\ 0 \\ 0 \\ q_{(k-1)} \\ q_{(k-2)} \\ q_{(k-3)} \end{bmatrix} \tag{48}$$

$$\frac{\partial q_{(k-1)}}{\partial v_i} = \frac{\partial \left(\mathbf{v} \cdot \boldsymbol{\xi}_{\mathbf{(k-1)}} \right)}{\partial v_i} = \xi_{i(k-1)} \tag{49}$$

Pro výše zvolenou funkci (Vztah 38) je pak vyjádření úpravy jednotlivých vah následující:

$$\mathbf{v}_{(k+1)} = \mathbf{v}_{(k)} + \mu \cdot e_{ref(k)} \cdot \mathbf{w} \cdot \left(-r_0 \right) \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \xi_{0(k-1)} & \cdots \; \xi_{i(k-1)} & \cdots \; \xi_{5(k-1)} \\ \xi_{0(k-2)} & \xi_{i(k-2)} & \xi_{5(k-2)} \\ \xi_{0(k-3)} & \xi_{i(k-3)} & \xi_{5(k-3)} \end{bmatrix} \tag{50}$$

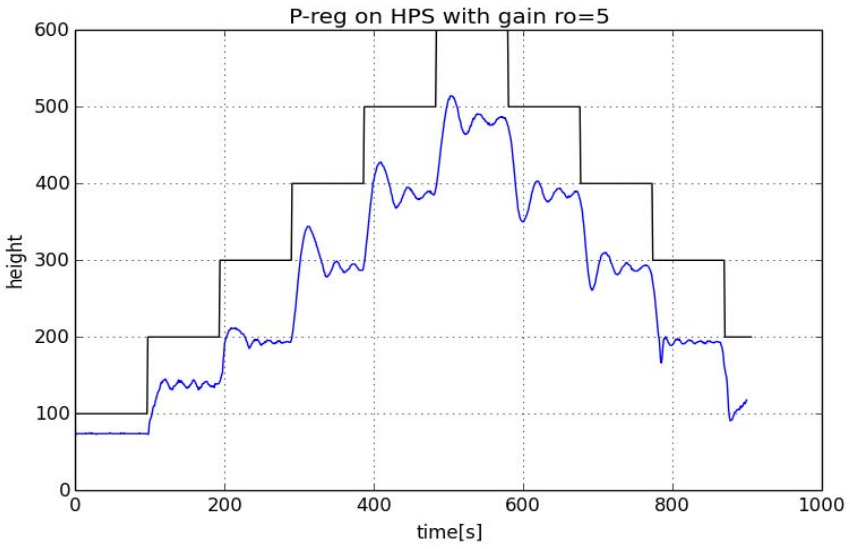
$$\mathbf{v}_{(k+1)} = \mathbf{v}_{(k)} + \mu \cdot e_{ref(k)} \cdot \mathbf{w} \cdot \left(-r_0 \right) \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & e_{(k)} & e_{(k-1)} & y_{r(k)} & y_{r(k-1)} & y_{r(k-2)} \\ 1 & e_{(k-1)} & e_{(k-2)} & y_{r(k-1)} & y_{r(k-2)} & y_{r(k-3)} \\ 1 & e_{(k-2)} & e_{(k-3)} & y_{r(k-2)} & y_{r(k-3)} & y_{r(k-4)} \end{bmatrix} \tag{51}$$

6. Předtrénování a adaptace LNU

V kapitole 5 Je popsána vnitřní struktura pohyb proměnných v soustavě pro řízení soustavy pomocí LNU. V přílohách 2-4 je jejich programová realizace v jazyce python běžícím na RPi. Následující popis shrnuje potřebné vstupy pro jednotlivé kroky a jejich realizaci

6.1. Příprava dat

Z měření na reálné úloze HPS vyplynula tato omezení. Diferenční tlakové čidlo bez vodního sloupce (na suchu) po nížení signálu na třetinové napětí, má na výstupu AD převodníku hodnotu 35 - 40 (z rozsahu 0 - 1023) Funkce žádané veličiny d=f(t), by tedy neměla zasahovat pod tuto hodnotu. Při naplnění obou sloupců a přetékání kapaliny oběma přepady, je hodnota poslaná přes SPI sběrnici cca 800. Tedy požadovat hodnotu vyšší je opět nesmyslné. Pro objektivní reprezentaci chování soustavy HPS s P-regulátorem jsem vybral funkci, která každou minutu a půl drží hladinu žádané veličiny a pak ji o 100 zvýší až na hodnotu 600 (75% objemu spodní nádrže) a poté začne opět po 100 klesat. Spodní omezení jsem definoval na hodnotu 50. Změřená charakteristika trvající cca 15 minut je zobrazena na Obr. 18 a je výsledkem regulace skriptem v příloze 2



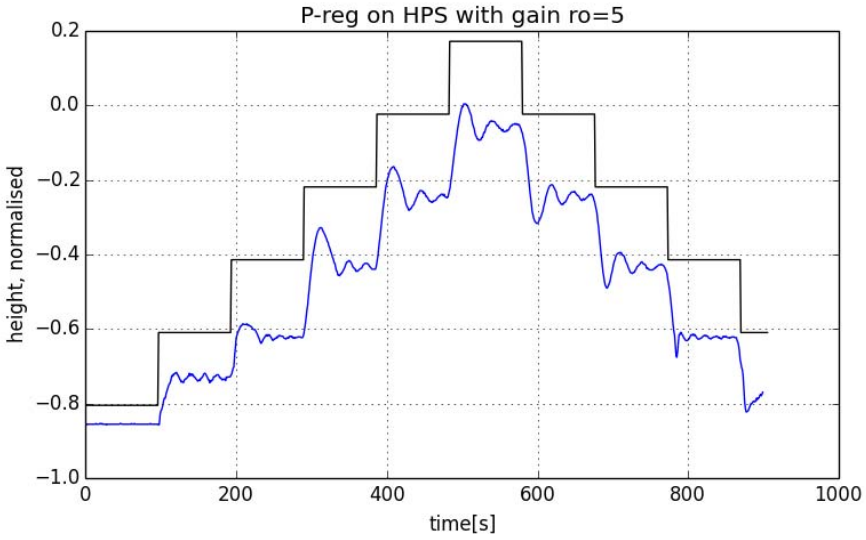
Obr. 18 Změřená charakteristika HPS s P-regulátorem o zesílení 5, modrá y_r, černá d

6.2. Zpracování naměřených dat

Pro další zpracování dat, je vhodné tyto nejdříve normalizovat do rozsahu <-1,1>. Vlastní rozsah, AD převodníku a PWM kontroléru svými rozsahy usnadňuje rozhodování o parametrech normalizace. AD převodník má na výstupu 10-ti bitovou hodnotu o rozsahu 0 - 1023 a PWM kontrolér má na vstupu 10-ti bitový integer, tedy 1-1024 Normalizovat budu tak, že od každé hodnoty odečtu její průměr a to vydělím polovinou maximální hodnoty reprezentující směrodatnou odchylku. Pro hodnotu y takto:

$$norm\_y_{(k)} = \frac{y_{(k)} - \overline{y_{(k)i}}}{\max(y_{(k)})/2} = \frac{y_{(k)} - 512}{512} \tag{52}$$

Pro tento případ budu předpokládat pro hodnotu průměru polovinu z 1024 a to samé v podílu. Tím pádem od každé hodnoty odečtu 512 a vydělím ji také 512

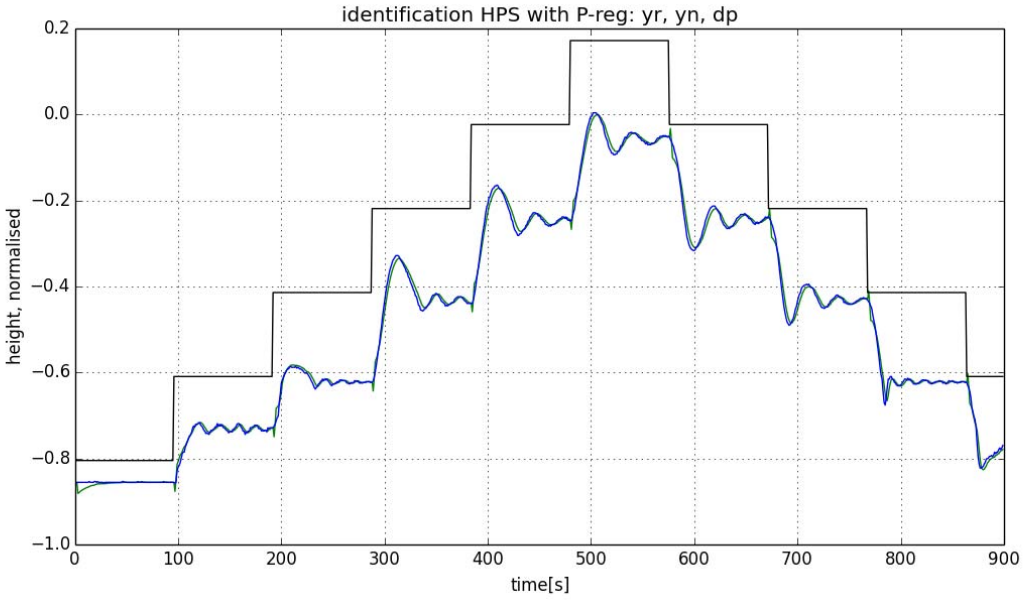


Obr. 19 Naměřené veličiny po normalizaci modrá y_r, černá d

6.3. Identifikace soustavy LNU

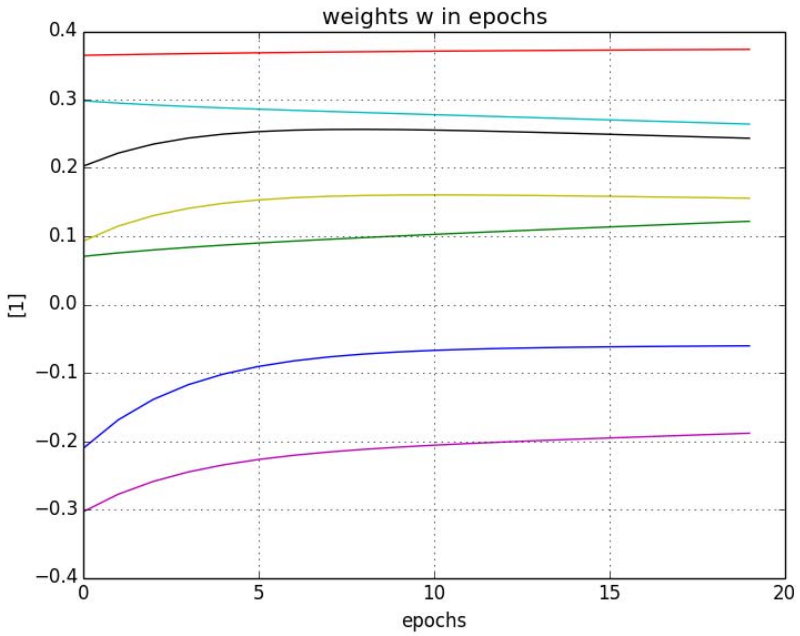
Na takto získaná data natrénuji LNU způsobem popsaným v kapitole 5.1 a dle toho vytvořeným skriptem, který je součástí komplexnějšího skriptu v příloze 3, tento je pak pro realizaci celkové adaptace popsané v další kapitole.

$$y_{n(k)} = w_0 + w_1 \cdot y_{r(k-1)} + w_2 \cdot y_{r(k-2)} + w_3 \cdot y_{r(k-3)} + w_4 \cdot d_{(k-1)} + w_5 \cdot d_{(k-2)} + w_6 \cdot d_{(k-3)}$$

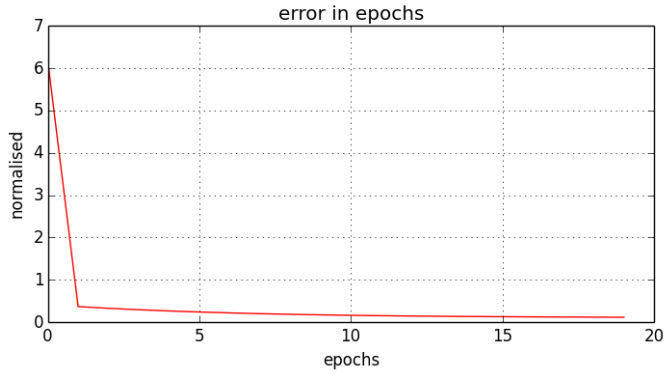


Obr. 20 Identifikovaná soustava po 20-ti epochách s μ=0,1, zelená y_n, modrá y_r, černá d

Vývoj vah v epochách je pak takovýto:

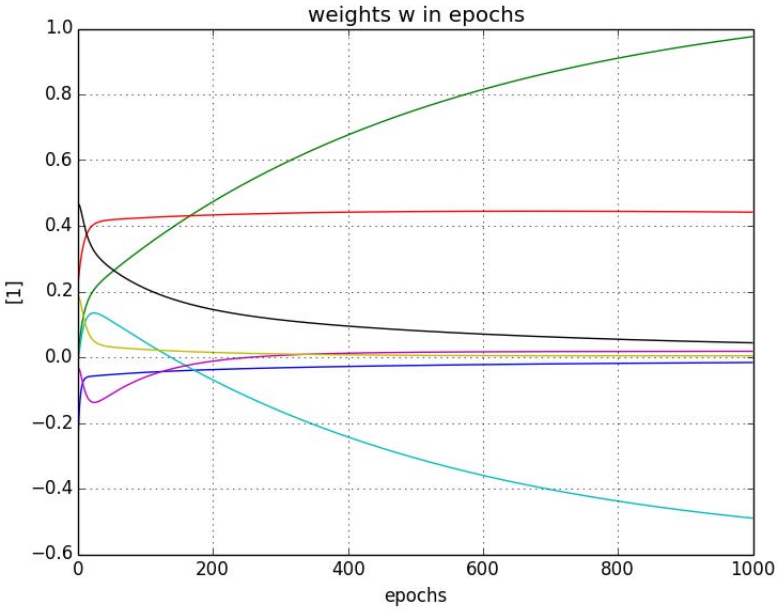


Obr. 21 Vývoj adaptace vah v průběhu učení 20 epoch, μ=0,1



Obr. 22 Vývoj kvadrátu chyb v epochách u identifikace 20 epoch, μ=0,1

Trénování větším počtem epoch dojde k dokonalému splynutí křivek yr a yn, Pro ilustraci tedy jen vývoj vah a kvadrátu chyb pro 1000 epoch

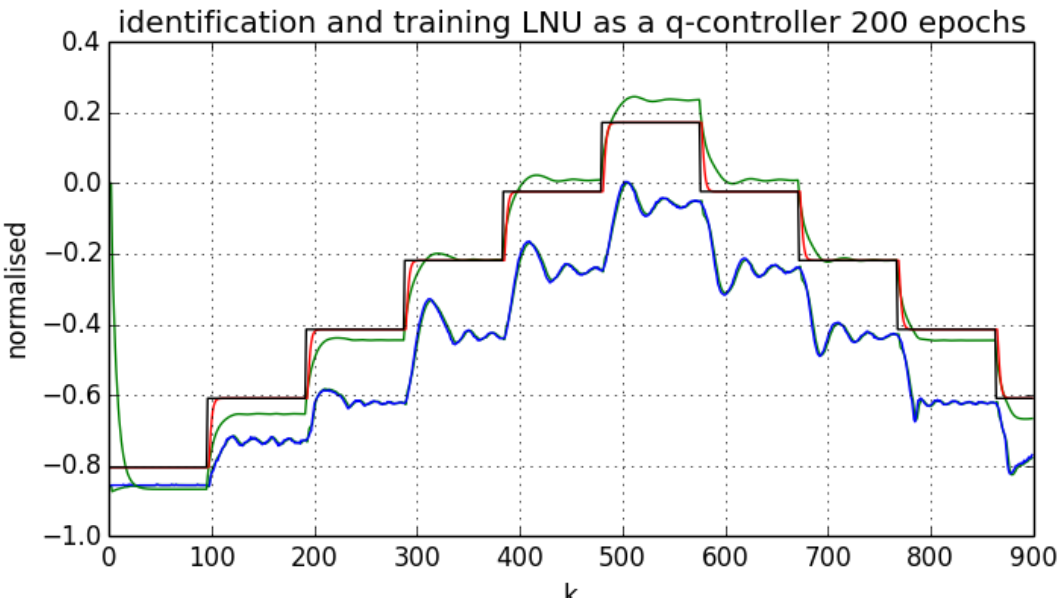


Obr. 23 Vývoj vah při identifikaci 1000 epoch μ=0,1

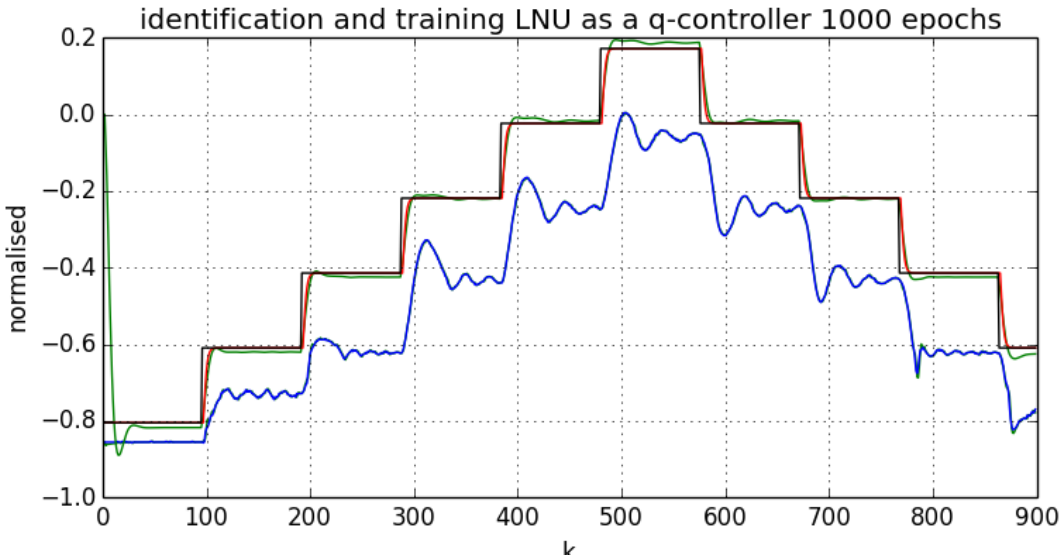
Obr. 24 Vývoj kvadrátu chyby při identifikaci 1000 epoch μ=0,1

6.4. Předtrénování LNU pro NR

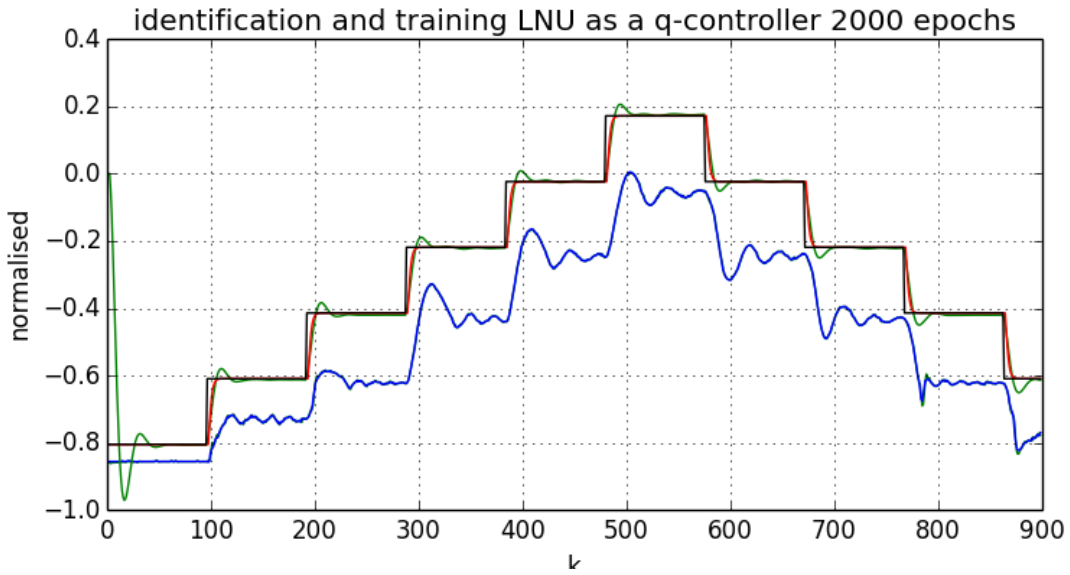
Při doplnění algoritmu identifikace o referenční model a adaptační funkci pro předtrénování LNU pro adaptaci NR získáme komplexní skript uvedený v příloze 3. jeho realizací získávám pro různý počet epoch následující výsledky:



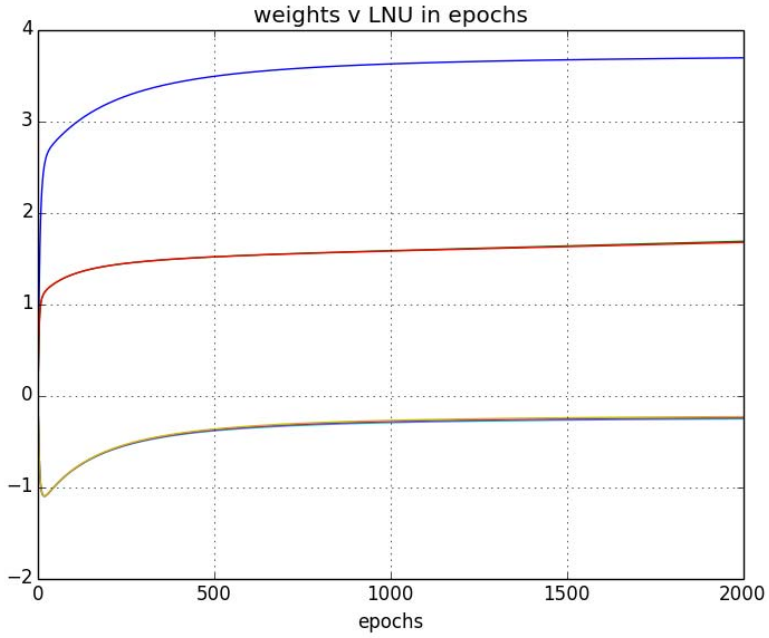
Obr. 25 Výsledek předtrénování LNU 200 epochami na naměřená data . zelená y_n po adaptaci, modrá naměřené y_r, černá žádané d, červená referenční y_{ref}, μ=0,1, μv=1



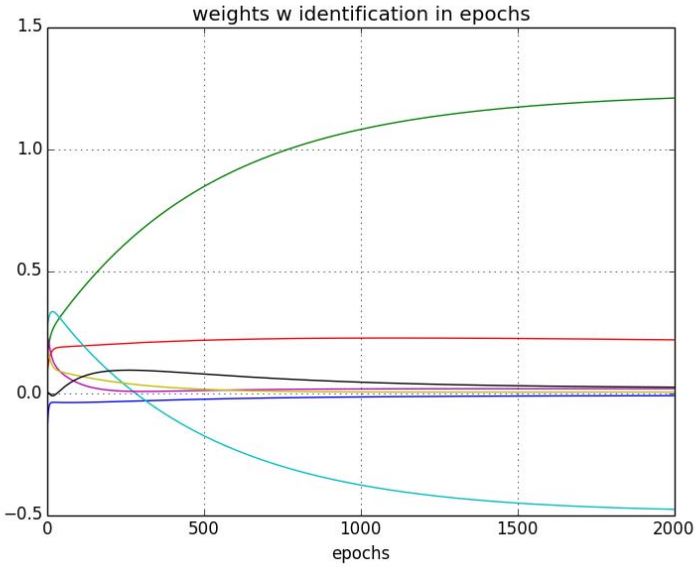
Obr. 26 Výsledek předtrénování LNU 1000 epochami na naměřená data . zelená y_n po adaptaci, modrá naměřené y_r, černá, žádané d, červená, referenční y_{ref}, μ=0,1, μv=1



Obr. 27 Výsledek předtrénování LNU 2000 epochami na naměřená data . zelená y_n po adaptaci, modrá naměřené y_r, černá, žádané d, červená, referenční y_{ref}, μ=0,1, μv=1



Obr. 28 Vývoj vah v, adaptujícího se LNU pro 2000 epoch



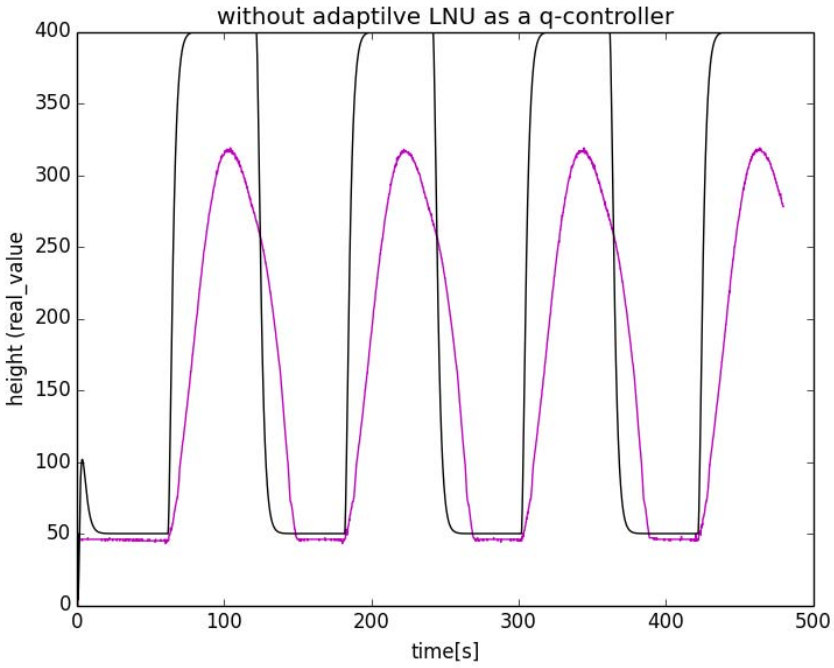
Obr. 29 vývoj vah w identifikujícího LNU pro 2000 epoch

7. Řízení úlohy

Úpravou skriptu pro adaptaci a identifikaci mohu získat výstup, který budu odečítat od žádané veličiny d, jak je navrženo v kapitole 5. Jako vhodnou funkci pro ověření funkce NR jsem zvolil pulzy od minimální hodnoty s amplitudou 350 a periodou 1min.

7.1. Řízení samotným P-regulátorem

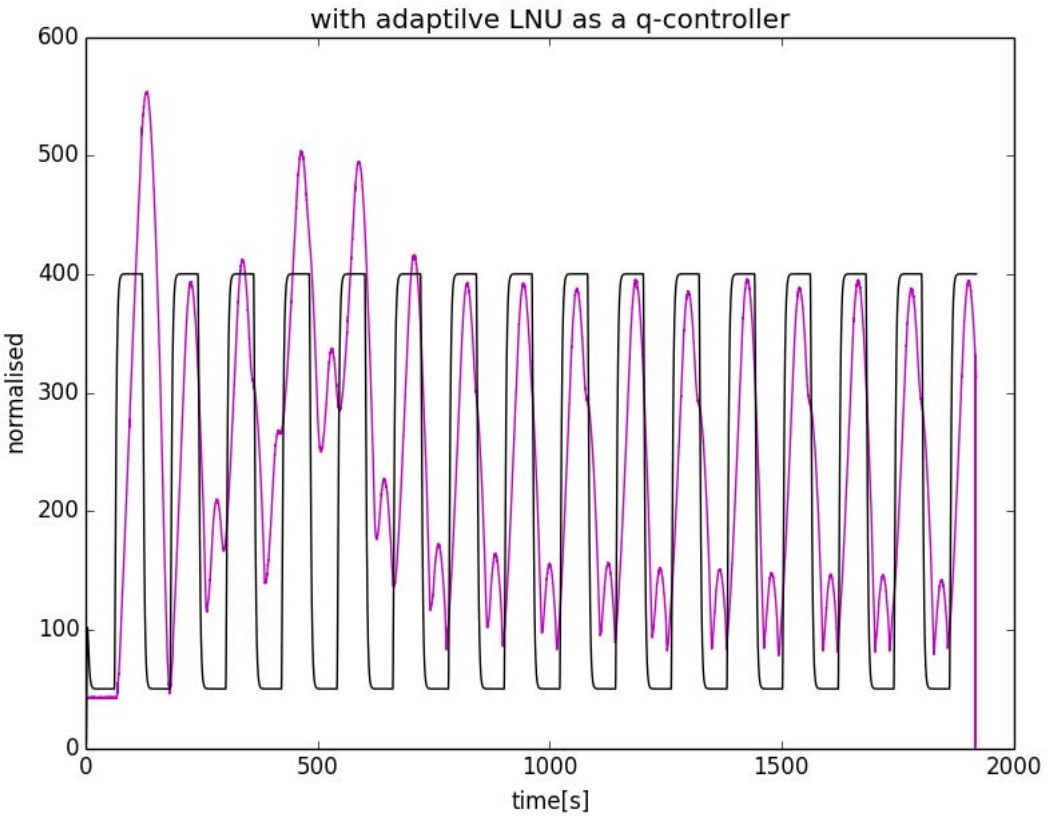
Pro pořádek zde uvádím výsledek regulace samotným P-regulátorem. Trvalá regulační odchylka není nijak překvapivá a řízení by jistě šlo bez požadavku na predikci vylepšit I nebo D složkou.



Obr. 30 Řízení HPS pouze P-regulátorem, černá y_{ref}, fialová y_r

7.2. Řízení bez předchozí identifikace nebo předtrénování

Na Obr. 31 je zobrazen výsledek regulace bez jakékoliv pomoci předidentifikace, nebo předtrénování uvedeným v kapitolách 6.3 a 6.4 Identifikace a adaptace NR se tedy provádí až za běhu regulátoru. Váhy pro identifikaci jsou zprvu malé náhodné a váhy pro adaptaci LNU jsou na začátku nulové.

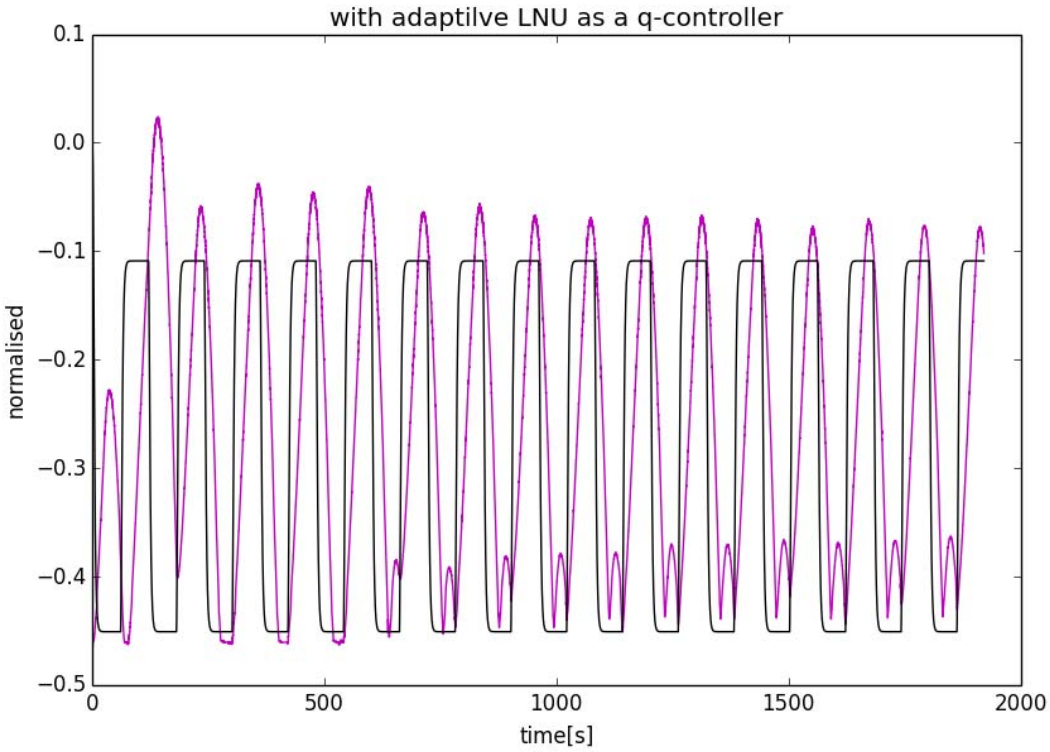


Obr. 31 Regulace na pulzy s periodou 60s bez předchozí identifikace soustavy, černá y_{ref}, fialová y_r, μ=0,1, μv=1, w=random, v =zeros

Je patrné, že prvních 15 minut (časový interval závisí na náhodných vahách)je regulace nestabilní a od extrémních výkyvů chrání soustavu HPS pouze její dlouhá odezva. Než tedy dosáhne měřená hodnota některého extrému dojde k adaptaci a soustava se postupně uklidní do výsledku, kdy je patrné, že NR predikuje ze znalosti předchozího vývoje žádané veličiny (potažmo jejímu referenčnímu výstupu) konec pulzu a reaguje v předstihu.

7.3. Řízení se předchozí identifikací soustavy HPS s P-reg

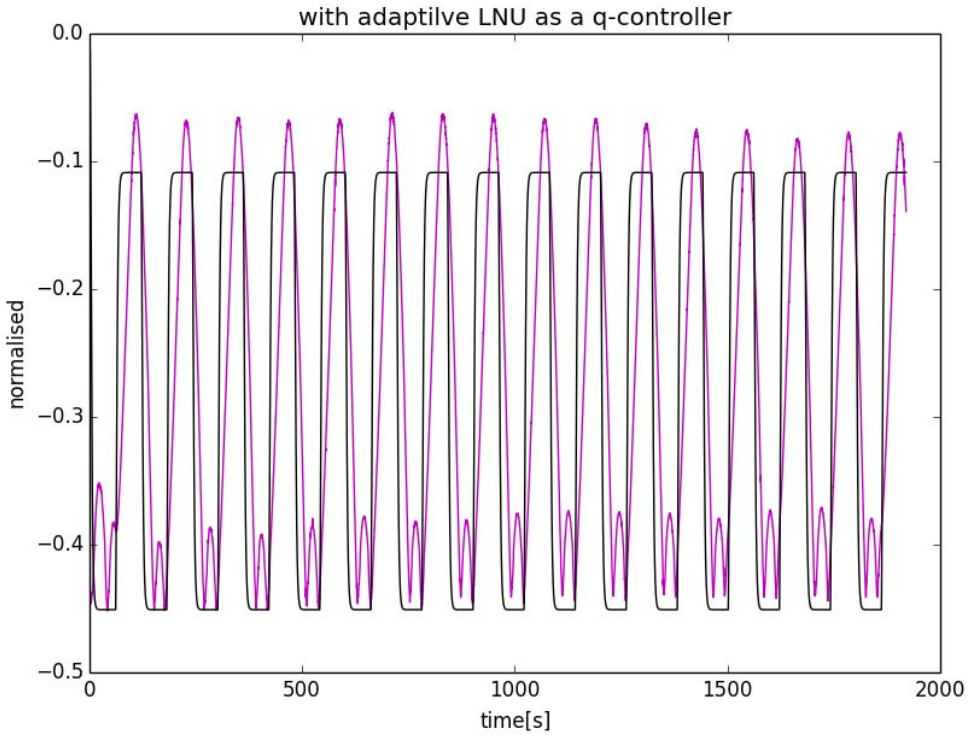
Na Obr. 32 je uveden výsledek pro regulaci soustavy s předchozí identifikací soustavy popsanou v kapitole 6.3 Je zde patrná výrazně stabilnější a kratší oblast na začátku regulace, než v předchozí kapitole. Navíc je výsledná regulace výrazně agresivnější. Horní nádoba je napouštěna rychleji a déle, tím i po vypnutí čerpadla se hladina ve spodní nádrži zvyšuje. Tím dochází k překročení žádané veličiny v horní amplitudě pulzu.



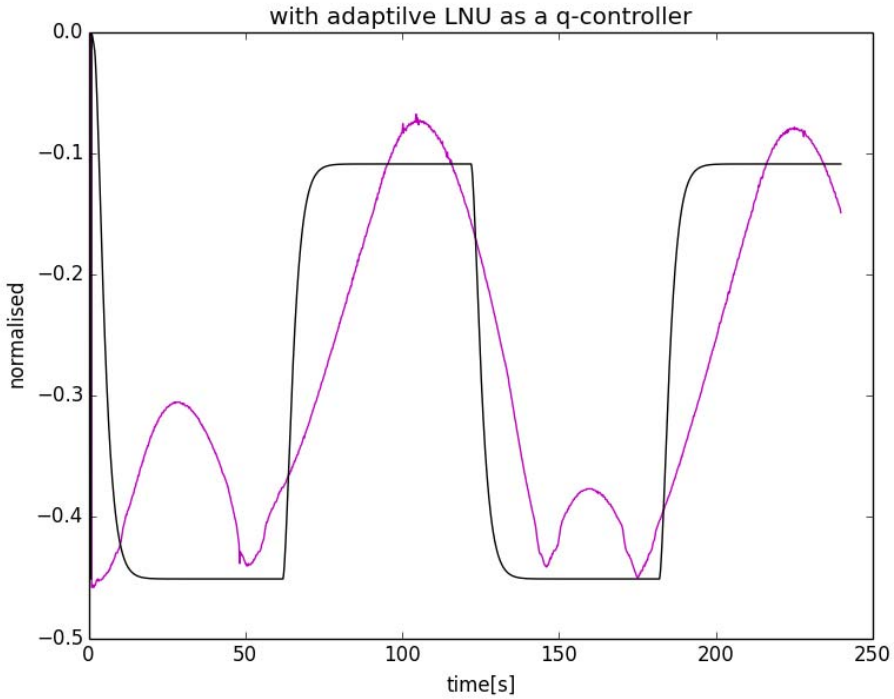
Obr. 32 Regulace na pulzy s periodou 60s s pouze s identifikovanou soustavou, černá y_{ref}, fialová y_r, μ=0,1, μv=1, v=zeros

7.4. Řízení s předchozí identifikací soustavy HPS s P-reg a předtrénováním NR

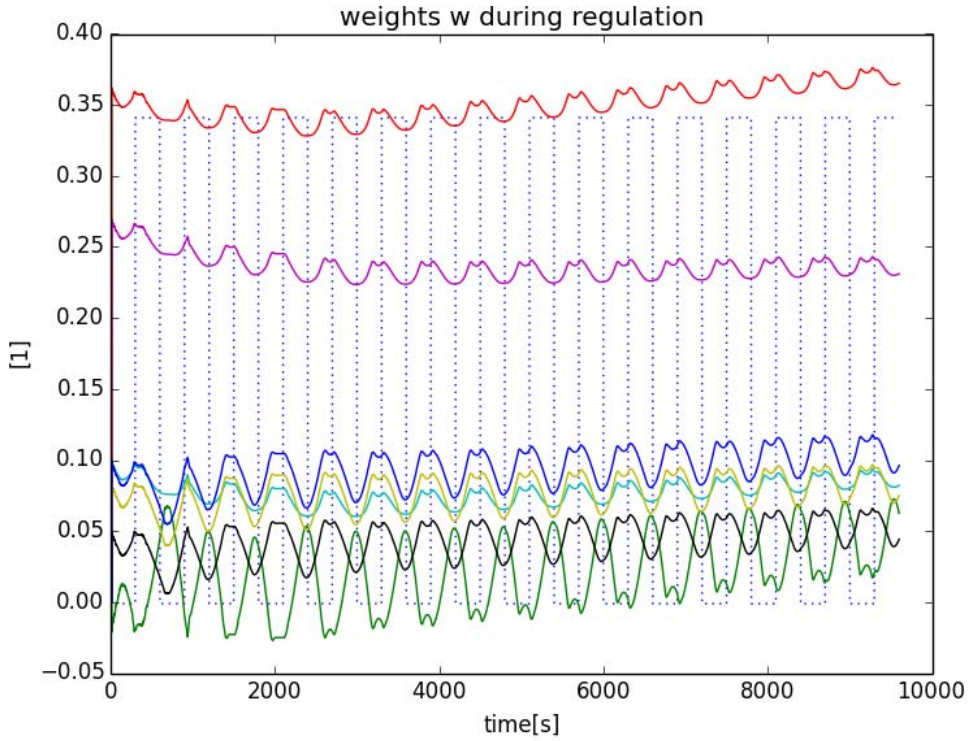
Na Obr. 33 je zobrazen výsledek regulace s předem identifikovanou soustavou na trénovací množině rozdílné od aktuální funkce žádané veličiny a předtrénováním soustavy na výstup y_n, takto identifikované soustavy. Identifikace s předtrénováním probíhala tak jak je popsáno v kapitolách 6.3 a 6.4 Je zde patrná vysoká stabilita od počátku. Přesto dochází k přeběhu v horní úvrati amplitudy a k nestabilitě v dolní úvrati. Dobře patrné je to na Obr. 34, kde je zobrazen detail začátku regulace. Tato nestabilita vzniká prediktivním chováním NR při adaptaci GD, který očekává změnu, ke které tak rychle ještě nedošlo. Toto nastane změnou směru gradientu chyby procházející identifikovanou soustavou. Vývoje vah **w** a **v** jsou zobrazeny na Obr. 35 Obr. 36. Je z nich patrné, že soustava ač jde o ne opakující se děj není stále ještě plně identifikována (váhy **w**). Adaptace řídícího LNU reaguje na měnící se váhy identifikovaného modelu. A navíc dvě váhy stále ještě rychle mění svou hodnotu.



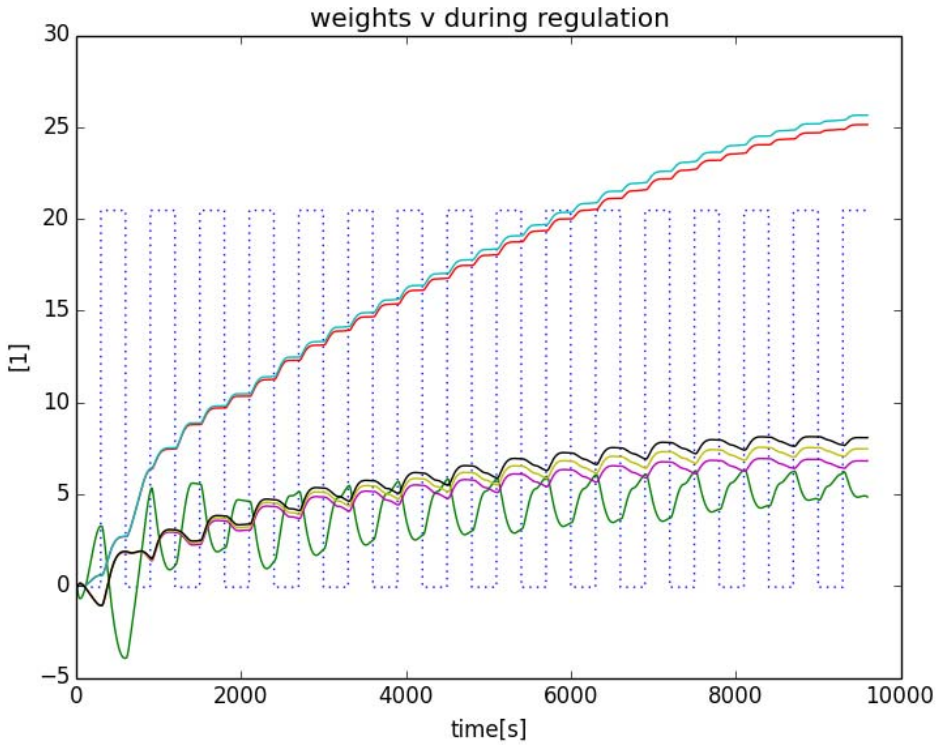
Obr. 33 Regulace na pulzy s periodou 60s s předtrénovaným neuronem a identifikovanou soustavou, černá y_{ref}, fialová y_r, μ=0,1, μv=1



Obr. 34 Krátká regulace na pulzy s periodou 60s s předtrénovaným neuronem a identifikovanou soustavou, černá y_{ref}, fialová y_r, μ=0,1, μv=1



Obr. 35 Vývoj vah w v průběhu regulace s předidentifikovanou a předtrénovanou soustavou. Tečkovaně je bez vertikálního měřítka vyznačena žádaná veličina d



Obr. 36Vývoj vah v v průběhu regulace s předidentifikovanou a předtrénovanou soustavou. Tečkovaně je bez vertikálního měřítka vyznačena žádaná veličina d

8. Závěr

V rámci této DP jsem ověřil možnost použití mikropočítače Raspberry pi (RPi) pro řízení reálných soustav. Její realizací na laboratorní úloze Hydro-pneumatická soustava (HPS) jsem vytvořil univerzální platformu (program viz příloha 1) pro individuální řízení podobných úloh pomocí RPi kdy vstupem je napěťový signál a výstupem pulsně šířkově modulovaný (PWM). Dalšími prostředky však lze dosáhnout i zpracování jiných vstupních a výstupních signálů. Knihovny pro RPi a python jsou stále zdokonalovány a i v mém případě jsem během realizace této DP, nejdříve pracoval s univerzálními moduly pro sběrnice a až s postupem času jsem po upravení modulů přímo pro RPi přešel na tyto robustnější a ozkoušené. Během práce s RPi a s vývojovou deskou Gertboard (GBd) jsem nalezl mnoho možností a uplatnění tohoto zařízení. RPi je vhodná i pro uživatele, kteří s linuxem nemají hluboké zkušenosti. K většině kroků kolem systému existují dokumentace, kde lze hledat řešení. To že je RPi oživeno operačním systémem na bázi linuxu umožňuje velmi nativní využití jazyka Python, který kromě toho že je open source, tak je velmi intuitivní a s použitím vhodných modulů vhodný pro rychlé řešení matematických úloh bez vysokých nároků na sílu hardwaru. RPi se mi stalo milou připomínkou toho, že ne vše co není vyrobeno pro čistě konzumní společnost, musí být nákladné a mnohdy komplikované. Proto jsem si dovolil v kapitole 1.2.1.3 odcitovat přeložené oficiální vyjádření nadace, která RPi vyvinula.

Soustava, kterou jsem měl pomocí RPi a jazyka python řídit (popsána v kapitole 1.3)je soustava dvou sériově vertikálně zapojených nádob s přepadem. Jedná se tedy o dvoukapacitní soustavu druhého řádu. Ačkoliv na tuto soustavu existuje fyzikální model, není její reálné řízení nijak primitivní. Jelikož se jedná i o hydraulickou soustavu, tak nelze opomíjet hydraulické ztráty, které jsou málokdy konstantní a minimálně dlouhodobě se mění zanášením kanálů a opotřebením čerpadla. A jak jsem byl také upozorněn, i výtok z nádoby dokáže změnit chování soustavy, její model se mění s výtokem do volného prostoru, nebo do kapaliny, což se může měnit s množstvím vody ve spodním zásobníku. Proto je použití adaptivního regulátoru na takovéto úloze snadno ospravedlnitelné. Už třeba jen proto, že druhá stavová veličina (výška hladiny v horní nádobě), není měřena a tím ses soustava více vymyká řízení dle předpokládaného fyzikálního modelu.

K řízení úlohy HPS s využitím RPi je použita deska GBd, nebo alespoň její část. Popis zapojení a komunikace s jednotlivými obvody je popsán společně s připojením k soustavě HPS kapitole 3. Popis rozhraní, pomocí něhož je realizována komunikace s externími zařízeními obecně je popsána už v kapitole 1.2.2

V kapitole 2 jsem uvedl pár základních kroků pro oživení RPi pro jeho využití tak jak je prováděno v této DP. Je zde také popsán návod pro instalaci modulů pro Python a pár poznámek ke spouštění Python skriptů na RPi. Žádný z těchto kroků však nebrání dalšímu využití RPi, jako např. jeho připojením k televizi a využití jako multimediálního přehrávače, nebo záznamníku. V kapitole 4 je pak rozebrána kostra skriptu uvedeném v příloze 1, který slouží jako jednoduchá aplikace se vstupem a výstupem. Pro přehlednost a funkčnost osazena funkcí P-regulátoru.

Kapitola 5 se již zabývá vlastním adaptivním neuroregulátorem (NR), který je v tomto případě realizován lineárním polynomem, kterým identifikuji soustavu a dále lineárním neuronem, který ve zpětné vazbě řídí soustavu jako adaptivní stavový regulátor. V této kapitole je rozebráno vnitřní

uspořádání regulačního obvodu a pohyb veličin. V následující kapitole 6 je řešena otázka předtrénování a identifikace, před započetím regulačního procesu. Pro ověření univerzálnosti metody je trénovací množina (žádané a měřené) reálná, avšak rozdílná od žádané při regulačním procesu.

Kapitola 7 Se pak již konečně věnuje přímo regulaci reálné soustavy a zkoumání výsledků. Nejdříve pro ukázku výsledek prosté proporcionální regulace (kapitola 7.1). Jedná se o obvod, který je pak pomocí adaptivního neuroregulátoru řízen tak jak je popsáno již v kapitole 5. P regulace dává očekávatelné výsledky s trvalou regulační odchylkou. V případě vyššího zesílení a malé změně žádané veličiny dochází k překmitu, ne však na modelových datech, pouze na trénovacích pro identifikaci soustavy. Dalším krokem (kapitola 7.2) jsem ukázal chování soustavy při řízení s adaptivním neuroregulátorem avšak bez předtrénování a předchozí identifikace. Chování bylo nejdříve nestabilní, avšak po 15-ti minutách byla regulace již výrazně lepší než prostý P-regulátor a jelikož se žádaná veličina opakovala, docházelo k predikci změny. V následující kapitole 7.3 jsem ukázal chování regulátoru, pokud je předem identifikována soustava na jiných reálných datech, než jsou pro regulaci. Výsledky jsou uspokojující, adaptivní neuroregulátor měl počáteční váhy nulové a po relativně krátké adaptaci došlo k řízení ne dokonalému, ale uspokojivému. A nakonec v kapitole 7.4 jsou výsledky pro předtrénovaný regulátorem s identifikovanou soustavou. Zhruba půlhodinové měření proběhlo bez známek nějaké nestability. Překmit v horní úvrati amplitudy se snižuje a zarážející je pouze chování v dolní úvrati amplitudy. Z pohledu do vývoje vah je patrné, že soustava není plně identifikována. Na vině je buďto malý počet cyklů, který nastává při práci s reálnými daty (nelze jednoduše adaptovat v epochách) nebo adaptivní neuroregulátor ve zpětné vazbě vnáší na vstup do identifikované soustavy příliš vysoké hodnoty, které jsou stejně nakonec omezeny fyzikálními mantinely, avšak v rámci identifikace mohou působit potíže.

Práce s neuroregulátorem na reálných úlohách tímto rozhodně nekončí, ale začíná. V rámci této DP vznikl nástroj, který umožňuje snadno přejít od simulačního řešení k reálnému, avšak výrazně tím prodloužil časové konstanty řešení jednotlivých problémů. S tímto nástrojem lze ověřovat hypotézy na reálných úlohách ale často tím místo odpovědi na otázky otevře několik dalších, na jejichž řešení se budu rád dále podílet.

9. Použitá literatura:

[1]. Laboratorní úloha "Hydro-pneumatická soustava", http://www., Laboratoř automatického řízení 111, U12110, FS, ČVUT v Praze, dostupné 1.12.2013.

[2]. Bukovsky, I., Homma, N., Smetana, L., Rodriguez, R., Mironovova M., Vrana S.,: "Quadratic Neural Unit is a Good Compromise between Linear Models and Neural Networks for Industrial Applications*", ICCI 2010 The 9th IEEE International Conference on Cognitive Informatics*, Tsinghua University, Beijing, China, July 7-9, 2010.

[3]. Bukovsky, I., Bila, J.: „Basic Classification of Nonconventional Artificial Neural Units"(In Czech), *Proceedings of Seminar Nove Hrady*, Czech Technical University in Prague, FME, ISBN: 978-80-01-03747-8, Czech Republic, 2007, pp. 76-80.

[4]. BÍLA, J. *Umělá inteligence a neuronové sítě v aplikacích*, Skriptum ČVUT, 1998.

[5]. Vyjádření Raspberry Pi Foundation dostupné z WWW: http://www.raspberrypi.org/about

[6]. Oficiální stránky Raspberry Pi dostupné z WWW: http://www.raspberrypi.org

[7]. Bukovský, ., Homma, N.: I.Dynamický backpropagation a predikce, Automatizace, ročník 53, číslo 1-2, únor 2010 <http://www.automatizace.cz/download.php?d=QXRtX0FydGljbGUscGRmX2FydCwyNzU2>

[8]. Danilo P. Mandic, Vanessa Su Lee Goh, *Complex Valued Nonlinear Adaptive Filters*: Noncircularity, Widely Linear and Neural Models, Appendix K, <http://onlinelibrary.wiley.com/doi/10.1002/9780470742624.app11/pdf>

[9]. Peter M. Benes: „Software Application for Adaptive Identification and Controller Tuning" Student's Conference STC, Faculty of Mechanical Engineering, CTU in Prague. 2013.

[10]. Ladislav Smetana: "Nonlinear Neuro-Controller for Automatic Control Laboratory System", Master's Thesis, Czech Tech. Univ. in Prague, 2008.

[11]. Sláma, Málek,Kroutil,Bedrna: „Identifikace laboratorní úlohy„Hydropneumatická soustava"", referát předmětu IDS, U12110, ČVUT FS, 2013

[12]. RPi Low-level peripherals. Popis GPIO portů dostupný na WWW: http://elinux.org/Rpi_Low-level_peripherals

[13]. Gertboard user manual, Gert Van Loo and Myra VanInwegen Revision 2.0, dostupný z WWW: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/RPi/Gertboard_UM_with_python.pdf

[14]. Datasheet MCP3002 dostupný na WWW: http://ww1.microchip.com/downloads/en/DeviceDoc/21294C.pdf

[15]. Datasheet BD6222 dostupný na WWW: http://www.kirathu.de/div/bd6222.pdf

10. Přílohy

10.1. Příloha 1, skript pro jednoduchý P-regulátor s AD převodníkem a PWM výstupem

```
#!/usr/bin/python2.7
# Python 2.7
# minimalize.py
# This is minimalistic motor-controll program for Raspberry Pi with
GertBoard (or only IC AD: MCP3002 and H-Bridge: BD6222)
# This will not work unless you have installed py-spidev and wiringpi
for Python.
#  SPI must be enabled on your system (commented here: sudo nano
/etc/modprobe.d/raspi-blacklist.conf)

from time import sleep, time
# reload spi drivers to prevent spi failures, sometimes has to be run
twice at firstime after reboot
import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
sleep(3)
import spidev
import wiringpi
import sys
board_type = sys.argv[-1]

wiringpi.wiringPiSetupGpio()                # Initialise wiringpi GPIO
wiringpi.pinMode(18,2)                      # Set up GPIO 18 to PWM
mode
wiringpi.pinMode(17,1)                      # GPIO 17 to output
wiringpi.digitalWrite(17, 0)                # port 17 off for rotation
one way
wiringpi.pwmWrite(18,0)                     # set pwm to zero initially

channel = iteration = 0                     # set ADC channel to 0 and
initial value for iteration counter
iterations = 1200                           # or some function with
time. 1200 - cca 1min for sleep 0.05
dt = 0.05                                   # time delay make sampling

def get_adc(channel):                      # read SPI data from MCP3002
chip
    if ((channel > 1) or (channel < 0)):  # Only channels 0 and 1 else
return -1
        return -1
    r = spi.xfer2([1,(2+channel)<<6,0])
    ret = ((r[1]&31) << 6) + (r[2] >> 2)
    return ret

def desired(sample):
    d = 100                                 # your desired function
will be here
    return d
```

```
def p_reg(adc_value,desired_value):         # P-reg function in range
0-1 for example
    if adc_value > desired_value:
        u_init = 0
    else :
        u_init = float(desired_value +1 - adc_value)/(1024/5)
    return u_init

### PRINT INSTRUCTION FOR CONNECTING TO GERTBOARD (for two revisions of
board, could be used similar for only simple ICs.)
print ("These are the connections for the analog input in range 0-3.3V
- motor PWM:")

if board_type == "m":
    print ("jumper connecting GPIO 8 to CSA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GPIO 17 --- MOTB")
    print ("GPIO 18 --- MOTA")
    print ("+ of external power source --- MOTOR +")
    print ("ground of external power source --- MOTOR - ")
    print ("one wire for your motor in MOTOR A screw terminal")
    print ("the other wire for your motor in MOTOR B screw terminal")

else:
    print ("jumper connecting GP11 to SCLK")
    print ("jumper connecting GP10 to MOSI")
    print ("jumper connecting GP9 to MISO")
    print ("jumper connecting GP8 to CSnA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GP17 in J2 --- MOTB (just above GP1)")
    print ("GP18 in J2 --- MOTA (just above GP4)")
    print ("+ of external power source --- MOT+ in J19")
    print ("ground of external power source --- GND (any)")
    print ("one wire for your motor in MOTA in J19")
    print ("the other wire for your motor in MOTB in J19")

raw_input("When ready hit enter.\n")

spi = spidev.SpiDev()
spi.open(0,0)         # The Gertboard ADC is on SPI channel 0 (CE0 -
aka GPIO8)

try:
    while iteration < iterations:
        adc_value = (get_adc(channel))      # GET: read ADC voltage in
range 0:1023

#_____________________________________________________________________
_____
        ###HERE IS THE CONTROL FUNCTION###
        #
        d = desired(iteration)
```

```
        u = p_reg(adc_value, d)             # this function is writen
in range 0:1
        ###OUTPUT HAST TO BE IN RANGE 1:1024###
        pwm = int((u)*1024)                 # integer in range 1:1024
        ###
        #
        wiringpi.digitalWrite(17, 0)        # motor spins one way (1
instead 0 for other way)
        wiringpi.pwmWrite(18, pwm)          # SET: send PWM value to
port 18
        sleep(dt)                           # need a delay for ssh
without is fastes
        iteration += 1                      # limit duration of program
by iterations a dt
except (KeyboardInterrupt, SystemExit):     # trap a CTRL+C keyboard
interrupt or error during scripting
    reset_ports()                           # reset ports on interrupt
reset_ports()       # reset ports on normal exit
```

10.2. Příloha 2, Skript generující signál, P-regulace a záznam hodnot

```
#!/usr/bin/python2.7
# Python 2.7
# motor_p_reg.py
# This will not work unless you have installed py-spidev and wiringpi
for Python.
#  SPI must be enabled on your system (commented here: sudo nano
/etc/modprobe.d/raspi-blacklist.conf)

from __future__ import print_function
from time import sleep

# reload spi drivers to prevent spi failures
import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
sleep(3)

import spidev
import wiringpi
import sys
from numpy import sin, pi, sign
from time import time
from matplotlib.pyplot import
show,plot,figure,subplot,xlabel,title,grid
import pickle
board_type = sys.argv[-1]

wiringpi.wiringPiSetupGpio()                # Initialise wiringpi GPIO
wiringpi.pinMode(18,2)                      # Set up GPIO 18 to PWM
mode
wiringpi.pinMode(17,1)                      # GPIO 17 to output
wiringpi.digitalWrite(17, 0)                # port 17 off for rotation
one way
```

```
wiringpi.pwmWrite(18,0)                    # set pwm to zero initially

channel = iteration = 0                    # set ADC channel to 0 and
initial value for iteration counter
iterations = 6000                          # 1200 - cca 1min for 0.05
dt
dt = 0.2
desired_value_max =  400                   # in range 0 - 1023
desired_frekvency = int(iterations/4)
y = []
u = []
d = []
t = []
#def desired_fcn(iteration, desired_frekvency, desired_value_max):
#    desired_value = 400 #sign (iteration - desired_value_max) *
desired_value_max/2) + desired_value_max/2
#    return desired_value
def desired(iteration, step):   #[sec]
    if iteration < (dt * 40 * 60):
        d = 1 * step
    elif iteration < 2 * (dt * 40 * 60):
        d = 2 * step
    elif iteration < 3 * (dt * 40 * 60):
        d = 3 * step
    elif iteration < 4 * (dt * 40 * 60):
        d = 4 * step
    elif iteration < 5 * (dt * 40 * 60):
        d = 5 * step
    elif iteration < 6 * (dt * 40 * 60):
        d = 6 * step
    elif iteration < 7 * (dt * 40 * 60):
        d = 5 * step
    elif iteration < 8 * (dt * 40 * 60):
        d = 4 * step
    elif iteration < 9 * (dt * 40 * 60):
        d = 3 * step
    elif iteration < 10 * (dt * 40 * 60):
        d = 2 * step
    else:
        d = 60
    return(d)



def p_reg(adc_value,desired_value):                    # P-reg
function in range 0-1
    if adc_value > desired_value:
        u_init = 0
    else :
        u_init = float(desired_value +1 - adc_value) / (102.4)
    return u_init

def get_adc(channel):                     # read SPI data from MCP3002
chip
    if ((channel > 1) or (channel < 0)):  # Only channels 0 and 1 else
return -1
        return -1
    r = spi.xfer2([1,(2+channel)<<6,0])
```

```
    ret = ((r[1]&31) << 6) + (r[2] >> 2)
    return ret

def reset_ports():                          # resets the ports for a
safe exit
    wiringpi.pwmWrite(18,0)                 # set pwm to zero
    wiringpi.digitalWrite(18, 0)            # ports 17 & 18 off
    wiringpi.digitalWrite(17, 0)
    wiringpi.pinMode(17,0)                  # set ports back to input
mode
    wiringpi.pinMode(18,0)

def display(iteration, adc_value, desired_value, pwm):
    print ('in iteration: ', iteration,' measured value is:
',adc_value, ', Desired value was:', desired_value, ' PWM set to: ',
pwm)
    sys.stdout.flush()


print ("These are the connections for the analog input in range 0-3.3V
- motor PWM:")

if board_type == "m":
    print ("jumper connecting GPIO 8 to CSA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GPIO 17 --- MOTB")
    print ("GPIO 18 --- MOTA")
    print ("+ of external power source --- MOTOR +")
    print ("ground of external power source --- MOTOR - ")
    print ("one wire for your motor in MOTOR A screw terminal")
    print ("the other wire for your motor in MOTOR B screw terminal")

else:
    print ("jumper connecting GP11 to SCLK")
    print ("jumper connecting GP10 to MOSI")
    print ("jumper connecting GP9 to MISO")
    print ("jumper connecting GP8 to CSnA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GP17 in J2 --- MOTB (just above GP1)")
    print ("GP18 in J2 --- MOTA (just above GP4)")
    print ("+ of external power source --- MOT+ in J19")
    print ("ground of external power source --- GND (any)")
    print ("one wire for your motor in MOTA in J19")
    print ("the other wire for your motor in MOTB in J19")

raw_input("When ready hit enter.\n")

spi = spidev.SpiDev()
spi.open(0,0)         # The Gertboard ADC is on SPI channel 0 (CE0 -
aka GPIO8)
time_init = time()
try:
```

```
    while iteration < iterations:
        adc_value = (get_adc(channel))      # GET: read ADC voltage to
control motor
        desired_value = desired(iteration,100) # make desired value
function
        u_init = p_reg(adc_value, desired_value)
        pwm = int((u_init)*1024)                        # integer in
range 1 : 1024
        wiringpi.digitalWrite(17, 0)        # motor spins one way (1
instead 0 for other way)
        wiringpi.pwmWrite(18, pwm)          # SET: send PWM value to
port 18
        y.append(adc_value)
        u.append(u_init)
        d.append(desired_value)
        t.append(time()-time_init)
        display(iteration, adc_value, desired_value, pwm)
                        # TODO: only for debug
        sleep(dt)                                   # need a delay for
ssh TODO: remove
        iteration += 1                              # limit duration of
program run to 30s [600 * 0.05]

except KeyboardInterrupt:                   # trap a CTRL+C keyboard
interrupt
    reset_ports()                           # reset ports on interrupt
reset_ports()       # reset ports on normal exit
with open('pickles/objs.pickle', 'w') as f:
    pickle.dump([u, y, d, t], f)
figure()
subplot(311)
plot(t,u),xlabel('t'),title('u'),grid()
subplot(312)
plot(t,y),xlabel('t'),title('y'),grid()
subplot(313)
plot(t,d),xlabel('t'),title('d'),grid()
show()
```

10.3. Příloha 3, skript identifikující soustavu a předtrénování NR v epochách

```
#!/usr/bin/python2.7
# Python 2.7
# offline_identifikation.py
import pickle
from matplotlib.pyplot import show,plot,figure,subplot, xlabel,
title,grid
from numpy import zeros, mean, floor, dot, median
from numpy.random import randn
with open('pickles/objs_p_reg_zes_5_delsi.pickle') as f:
    #objs_p_reg_zes_1.pickle
    #objs_p_reg_zes_100.pickle
    #objs_p_reg_zes_5.pickle
    u, y, d, t = pickle.load(f)


def resample_fcn(torsmp,n):
```

```
#    Pocet vzorku zaokrouhluje dolu, je tedy mozne ze na posledni
vzorky nebude bran zretel
    Nv = len(torsmp)
    rsmp = zeros(floor(Nv/n))
    for k in range (0,Nv/n):
        rsmp[k] = median(torsmp[(k+1)*n-n:(k+1)*n])
    return (rsmp)

def cutdata(tocut, fromsample, tosample):
    cuted = zeros(tosample-fromsample)
    cuted[:tosample-fromsample] = tocut[fromsample:tosample]
    return cuted

# parameters for Normalization

Dy=512.
Dd=512.
meany=512.
meand=512.

#--Normalization
def zy(y):
    global Dy,meany
    ynorm=(y-meany)/Dy
    return(ynorm)
def izy(ynorm): #inversion
    global Dy,meany
    y=ynorm*Dy + meany
    return(y)
def zd(d):
    global Dd,meand
    dnorm=(d-meand)/Dd
    return(dnorm)
def izd(dnorm): #inversion
    global Dd,meand
    d=dnorm*(Dd)+meand
    return(d)


# resaple from 0.2(5Hz) to 1 Hz, as median fifth (filtr)
yr = resample_fcn(y,5)
dp = resample_fcn(d,5)
yr = cutdata(yr,0,900)
dp = cutdata(dp,0,900)
Ny = len(yr)
# yn(k)=[w0,w1,w2,w3,w4,w5,w6]*[1;yr(k-1);yr(k-2);yr(k-3);dp(k-1);dp(k-
2);dp(k-3)]
# e(k) = yr(k)-yn(k)
# dw0 = -mu * de^2/dw0
# dw0 = mu * e * 1
# dw1 = mi * e * yr(k-1) ... dw6 = mi * e * dp(k-3)
# x = [1;yr(k-1);yr(k-2);yr(k-3);dp(k-1);dp(k-2);dp(k-3)]
# dw = mu e x
nw = 7
w = randn(nw)/nw
x = zeros(nw)
e = zeros(Ny)
yn = zeros(Ny)
```

```
yr = zy(yr)
dp = zd(dp)

mu = 0.1
epochs = 20
wall = zeros((epochs,nw))
eall = zeros(epochs)
epoch = 0
yn[0:3]=yr[0:3]
ww=zeros((Ny,nw))
# ref model
##---reference model setup
dt=float(1)
Td=1   # input delay
rmtau1=2 # [sec]
rmtau2=2 # [sec]
rmSu1=1
rmSu2=1
rmh=dt/rmtau1*(rmSu1*dp[2])
yref=zeros(Ny)
yref[:3]=dp[:3]

def ref_model_fcn(k,dp,yref):
    global dt, Td, rmtau1, rmtau2, rmSu1, rmSu2, rmh
    rmh=rmh+dt/rmtau1*(rmSu1*dp[k-2]-rmh) # d...desired, yref...ref.
mod. out
    yref[k]=yref[k-1]+dt/rmtau2*(rmSu2*rmh-yref[k-1])
    yref[:10]=dp[:10]
    return(yref)

try:
    while epoch < epochs:  # ident loop
        for k in range(3,Ny):
            x[0:7]=[1,yr[k-1],yr[k-2],yr[k-3],dp[k-1],dp[k-2],dp[k-3]]
            yn[k]=dot(w,x)  #yn(k)
            e[k] = yr[k] - yn[k] #len = 600
            dw = mu * e[k] *x
            w = w + dw
            ww[k,:]=w
            if epoch == epochs - 1:
                yref = ref_model_fcn(k,dp,yref)
        wall[epoch,:]=w
        eall[epoch] = sum(e*e)
        epoch += 1

#    for k in range(3,Ny):  # controller loop
#        rmh=rmh+dt/rmtau1*(rmSu1*dp[k-2]-rmh) # d...desired,
yref...ref. mod. out
#        yrm[-1]=yrm[-2]+dt/rmtau2*(rmSu2*rmh-yrm[-2])
#        yrmall[k]=yrm[-1]


except (KeyboardInterrupt, SystemExit):
    print('exception')

with open('pickles/objs_with_weigths.pickle', 'w') as f:
    pickle.dump([u, y, d, t, yr, dp, w], f)
```

```
figure()
subplot (411)
plot (wall),xlabel('epochs'),title('weight in epochs'),grid()
subplot (412)
plot (ww),xlabel('epochs'),title('weight in one epoch'),grid()
subplot (413)
plot(eall),xlabel('epochs'),title('error in epochs'),grid()
subplot(414)
plot((yn),'g')
plot((yr))
plot((yref))
plot((dp),'k'),xlabel('k'),title('yr, yn, dp, yref'),grid()

show()
#_ = raw_input("Press [enter] to continue.") # wait for input from the
user
#close()    # close the figure to show the next one.


#figure()
#subplot(411)
#plot(t,u),xlabel('t'),title('u in range 0:1 (more than 1 is still
1)'),grid()
#subplot(412)
#plot(t,y),xlabel('t'),title('y'),grid()
#subplot(413)
#plot(t,d),xlabel('t'),title('d'),grid()
#subplot(414)
#plot(yr),xlabel('k'),title('yr'),grid()
#show()
```

10.4. Příloha 4, skript pro neuroregulaci s provedeným předtrénováním

```
#!/usr/bin/python2.7
# Python 2.7
# NR_with_init_weights.py
# This will not work unless you have installed py-spidev and wiringpi
for Python.
#  SPI must be enabled on your system (commented here: sudo nano
/etc/modprobe.d/raspi-blacklist.conf)

from __future__ import print_function
from time import sleep, time

# reload spi drivers to prevent spi failures
import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
sleep(3)

import spidev
import wiringpi
import sys
```

```
from numpy import
sign,sin,arange,pi,loadtxt,ones,mean,std,array,dot,ones,zeros,savetxt
from numpy.random import randn
from scipy import ceil
from matplotlib.pyplot import
semilogy,show,plot,figure,subplot,xlabel,title,grid
import pickle
board_type = sys.argv[-1]

wiringpi.wiringPiSetupGpio()                # Initialise wiringpi GPIO
wiringpi.pinMode(18,2)                      # Set up GPIO 18 to PWM
mode
wiringpi.pinMode(17,1)                      # GPIO 17 to output
wiringpi.digitalWrite(17, 0)                # port 17 off for rotation
one way
wiringpi.pwmWrite(18,0)                     # set pwm to zero initially

channel = iteration = 0                     # set ADC channel to 0 and
initial value for iteration counter
iterations = 1200                           # 1200 - cca 1min for sleep
0.05

pwm_all = []
t_measure = []

def p_reg(adc_value,desired_value):                    # P-reg
function in range 0-1
    if adc_value > desired_value:
        u_init = 0
    else :
        u_init = float(desired_value +1 - adc_value)/(1024/5)
    return u_init

def get_adc(channel):                      # read SPI data from MCP3002
chip
    if ((channel > 1) or (channel < 0)):  # Only channels 0 and 1 else
return -1
        return -1
    r = spi.xfer2([1,(2+channel)<<6,0])
    ret = ((r[1]&31) << 6) + (r[2] >> 2)
    return ret

def reset_ports():                          # resets the ports for a
safe exit
    wiringpi.pwmWrite(18,0)                 # set pwm to zero
    wiringpi.digitalWrite(18, 0)            # ports 17 & 18 off
    wiringpi.digitalWrite(17, 0)
    wiringpi.pinMode(17,0)                  # set ports back to input
mode
    wiringpi.pinMode(18,0)

def display(iteration, adc_value, desired_value, d, q, pwm):
    print ('in teration:', iteration,' measured value is: ',adc_value,
', Desired value was:', d, 'Now is it: ', desired_value,' by q=:', q, '
PWM set to: ', pwm)
    sys.stdout.flush()

dt=0.2  # rychlost vzorkovani Maliny (dt=const!) [sec]
```

```
def desired(dt,T,Tstop):   #[sec]
    k=arange(0,Tstop/dt) # time in samples
    t=k*dt
    d=sin((pi/T)*k*dt+pi)
    d=(sign(d)+1)/2.
    return(t,d)

#def desired(iteration, step):   #[sec]
#    if iteration < (dt * 2 * 60):
#        d = 1 * step
#    elif iteration < 2 * (dt * 2 * 60):
#        d = 2 * step
#    elif iteration < 3 * (dt * 2 * 60):
#        d = 3 * step
#    elif iteration < 4 * (dt * 2 * 60):
#        d = 4 * step
#    elif iteration < 5 * (dt * 2 * 60):
#        d = 5 * step
#    elif iteration < 6 * (dt * 2 * 60):
#        d = 6 * step
#    elif iteration < 7 * (dt * 2 * 60):
#        d = 5 * step
#    elif iteration < 8 * (dt * 2 * 60):
#        d = 4 * step
#    elif iteration < 9 * (dt * 40 * 60):
#        d = 3 * step
#    elif iteration < 10 * (dt * 40 * 60):
#        d = 2 * step
#    else:
#        d = 60
#    return(d)
#----------
maxy=1024
miny=1
maxd=1023
mind=0
Dy=maxy-miny
Dd=maxd-mind
meany=miny+(maxy-miny)/2.
meand=mind+(maxd-mind)/2.

#--
def zy(y):
    global maxy,miny,meany
    ynorm=(y-meany)/float((maxy-miny))
    return(ynorm)
def izy(ynorm):
    global maxy,miny,meany
    y=ynorm*(maxy-miny)+meany
    return(y)
def zd(d):
    global maxd,mind,meand
    dnorm=(d-meand)/float((maxd-mind))
    return(dnorm)
def izd(dnorm):
    global maxd,mind,meand
    d=dnorm*(maxd-mind)+meand
    return(d)
```

```
dn=5  # prevzorkovani pro model URO (pomale vzorkovani modelu URO -
vlastni vzorkovaci perioda)

t,d=desired(dt,60,iterations*dt)
d=zd((d*350)+50)
#d=zeros(iterations*dt)
#t=zeros(iterations*dt)
#for kt in range(0,int(iterations*dt)):
#   d[kt]=(desired(kt, 100))
#   t[kt]=(kt/dt)

##plot(t,d)
##show()
##---real system model
tau1=6 # [sec]
tau2=12 # [sec]
Td=1 # input delay [sec]
Su1=1.5
Su2=1.3
#-real system model inits
NTd=int(min(ceil(Td/dt),6))
Nstart=max(3*dn,NTd)
yr=zeros(Nstart)
h=0
##---reference model setup
rmtau1=2# [sec]
rmtau2=2 # [sec]
rmTd=Td+dt*dn # input delay [sec]
rmSu1=1
rmSu2=1
#-reference model inits
rmNTd=int(ceil(rmTd/dt))
dref=zeros(3*dn+1)
rmh=0
#-- LNU model setup
muw=1
nw=1+3+3
Njenidentifikace=3*dn #adaptuj hned
#Njenidentifikace=1000/dt # [sec/fast sampling]
#---control setups
ro=1
muro=0.4
nxi=1+2+3
muv=3
q=0      # inic. cond. of adaptive cont.
#---
N=len(d)
ydall=zeros(N)# just for plotting

w=[-0.02974774,0.6824019,0.2877426,-
0.10420276,0.01743145,0.02326079,0.08257557]
yr=zeros(3*dn+1)
yn=zeros(3*dn+1)
yrall=zeros(N)
ynall=zeros(N)
nx=nw
x=ones(nx)
```

```
u=d[:3*dn+1].copy()
xi=zeros((nxi,3*dn+1))
xi[0,:]=1
nv=nxi  # for LNU
v=[zy(307.44696567),zy(139.80996209),zy(140.17641411),zy(-
69.17941685),zy(-69.37652568),zy(-68.51797149)]
#v=randn(nv)/nv
dxdv=zeros((nx,nv))
drefall=d.copy()
dxdro=zeros(nx)
wall=zeros((N,nw))
vall=zeros((N,nv))
roall=zeros(N)



print ("These are the connections for the analog input in range 0-3.3V
- motor PWM:")

if board_type == "m":
    print ("jumper connecting GPIO 8 to CSA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GPIO 17 --- MOTB")
    print ("GPIO 18 --- MOTA")
    print ("+ of external power source --- MOTOR +")
    print ("ground of external power source --- MOTOR - ")
    print ("one wire for your motor in MOTOR A screw terminal")
    print ("the other wire for your motor in MOTOR B screw terminal")

else:
    print ("jumper connecting GP11 to SCLK")
    print ("jumper connecting GP10 to MOSI")
    print ("jumper connecting GP9 to MISO")
    print ("jumper connecting GP8 to CSnA")
    print ("Analog input connections:")
    print ("  connect analog input to AD%d" % channel);
    print ("  connect grounding to GND")
    print ("Motor connections:")
    print ("GP17 in J2 --- MOTB (just above GP1)")
    print ("GP18 in J2 --- MOTA (just above GP4)")
    print ("+ of external power source --- MOT+ in J19")
    print ("ground of external power source --- GND (any)")
    print ("one wire for your motor in MOTA in J19")
    print ("the other wire for your motor in MOTB in J19")

raw_input("When ready hit enter.\n")

spi = spidev.SpiDev()
spi.open(0,0)         # The Gertboard ADC is on SPI channel 0 (CE0 -
aka GPIO8)
time_init = time()
try:
    while iteration < iterations:
        k=iteration
```

```
        adc_value = (get_adc(channel))      # GET: read ADC voltage to
control motor
        adc_value = zy(adc_value)
        if k < NTd:
            yr[0:3*dn]=yr[1:]
            yr[-1]=adc_value
            u[:3*dn]=u[1:]
            u[-1]=ro*(d[k]-yr[-1])
        else:
            u[:3*dn]=u[1:]
            if k < Njenidentifikace:
                u[-1]=d[k]
            else:
                u[-1]=d[k]-zd(ro*q)

            yr[0:3*dn]=yr[1:]

            #h=h+dt/tau1*(Su1*u[-1-NTd]-h)
            #yr[-1]=yr[-2]+dt/tau2*(Su2*h-yr[-2])
            yr[-1]=adc_value
            yrall[k]=yr[-1]  # just for plotting
        #reference model
            dref[0:3*dn]=dref[1:]
            rmh=rmh+dt/rmtau1*(rmSu1*d[k-rmNTd]-rmh) # d...desired,
dref...ref. mod. out
            dref[-1]=dref[-2]+dt/rmtau2*(rmSu2*rmh-dref[-2])
            drefall[k]=dref[-1]
        #real system ident
            if k > 3*dn:  # bezi na rychlem vzorkovani
                # plant ident
                yn[0:3*dn]=yn[1:]
                x[1]=(yr[-1-dn])  #yr(k-1)
                x[2]=(yr[-1-2*dn])
                x[3]=(yr[-1-3*dn])
        ##        x[1]=zy(yn[-1-dn])  #yr(k-1)
        ##        x[2]=zy(yn[-1-2*dn])
        ##        x[3]=zy(yn[-1-3*dn])
                x[4]=zd(u[-1-dn])
                x[5]=zd(u[-1-2*dn])
                x[6]=zd(u[-1-3*dn])
                yn[-1]=(dot(w,x))  #yn(k)
                ynall[k]=yn[-1]
                e=yr[-1]-yn[-1]
                dyndw=x#*Dy
                dw=muw/(1.+sum(x**2))*e*dyndw
                w=w+dw
                muw=muw*0.999
                muw=max(muw,0.05)
                wall[k,:]=w
        ### controller
            if k >= Njenidentifikace:
                xi[:,:3*dn]=xi[:,1:]
                xi[1,-1]=(yr[-1]-d[k])
                xi[2,-1]=(yr[-1-dn]-d[k-dn])
                xi[3,-1]=(yr[-1])
                xi[4,-1]=(yr[-1-1*dn])
                xi[5,-1]=(yr[-1-2*dn])
                q=izd(dot(v,xi[:,-1]))
```

```
                dxdv[4,:]=-xi[:,-1-dn]
                dxdv[5,:]=-xi[:,-1-2*dn]
                dxdv[6,:]=-xi[:,-1-3*dn]
                dyndv=dot(w,dxdv)
                dv=muv/(1+sum(x**2)+sum(xi[:,-1]**2))*(dref[-1]-yr[-
1])*dyndv
                v=v+dv
                vall[k,:]=v
                muv=muv*0.9999
                muv=max(muv,0.2)
                dxdro[4]=(d[k-dn]-yr[-1-dn])
                dxdro[5]=(d[k-2*dn]-yr[-1-2*dn])
                dxdro[6]=(d[k-3*dn]-yr[-1-3*dn])
                dro=muro/(1+sum(x**2)+sum(xi[:,-1]**2))*(dref[-1]-yr[-
1])*dot(w,dxdro)
                ro=ro+dro
                roall[k]=ro
                muro=muro*0.9999
                muro=max(muro,0.1)
        #print (adc_value, izd(u[-1]))
        u_init = p_reg(adc_value, izd(u[-1]))
        pwm = int((u_init)*1024)                        # integer in
range 1 : 1024
        wiringpi.digitalWrite(17, 0)        # motor spins one way (1
instead 0 for other way)
        wiringpi.pwmWrite(18, pwm)          # SET: send PWM value to
port 18
        pwm_all.append(pwm)
        t_measure.append(time()-time_init)
        display(iteration, izy(adc_value), izd(u[-1]),
izd(d[iteration]), q, pwm)                              # TODO: only
for debug
        sleep(dt)                                   # need a delay for
ssh TODO: remove
        iteration += 1                              # limit duration of
program run to 30s [600 * 0.05]

except (KeyboardInterrupt, SystemExit):                   # trap a
CTRL+C keyboard interrupt or error during scripting
    reset_ports()                           # reset ports on interrupt
reset_ports()       # reset ports on normal exit

with open('pickles/objs_outputs', 'w') as f:
    pickle.dump([yrall, drefall, d, t, wall, vall, roall], f)

subplot(512)
plot(t,d)
plot(t,yrall,'m'),title('with adaptilve LNU as a q-controller')
##plot(t,ynall,'g')
plot(t,drefall,'k')
subplot(513)
plot(t,wall),title('w')
subplot(514)
plot(t,vall),title('v')
subplot(515)
plot(t,roall,'k'),title('ro')
show()
```