UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

ŘÍDICÍ JEDNOTKA HYDRAULICKO-PNEUMATICKÉ LABORATORNÍ SOUSTAVY

Petr Linhart

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Linhart**
Osobní číslo:	**I14048**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Název tématu:	**Řídicí jednotka hydraulicko-pneumatické laboratorní soustavy**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cíl: Navrhnout a realizovat řídicí jednotku pro připojení laboratorní soustavy k počítači pomocí platformy Arduino.
Obsah teoretické části: Jednočipové mikropočítače - HW a programování, experimentální identifikace regulace, číslicový regulátor.
Obsah implementační části: Návrh řídicí jednotky, vytvoření programu pro mikropočítač a pro PC, identifikace soustavy, návrh číslicového regulátoru, regulační experimenty.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

DRÁBEK, O., MACHÁČEK, J. Experimentální identifikace. Vysoká škola chemicko-technologická v Pardubicích, 1987.
BALÁŤE, J. Automatické řízení. 2 vyd. Praha: BEN - technická literatura, 2004.
DUŠEK, F., HONC, D. Matlab a Simulink, Úvod do používání. skriptum, Univerzita Pardubice, vydání první, Pardubice, 2005.

Vedoucí bakalářské práce: **Ing. Daniel Honc, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **7. prosince 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 15. prosince 2016

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 12. 05. 2017

Petr Linhart

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Danielovi Honcovi, Ph.D. za vedení práce, ochotu, rady a připomínky ke zpracování. Dále bych rád poděkoval rodině, přítelkyni a přátelům za podporu po celou dobu studia.

V Pardubicích dne 12. 05. 2017

Petr Linhart

ANOTACE

Tato bakalářská práce se zabývá řízením hydraulicko-pneumatické laboratorní soustavy pomocí platformy Arduino. V práci je popsán návrh řídící jednotky, číslicového regulátoru a vytvoření programu pro mikropočítač a PC pomocí softwaru MATLAB, ve kterém bylo vytvořeno i grafické rozhraní pro nastavování parametrů regulátoru a vyobrazování průběhů. Dále je provedena identifikace soustavy a regulační experimenty s touto soustavou.

KLÍČOVÁ SLOVA

Jednočipové mikropočítače, řídící jednotka, regulace, regulátor, identifikace soustavy, Arduino, Matlab, hydraulicko-pneumatická soustava

TITLE

CONTROL UNIT OF HYDRAULIC-PNEUMATIC LABORATORY SYSTEM.

ANNOTATION

This bachelor thesis deals with the control of hydraulic-pneumatic laboratory systém using Arduino platform. Thesis focused on design of a control unit, digital controller and create a program for a microcomputer and PC using Matlabe software, in witch a graphical interface for setting the controller parametres and viewing of data was created. The thesis describes the identification and control systems experiments with this system as well.

KEYWORDS

Microcontroller, Control unit, Control, Controller, System identification, Arduino, Matlab, Hydraulic.pneumatic system

OBSAH

SEZNAM ZKRATEK A ZNAČEK

A	Akční člen
A/D	Analog/Digital
ALU	Arithmetical logic unit
CPU	Central processor unit
D/A	Digital/Analog
EEPROM	Electrically EPROM
EPROM	Erasable PROM
F	Filtr
GUI	Graphical user interface
PC	Počítač processor unit
PID	Proporcionálně integračně derivační
PROM	Programmable read only memory
PSD	Proporcionálně sumačně derivační
PWM	Pulse width modulation
R	Řídicí systém
RAM	Random access memory
ROM	Read only memory
S	Soustava
Sn	Snímač
SRAM	Static random access memory
USB	Universal seriál bus

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

Θ	normalizované dopravní zpoždění
ω_k	kritická úhlová frekvence
A_S	plocha nad křivkou
C	kapacita, F
e	regulační odchylka
f_C	mezní frekvence, Hz
K	zesílení
k	krok
n	šum
q	parametr
R	elektrický odpor, Ω
r_0	zesilující složka
r_1	integrační složka
r_2	derivační složka
r_k	kritické zesílení
S_L	model levého systému
S_P	model pravého systému
t	čas, s
T	perioda vzorkování, s
T_D	derivační časová konstanta, s
T_I	integrační časová konstanta, s
T_k	perioda ustálených kmitů, s
T_U	doba průtahu, s
T_N	doba náběhu, s
T_{Ust}	doba ustálení přechodového děje
T_Σ	souhrnná časová konstanta
U	elektrické napětí, V
u	akční veličina
w	žádaná hodnota
y	výstupní veličina

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

V dnešní době se s pojmy jako řízení či regulace setkáváme stále častěji. Může to být v kotelnách, průmyslových halách, tepelných výměnících, obytných domech, automobilech atd. Řídit nebo regulovat lze celou škálu systémů a to za pomocí různých prostředků a metod, které se většinou volí s ohledem na znalosti teorie, požadavky řízení, ale i finanční možnosti realizace.

V této práci se zabývám řízením laboratorní soustavy pomocí platformy Arduino propojenou se softwarem MATLAB. V teoretické části seznamuji čtenáře s jednočipovými mikropočítači a popisuji jejich hardware, programování a využití. Dále se pak zabývám různými typy regulací systému, regulačními obvody, jednotlivými složky PID regulátoru a některými metodami nastavování PID regulátoru. V praktické části se zabývám samotným návrhem a realizací řídící jednotky pro připojení laboratorní soustavy k počítači pomocí platformy Arduino. Vytvářím program pro mikropočítač a PC pomocí softwaru MATLAB, ve kterém vytvářím i grafické rozhraní pro nastavování parametrů regulátoru a vyobrazování průběhů. Dále identifikuji soustavu, optimalizačními metodami určuji PID regulátor a řídím hydraulicko-pneumatickou laboratorní soustavu.

1 TEORETICKÁ ČÁST

1.1 Jednočipové mikropočítače

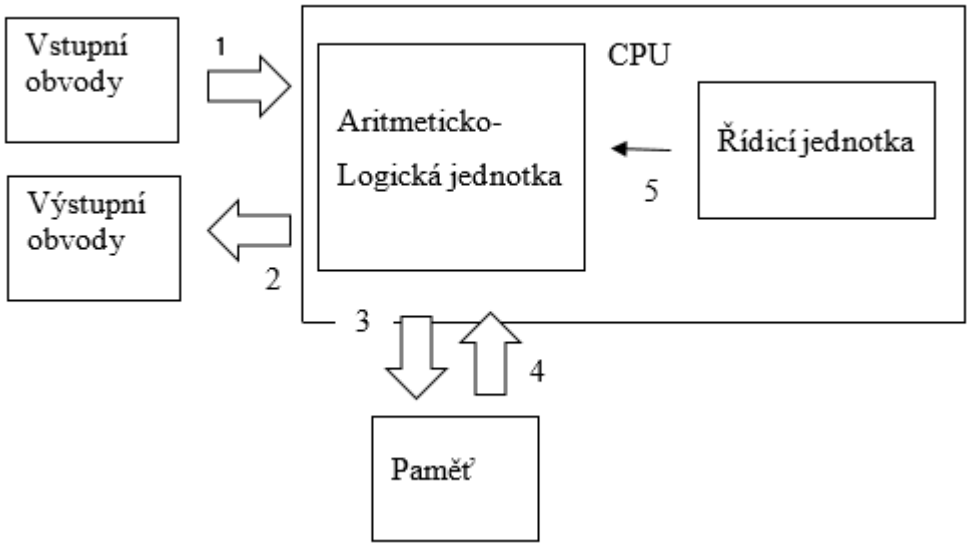
1.1.1 Úvodní seznámení

Jednočipový mikropočítač je mikroprocesor a veškeré potřebné obvody, které potřebuje pro obsažení celé aplikace, integrovány do jednoho čipu. Podle konkrétního typu využití se podpůrné obvody v jednotlivých typech mikropočítačů liší. Aplikace mikropočítačů je prakticky neomezená. Využívají se v běžných elektronických přístrojích jako je kalkulačka, domácí kino, myčka na nádobí nebo rádiu, ale i v obtížnějších oblastech využití leteckého, automobilového či zbrojního průmyslu. Mikropočítač se volí tak, aby obsáhl celou úlohu a byl zároveň co nejjednodušší z důvodu snížení ceny. (Pinker, 2004)

1.1.2 Hardware

CPU

Hardware jednočipového mikropočítače tvoří mnoho obvodů. Tím hlavním a nejdůležitějším je mikroprocesor neboli CPU, který se bere za mozek a srdce celého systému. Je tvořen ALU a řídící jednotkou. ALU provádí aritmetické a logické operace s daty. Řídicí jednotka určuje, z jakého zdroje ALU bere data, jakou operaci s nimi provede a kam uloží výsledek. Blokové schéma mikroprocesoru s prováděnými akcemi je na obrázku 1.1. (Havlíček, 2016)



Obrázek 1.1 – Blokové schéma mikroprocesoru

Význam prováděných akcí je následující:

1 … Přenesení dat ze vstupních obvodů

2 … Uložení výsledků do výstupních obvodů

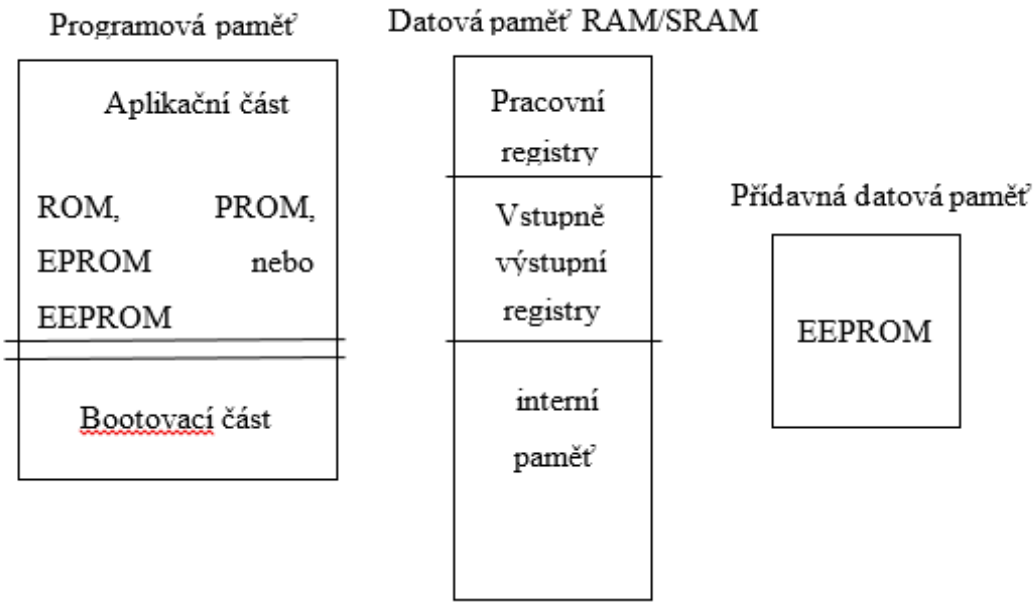
3 … Uložení výsledku do paměti

4 … Přenesení dat z paměti

5 … Řídicí signál určující zdroj dat, uložení výsledku a operaci s daty

Paměť

Mikropočítač obsahuje několik typů pamětí. První je operační (datová) paměť, která je většinou typu RAM případně SRAM, kde je vyhraněný prostor pro data, pracovní registry a vstupně-výstupní registry. Paměti typu ROM, PROM, EPROM nebo EEPROM realizují programovou (pevnou) paměť. Programová paměť je vyhrazena v mikropočítači pro samotný program. Může i obsahovat přídavnou datovou paměť typu EEPROM pro uchování proměnných po restartu mikropočítače. Rozdělení jednotlivých pamětí je na obrázku 1.2.



Obrázek 1.2 – Rozdělení paměti

Vstupně/výstupní brány a rozhraní

Vstupně/výstupní brány a rozhraní (sériové nebo paralelní) jsou pro řízení periferií. Dovolují připojení tlačítek, maticových a jiných klávesnic nebo zobrazovačů (např. segmentových, maticových tzv. bodových a LED zobrazovačů).

Časovače/čítače

Časovače/čítače vykonávají různé funkce. Většinou mikropočítač obsahuje více časovačů/čítačů, které se liší funkcemi a tím, zda čítají vzestupně, sestupně nebo obojí. Čítají externí události a interní hodiny. Některé obsahují komparační a zachytávací jednotky. Často se využívají jako frekvenční generátory PWM signálu, kterým se řídí buď jas, nebo s pomocí dolní propusti digitálně nahrazuje analogoví signál.

A/D a D/A převodníky

A/D a D/A převodníky nám převádí analogoví signál na digitální (např. proud na číslo) a naopak (např. TTL úrovně na napětí). Přesnost převodu je dána rozlišením (počet bitů pro vyjádření hodnoty). Obsahují i komparátor, který se dá použít pro porovnávání hodnot u aplikací, kde je to žádoucí a potřebné.

1.1.3 Programování

Programovat mikropočítač je možné různými způsoby. Mikropočítač pracuje se strojovým kódem (posloupnost binárních čísel). Proto byli vytvořeny různé programovací jazyky, které se převádí (kompilují) do strojového kódu. Jazyk, ve kterém se mikropočítač programuje, se volí podle nároků aplikace. Pokud je potřeba velmi úsporný a rychlí kód, tak je vhodné pro programování využít nízkoúrovňový jazyk, kterým je například Assembler. Assembler je nejnižší programovací jazyk a program je zde popsán v jednotlivých povelech. To umožnuje úplnou kontrolu a plné využití mikropočítače. Ve vyšších programovacích jazycích, jako je například C, je možné většinu aplikací naprogramovat daleko pohodlněji. Poskytují možnost využít předem naprogramované funkce a lehce se přenáší mezi platformami, ale může zde být značné omezení v maličkostech (jazyk nemusí obsahovat některé konkrétní funkce). Další menší nevýhodou vyšších programovacích jazyků je větší velikost programu, který v něm vytváříme pro mikropočítač, takže musíme více hledět na paměť mikropočítače. (Hankovec, 2002)

1.2 Regulační obvody

Regulační obvody neboli regulátory jsou dvojího typu. První typ pracuje se stejnou veličinou jako řízený systém, tedy neprovádí se zde žádný převod veličin, a proto se nazývá bez převodu veličin. Tento typ se využíval spíše v minulosti, např. u parního stroje byl Wattův regulátor otáček, ale dnes se využívá jen zcela výjimečně. V dnešní době se nejběžněji používají

regulátory s převodem veličin. Nejčastější zastoupení mají elektrické regulátory. Existují i jiné prostředky např. pneumatické a hydraulické, které mají stále své využití v průmyslu, z důvodu výbušného prostředí, nebo z důvodu ceny, kde elektrické ekvivalenty jsou podstatně dražší.

1.2.1 Regulace systému

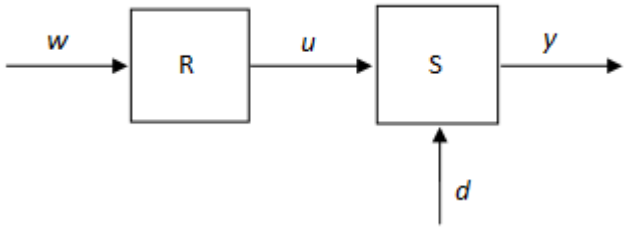
Regulace systému je proces automatického udržování parametru procesu na požadované hodnotě nebo hodnotách buď konstantních, nebo měnící se definovaným způsobem v čase. Princip regulace spočívá v tom, že regulátor pomocí akční veličiny hýbe s regulovanou veličinou tak, aby byla co nejvíce konstantní a blízko požadované hodnotě. Regulace je realizována obvodem, který může být dvojího typu:

- Otevřený
- Uzavřený (zpětnovazební)

Otevřené řízení

Otevřené řízení, které je také označováno jako ovládání, se dá použít pouze v případě, pokud jsou vnější vlivy neustále konstantní, nebo se mění dle známých průběhů. Dále musíme být zcela obeznámeni s celou soustavou. Regulátor nemá žádnou zpětnou vazbu s regulovanou soustavou, a tak neví, jak soustava zareagovala na zásah. Tyto reakce známe z předchozích zkušeností nebo matematických výpočtů. Takováto znalost vnějších vlivů a celé soustavy je v praxi skoro nemožná a vzácná. (Cvejn, 2015)

Působení vstupní, poruchové a výstupní veličiny jsou znázorněny ve schématu na obrázku 1.3 pomocí šipek. Tyto vazby jsou často označovány jako signály, protože vzájemné působení lze chápat jako přenos informace.



Obrázek 1.3 – Otevřený obvod řízení

Význam veličin a symbolů je následující:

w … požadovaná hodnota

R … řídicí systém

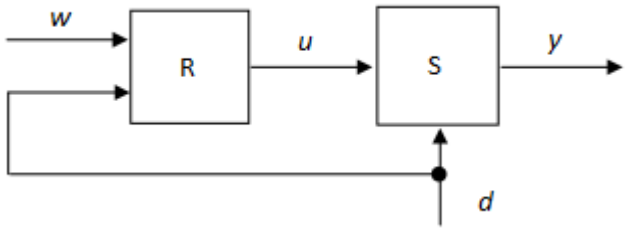
u … akční veličina (výstup řídicího systému)

S … řízený systém

d … vnější vlivy (poruchy)

y … výstupní veličina (výsledek)

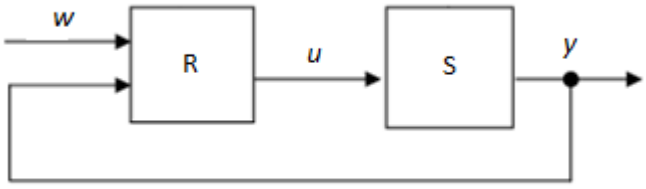
Schéma lze rozšířit o měření poruch v případě působení vnějších vlivů. Rozšířený obvod je znázorněn na obrázku 1.4.



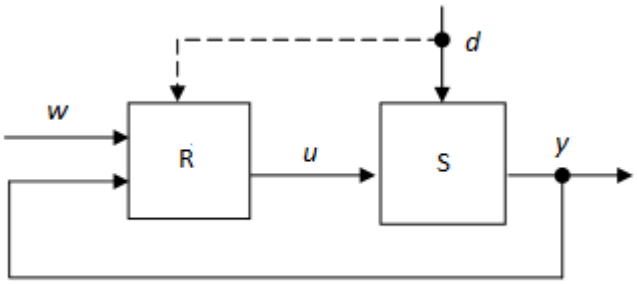
Obrázek 1.4 – Otevřený obvod řízení s měřením poruchy

Zpětnovazební řízení

Zpětnovazební řízení oproti otevřenému řízení má zpětnou vazbu, která nám zajišťuje dosažení požadovaných hodnot i bez znalosti vnějších vlivů a parametrů systému. Tento typ řízení je znázorněn na obrázku 1.5. Zpětnovazební řízení je možné zdokonalit měřením a částečnou kompenzací poruchy jako je na obrázku 1.6.



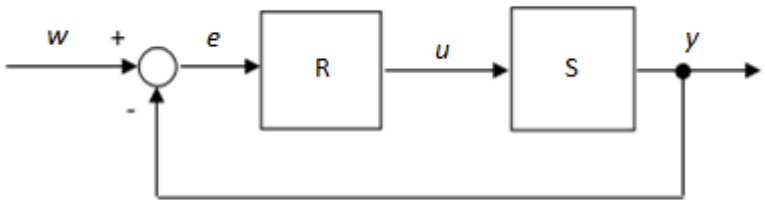
Obrázek 1.5 – Zpětnovazební řízení



Obrázek 1.6 - Zpětnovazební řízení s měřením a kompenzací vnějších vlivů

Do řídicího systému daleko častěji vstupuje regulační odchylka *e*, která je získána porovnáním měřené regulované veličiny s referenční hodnotou. Akční veličina se mění tak, aby

se regulační odchylka $e(t) = w(t) - y(t)$ co nejvíce blížila k nule. Tato možnost je znázorněna na obrázku 1.7.



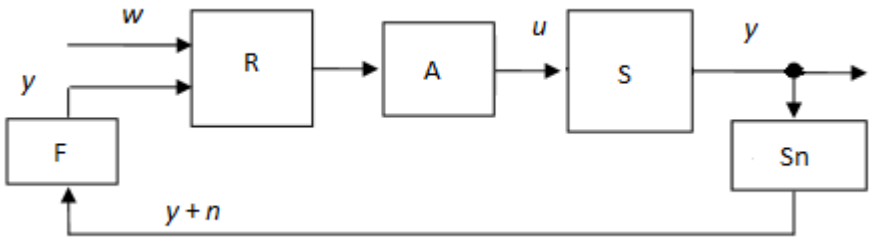
Obrázek 1.7 - Zpětnovazební řízní s regulační odchylkou

1.2.2 Elektronické řídicí systémy

Elektronické řídicí systémy jsou děleny na analogové a číslicové. Analogové se využívají v největším množství v podobě jednoduchého analogového regulátoru. Číslicové mají podobu řídících počítačů a kompaktních číslicových regulátorů. Dále jsou děleny na spojité a nespojité. Spojité regulátory mají vstupní i výstupní signál ve formě spojité funkce času a v jakýkoliv časový okamžik se může měnit. U nespojitých se vstupní, výstupní nebo obojí mění s časem nespojitě. Většinou dochází skoková změna z jedné hodnoty na jinou v určitém časovém okamžiku a ta se udržuje konstantní do další skokové změny.

Analogový elektronický regulátor

Analogový elektronický regulátor musí mít obstarán převod veličin na straně výstupu i na straně vstupu do systému. U výstupu systému se o převod stará třeba snímač (např. tlaku), který v podobě napěťového nebo proudového signálu podává informaci o hodnotě výstupu v danou chvíli. Na vstupu převod provádí akční člen, který signál ze senzoru převede na akční veličinu příslušného typu (např. napětí-průtok). Rušivé vlivy, které zkreslují signál během přenosu k řídicímu systému, mohou velmi ovlivnit činnost regulátoru. Tato situace se většinou řeší přidáním filtru, který má za úkol potlačit tuto šumovou složku signálu, před vstupem signálu do regulátoru. Toto řešení je zobrazeno na obrázku 1.8.



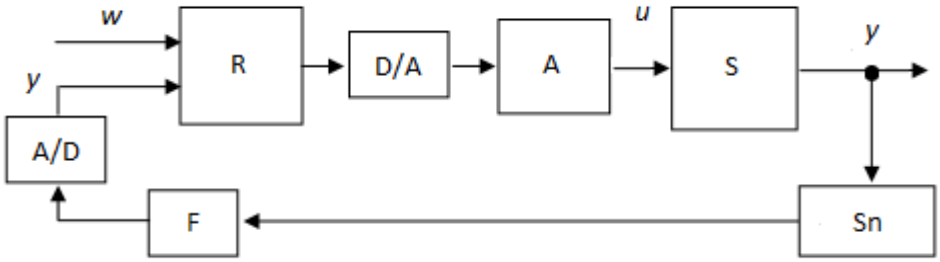
Obrázek 1.8 - Analogový regulační obvod

Význam veličin a symbolů je následující:

w … požadovaná hodnota
R … řídicí systém
u … akční veličina (výstup řídicího systému)
S … řízený systém
y … výstupní veličina (výsledek)
A … akční člen
Sn … snímač
F … filtr
n … šum

Číslicový regulační obvod

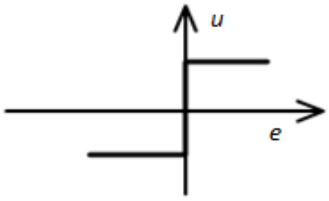
Číslicový regulační obvod operuje s posloupností hodnot, kterou získává pomocí vzorkování výstupní veličiny v určitých intervalech, ze kterých určuje akční zásah a výsledek přivádí na vstup systému. Vstupní a výstupní veličiny musí zůstat ve formě spojitých hodnot. Tyto náležitosti však znemožňují použití u rychlých procesů, kde jsou časové konstanty řádově v milisekundách, protože je zapotřebí mít čas na vzorkování a převod. Vzorkování a převod analogového signálu na digitální zajišťuje A/D převodník. D/A převodník převádí digitální signály na analogové. Tyto převodníky mají však omezenou přesnost, která je dána kvantizační chybou. Tato chyba vzniká z důvodu omezení počtu hodnot, kterých může digitální signál nabývat (počet bitů pro vyjádření hodnoty). Číslicový regulátor je zobrazen na obrázku 1.9.



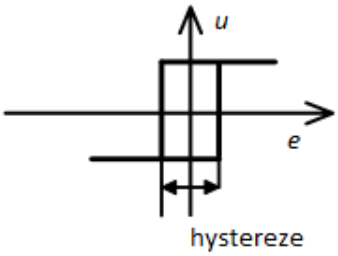
Obrázek 1.9 - Číslicový regulační obvod

Nespojitý regulátor

Nespojité regulátory se převážně vyrábí dvojího typu. Dvoupolohový, který dokáže spínat pouze kladnou nebo zápornou hodnotu akční veličiny, a třípolohový, který může spínat zápornou i kladnou hodnotu akční veličiny. Prakticky jsou zrealizovány komparačním obvodem a spínacím prvkem. Aby se spínací prvek při častém spínání rychle neopotřeboval, tak se přidává umělá hystereze, která zvyšuje nepřesnost (způsobuje větší rozkmit regulované veličiny), ale zabraňuje rychlému spínání.



Obrázek 1.10 - Statická charakteristika dvoupolohového regulátoru bez hystereze



Obrázek 1.11 - Statická charakteristika dvoupolohového regulátoru s hysterezí

Spojitý regulátor

Spojité regulátory se nejčastěji vyskytují ve formě PID regulátoru, který vyhodnocuje akční zásah z regulační odchylky, jejího integrálu a její derivace. Tento druh regulátoru je velmi oblíbený z důvodu jednoduchosti, univerzálnosti a snadné realizace elektronickými obvody. Lze je použít i pro regulaci složitých nelineárních systémů. Rovnice PID regulátoru je

$$u(t) = r_0 e(t) + r_1 \int_{\tau=0}^{t} e(\tau)\mathrm{d}\tau + r_2 e'(t), \tag{1.1}$$

kde u je akční zásah,

r_0 – zesilující složka (P),

r_1 – integrační složka (I),

r_2 – derivační složka (D),

e – regulační odchylka.

Přenos regulátoru je

$$R(s) = r_0 + r_1 \frac{1}{s} + r_2 s\,. \tag{1.2}$$

Hodnoty 3 složek určují varianty PID regulátorů.

P-regulátor (záporná zpětná vazba) … $r_1 = r_2 = 0$

PD-regulátor (derivační složka navíc) … $r_1 = 0$

PI-regulátor (integrační složka navíc) … $r_2 = 0$

I-regulátor (samotná integrační složka) … $r_0 = r_2 = 0$

D-regulátor se nevyužívá (nemá zpětnou vazbu) … $r_0 = r_1 = 0$

PID složky a jejich význam

P složka

V podstatě to je zesílená zpětná vazba. Rychlost regulačního děje je závislá na velikosti zesílení. Pro příliš vysoké hodnoty je kmitavý a může být nestabilní. Takže P složka zesiluje, čímž pomáhá se dostat na požadovanou hodnotu, ale způsobuje trvalou regulační odchylku na výstupu. Samostatná P složka nezaručí dosažení požadované hodnoty, proto pro dosažení nulové regulační odchylky u regulátorů kombinuje s dalšími složkami.

I složka

Integrační složka umožní dosáhnout nulové regulační odchylky i u statických soustav. Však zvyšuje řád soustavy a prodlužuje regulační děj, takže velikost I složky je volena podle rychlosti soustavy (u rychlé soustavy je možné volit velikou I složku). Je dosaženo lepší přesnost za cenu prodloužení doby regulace.

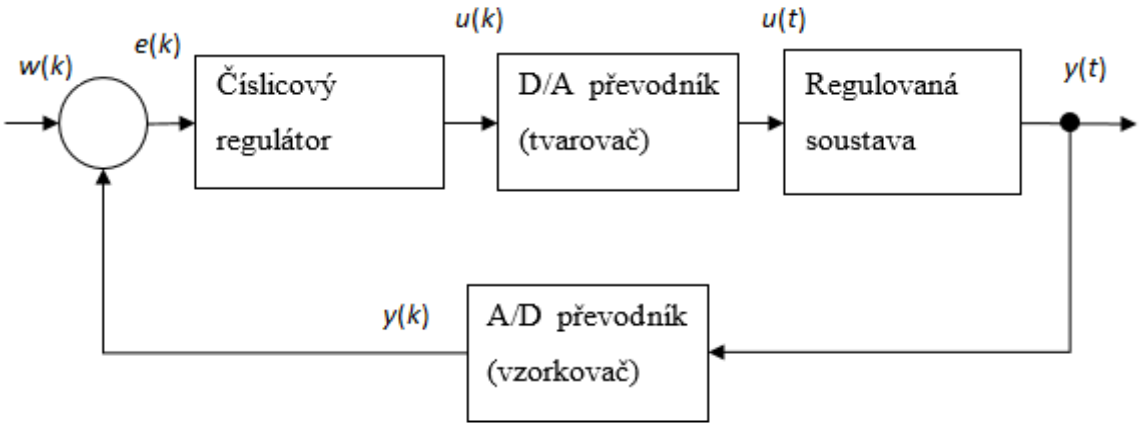
D složka

D složka zkracuje dobu regulace. Poskytuje možnost reagovat s určitým předstihem, ale zesiluje šum. Její vliv mizí v ustáleném stavu, protože reaguje pouze na změnu. Snižuje

relativní řád přenosu otevřené smyčky, tím zmenšuje prodlevu, než systémy vyšších řádů zareagují. Prakticky ho ale nelze zrealizovat, protože by jeho amplituda musela být rovna nekonečnu během nekonečně krátkého intervalu. Prakticky zrealizovaný má zkreslení a tím zesiluje šum.

1.2.3 Číslicový regulátor

Pokud je soustava řízena počítačem nebo mikropočítačem, tak se využívá regulátoru vyhodnocující informaci v diskrétních okamžicích $t = kT$ ($k = 0, 1, 2, \ldots$). Vzorkovací perioda by měla být dost krátká, aby se zajistila dostatečná přesnost. O vzorkování se stará A/D převodník, který následně informaci převede do číslicového signálu. Tento signál se zpracuje v příslušném počítači nebo mikropočítači a je vypočten akční zásah. Tento akční zásah převede D/A převodník spolu s tvarovačem na spojitý průběh. Schéma můžeme vidět na obrázku 1.12. (Balátě, 2004)



Obrázek 1.12 - Schéma číslicového regulátoru

Spojitý regulátor vypočítává hodnotu akční veličiny $u(t)$ v jakémkoliv okamžiku t na základě znalosti regulační odchylky $e(t) = w(t) - y(t)$. To lze popsat rovnicí (1.4) s odpovídajícím L – přenosem, který je v rovnici

$$G_R(s) = \frac{U(s)}{E(s)} = r_0\left(1 + \frac{1}{T_I s} + T_D s\right). \tag{1.3}$$

$$u(t) = r_0\left[e(t) + \frac{1}{T_I}\int_{\tau=0}^{t} e(\tau)d(\tau) + T_D\frac{de(t)}{dt}\right] \tag{1.4}$$

Pokud libovolný okamžik spojitého regulátoru splyne s k-tým vzorkovacím okamžikem je možné rovnici (1.3) přepsat na rovnici

$$u(kT) = r_0 e(kT) + \frac{r_0}{T_\mathrm{I}} I(kT) + r_0 T_\mathrm{D} D(kT). \tag{1.5}$$

Tuto rovnici lze využít pro vypočtení akčního zásahu v k-tém kroku, ale je nutné znát hodnoty integrálu $I(kT)$ a derivace $D(kT)$ v daném diskrétním časovém okamžiku $t = kT$. Hodnoty integrálu a derivace jsou numericky vypočteny z diskrétních hodnot regulační odchylky. Ze současné hodnoty a z minulých hodnot, které musí být uložené. Integrace se nahrazuje sumací. Užívají se tři druhy náhrad integrace:

- Zpětná obdélníková – ZOBD (stupňovitá náhrada zpět)
- Dopředná obdélníková – DOBD
- Lichoběžníková – LICHO (sečnová náhrada).

Nejčastěji se používá zpětná obdélníková, kterou lze popsat rovnicí (1.6), kde plocha pod původní křivkou e (t) je nahrazena součtem ploch obdélníků.

$$I(kT) = \int_0^t e(\tau)\mathrm{d}\tau \approx T\sum_{i=1}^{k} e(iT) \hat{=} \frac{Tz}{z-1} \tag{1.6}$$

Derivace se nahrazuje zpětnou diferencí popsanou rovnicí

$$D(kT) = \frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \frac{e(kT) - e[(k-1)T]}{T} = \frac{z-1}{Tz}. \tag{1.7}$$

Rovnice (1.8), která popisuje polohový regulátor, je získána nahrazením integrace zpětnou obdélníkovou metodou a derivace zpětnou diferencí v rovnici (1.5).

$$e(kT) = r_0\left\{e(kT) + \frac{T}{T_\mathrm{I}}\sum_{i=1}^{k} e(iT) + \frac{T_\mathrm{D}}{T}\{e(kT) - e[(k-1)T]\}\right\} + u(0) \tag{1.8}$$

Tento regulátor má tři složky:

- P – proporcionální
- S – sumační
- D – diferenční

Proto se tento regulátor nazývá PSD regulátor. Ale tento polohový regulátor je nerekurentní (musí se uchovat všechny hodnoty regulační odchylky), což je nevýhodné. Pracovat s rekurentním regulátorem je daleko vhodnější. Takový regulátor je přírůstkový, který neurčuje celou hodnotu akční veličiny pro daný okamžik, ale pouze její změnu (přírůstek oproti hodnotě v předchozím kroku). Tento tvar se dá zapsat rovnicí (1.9), kde jsou zavedeny parametry q_0, q_1 a q_2, které jsou vyjádřeny v rovnici (1.10). Tento přírůstkový PSD regulátor generuje akční zásahy z aktuální regulační odchylky, minulé regulační odchylky a předminulé regulační odchylky a tím odpadá nutnost uchovávat všechny hodnoty regulační odchylky.

$$\nabla u(kT) = q_0 e(kT) + q_1 e[(k-1)T] + q_2 e[(k-2)T] \tag{1.9}$$

$$q_0 = r_0\left(1 + \frac{T}{T_\mathrm{I}} + \frac{T_\mathrm{D}}{T}\right)$$

$$q_1 = -r_0\left(1 + 2\frac{T_\mathrm{D}}{T}\right) \tag{1.10}$$

$$q_2 = r_0 \frac{T_\mathrm{D}}{T}$$

Parametry k_R, T_I, T_D jsou stavitelné a je možné do nich přímo dosadit hodnoty získané jakoukoliv metodou pro spojitý PID regulátor.

1.3 Metody pro nastavení PID regulátoru

1.3.1 Metody Zielera a Nicholse

Existují dvě varianty Metoda ustálených kmitů a Metoda vyhodnocení přechodové charakteristiky. Obě metody byly původně pro regulaci chemických procesů a jejich výhodou je, že nevyžadují znalost přenosu. (Cvejn, 2015)

Metoda ustálených kmitů

Tato metoda využívá postupného zvětšování proporcionálního zesílení, při nulových složkách I a D, až do okamžiku, kdy se soustava dostane na tzv. mez stability. Na mezi stability se objeví netlumené kmity s konstantní amplitudou. Zesílení, na kterém se toto stane, se označuje jako kritické zesílení r_k a perioda ustálených kmitů onačíme T_k. T_k se dá vyjádřit rovnicí

$$T_\mathrm{k} = \frac{2\pi}{\omega_\mathrm{k}}. \tag{1.11}$$

Dosazením hodnot r_k a T_k do tabulky 1.1 určíme doporučené hodnoty parametrů PID regulátoru.

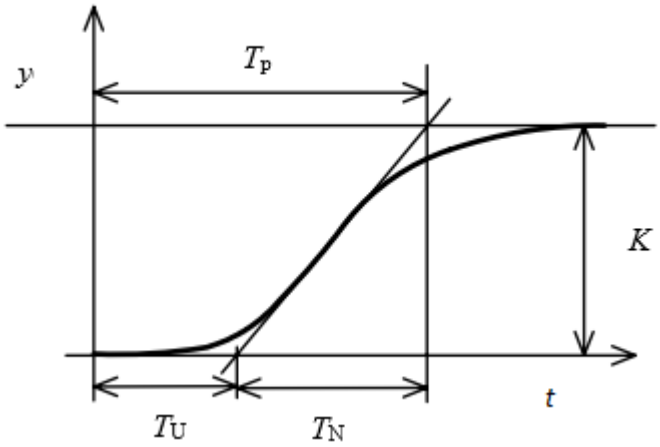
Tabulka 1.1 – Metoda kritického zesílení

Regulátor	r_0	T_I	T_D
P	$0{,}5r_\mathrm{k}$	-	-
PI	$0{,}45r_\mathrm{k}$	$0{,}85T_\mathrm{k}$	-
PD	$0{,}4r_\mathrm{k}$	-	$0{,}05T_\mathrm{k}$
PID	$0{,}6r_\mathrm{k}$	$0{,}5T_\mathrm{k}$	$0{,}125T_\mathrm{k}$

Tuto metodu je vhodné použít pouze u systému třetího a vyššího řádu. Kdybychom ji aplikovali na soustavu druhého řádu, tak jsme schopni určit pouze kritické zesílení.

Metoda na základě přechodové charakteristiky

Tato metoda je odvozená z metody kritických kmitů a umožnuje aplikaci na soustavách nižších řádů. Je doporučena pro systémy, které nekmitají a mají hodnotu normalizovaného zpoždění v rozsahu 0,1 - 0,5. Využívá, jak název napovídá, znalosti přechodové charakteristiky systému. Příklad přechodové charakteristiky je na obrázku 1.13.



Obrázek 1.13 - Přechodová charakteristika systému

Z této charakteristiky lze zjistit hodnoty doby náběhu T_N a doby průtahu T_U. Tyto dvě hodnoty definují tzv. normalizované dopravní zpoždění vyjádřené v rovnici

$$\Theta = \frac{T_U}{T_N}. \tag{1.12}$$

Důkaz, že tato metoda je odvozená z předchozí, je dokázáno rovnicí

$$r_k \approx \frac{2}{K\Theta}, T_k \approx 4T_U. \tag{1.13}$$

Je možnost hodnoty přepočítat a dosadit do tabulky 1.1, nebo do upravené tabulky 1.2.

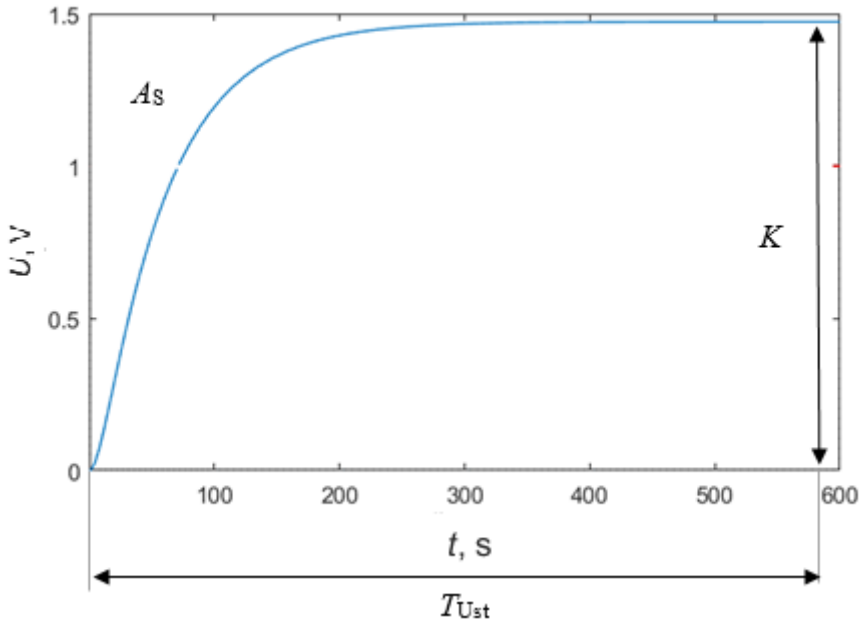
Tabulka 1.2 – Nastavení PID regulátoru na základě přechodové charakteristiky

Regulátor	r_0	T_I	T_D
P	$1/(K\Theta)$	-	-
PI	$0{,}9/(K\Theta)$	$3T_U$	-
PID	$1{,}2/(K\Theta)$	$2T_U$	$0{,}5T_U$

1.3.2 Kuhnova metoda

Kuhnova metoda, nebo také pravidlo souhrnné časové konstanty, byla odvozena v roce 1995 a využívá přechodové charakteristiky soustavy. Kuhnova pravidla jsou poměrně konzervativní. Regulační pochod má přibližně stejnou odezvu na změnu žádané hodnoty i na vstupní poruchu a obvykle málo kmitá. Pokud Kuhnova metoda porovnána s jinými, tak je zjištěno, že podává dobré výsledky hlavně u PI regulátorů, ale méně u PID regulátorů. (Honc, 2015)

Z přechodové charakteristiky na obrázku 1.14 lze zjistit parametry zesílení K a plochu nad křivkou A_S. Plocha A_S je definována pomocí rovnice (1.14). Z těchto dvou parametrů se počítá souhrnná časová konstanta T_Σ podle rovnice (1.15).



Obrázek 1.14 - Přechodová charakteristika systému

$$A_S = KT_{Ust} - \int_0^{T_{Ust}} y(t)\mathrm{d}t \tag{1.14}$$

$$T_\Sigma = \frac{A_S}{K} \tag{1.15}$$

Když je známo zesílení K a souhrnná časová konstanta T_Σ, tak lze je dosadit do tabulky 1.3 nebo tabulky 1.4, čímž budou zjištěny parametry PID regulátoru.

Tabulka 1.3 – Nastavení PID regulátoru Kuhnovou metodou – rychlé nastavení

Regulátor	r_0	T_I	T_D
P	$1/K$	-	-
PI	$2/K$	$0{,}7T_\sum$	-
PID	2/K	$0{,}8T_\sum$	$0{,}194T_\sum$

Tabulka 1.4 – Nastavení PID regulátoru Kuhnovou metodou – normální nastavení

Regulátor	r0	T_I	T_D
P	$1/K$	-	-
PI	$0{,}5/K$	$0{,}5\ T_\sum$	-
PID	$1/K$	$0{,}66\ T_\sum$	$0{,}167T_\sum$

Použití rychlé metody je vhodné, pokud je dominantní dynamika regulované soustavy popsána systémem prvního nebo druhého řádu.

2 PRAKTICKÁ ČÁST

2.1 Arduino uno

Arduino uno je vývojová deska, která je založena na jednočipu ATmega328P. Obsahuje 14 digitálních vstupně výstupních pinů, z toho 6 lze použít jako výstupy PWM, 6 analogových vstupů, 16MHz krystal, USB konektor, napájecí konektor typu jack, ICSP rozhraní a resetovací tlačítko. Obsahuje tedy vše potřebné k použití mikrokontoléru. Uno je první z výrobků Arduino, které má možnost propojení přes USB port, což zajišťuje snadné připojení k počítači. Desku lze napájet pomocí USB, adaptéru nebo externí baterie. (Voda, 2015)

Arduino je možné programovat v jazyce C, C++ a pomocí knihovny Wiring. Knihovna Wiring je velice rozšířená a komplexní, a proto se mluví jako o samostatném programovacím jazyku, který je pro programování Arduina nejjednodušší. Toto nahrávání kódu se provádí pomocí bootloaderu bez použití externího hardwarového programátoru. Bootloader se dá obejít a mikrokontrolér se dá programovat přes ICSP (In-Circuit Seriál Programming) rozhraní.

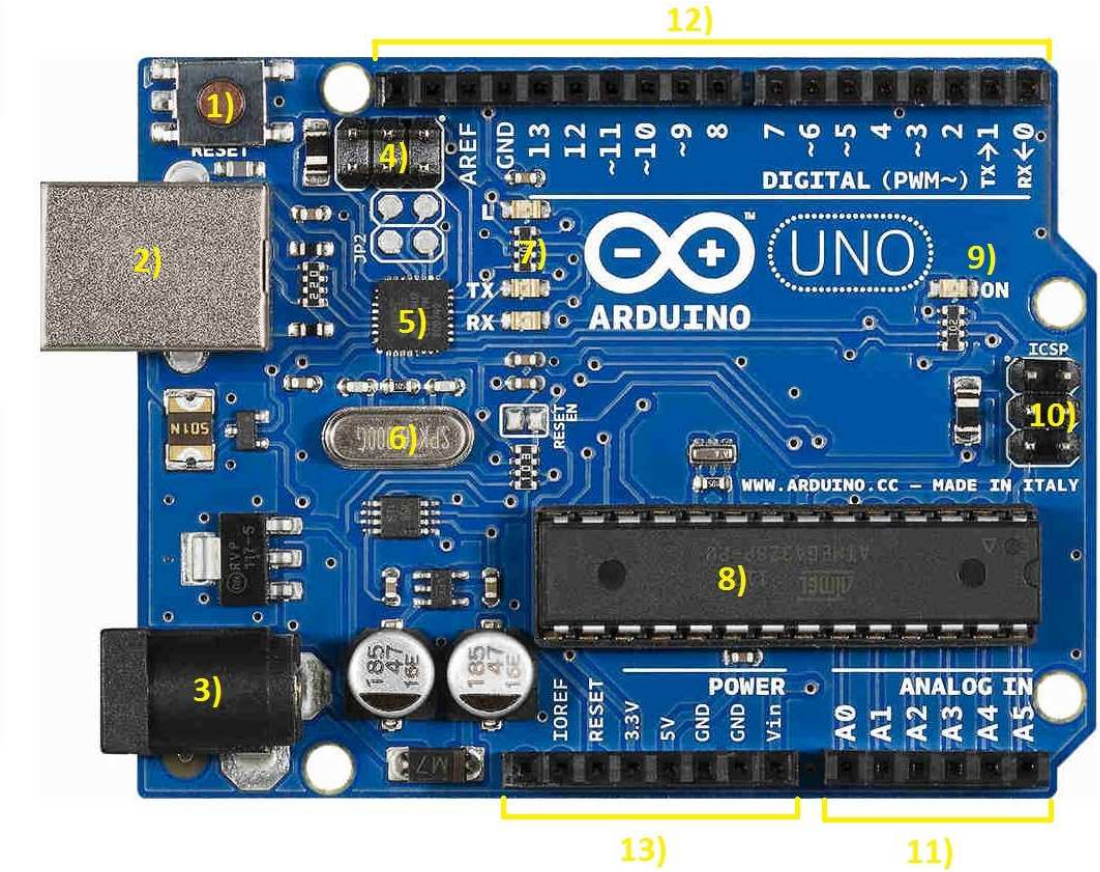
Arduino lze napájet přes USB nebo externím napájením. Napájecí zdroj je vybírán automaticky. Externí zdroj může být buď adaptér, nebo baterie. Externí napájecí napětí může být 6 - 20V. Při napájení menším než 7V je možnost, že piny nebudou schopny dodávat plných 5V, které slibují, a deska může být nestabilní. Pokud je napájení větší jak 12V, tak regulátor napětí může zapříčinit přehřátí a zničení desky. Doporučené napětí je 7 - 12V.

ATmega328P obsahuje paměti typu Flash, SRAM a EEPROM. Flash pamět o velikosti 32KB poskytuje 0.5KB bootloaderu pro nahrávání programu a zbytek je pro samotný program. Paměť SRAM, která je veliká 2KB, je pro data. EEPROM je datová paměť sloužící pro dlouhodobé uložení dat (např. tabulky hodnot) do velikosti 1KB.

Digitální vstupně/výstupní piny, kterých má Arduino uno čtrnáct, fungují na úrovni 5V. U každého z pinů je doporučeno manipulovat s proudem 20 mA při použití v obou režimech. Každý z pinů má vnitřní pull-up rezistor (odpojený ve výchozím stavu) 20-50k ohmů. Na žádném z pinů se nesmí překročit proud 40mA, aby se zabránilo trvalému poškození mikrokontroléru. Některé piny mají specializované funkce jako RX, TX, externí přerušení, PWM, SPI, TWI, SDA a SCL. Arduino uno obsahuje 6 analogových vstupů, které mají 10 bitů rozlišení (1024 hodnot). Ve výchozím nastavení měří oproti zemi až 5V. Lze také měřit oproti referenčnímu zdroji napětí.

Arduino má spoustu možností pro komunikaci s počítačem. ATmega328 poskytuje UART TTL (5V) sériovou komunikaci, která je k dispozici na digitálních pinech RX a TX. Dále také podporuje I2C (TWI) a SPI komunikaci. Sériová komunikace přes USB je zajištěna

pomocí ATmega16U2, která je na desce, a jeví Arduino v počítači jako virtuální port COM. Pokud se data přesouvají pomocí USB, tak indikační led RX a TX svítí.



Obrázek 2.1 - Arduino Uno

Tabulka 2.1 – Legenda pro Arduino uno

Číslo	Popis
1	Tlačítko reset
2	USB konektor typu B
3	Napájecí konektor
4	ICSP rozhraní pro ATmega16U2
5	ATmega16U2
6	16MHz krystal
7	Indikační led RX, TX a L
8	ATmega328P
9	Indikační led On
10	ICSP rozhraní pro ATmega328P
11	Analogové vstupy
12	Digitální piny
13	Napájecí výstupy a reset

2.2 Matlab

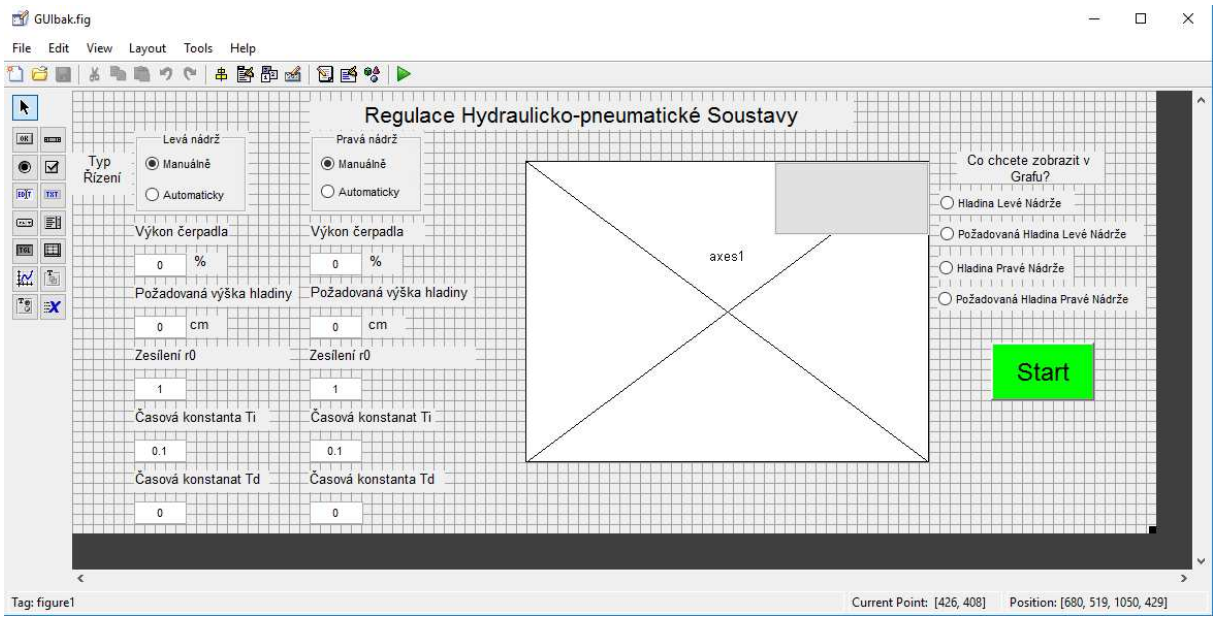
2.2.1 Úvodní seznámení

MATLAB (MATrix LABoratory) vyvíjí firma MathWorks a spojuje technické výpočty, vizualizaci dat a programovací jazyk do jednoho výkonného interaktivního prostředí. Toto prostředí je možné rozšířit o různé moduly, knihovny a toolboxy. Mezi jeho klíčové vlastnosti patří vysokoúrovňový jazyk pro technické výpočty, otevřený a rozšířitelný systém, velké množství aplikačních knihoven, podpora vícerozměrných polí a datových struktur, alternativní nástroje pro tvorbu grafického uživatelského rozhraní, import a export dat mnoha formátů, komunikace s externími měřícími a monitorovacími přístroji v reálném čase a rozšiřitelnost modulů jazyky C, C++, Fortran a Java. S těmito předpoklady se stává vhodným prostředkem pro inženýrské výpočty, vývoj algoritmů, modelování, simulaci a vývoj prototypů, analýzu dat a jejich vizualizaci, inženýrskou grafiku nebo vývoj aplikací včetně tvorby grafického uživatelského rozhraní. (Dušek, 2002) (Zaplatílek, 2005) (Horáček, 2014)

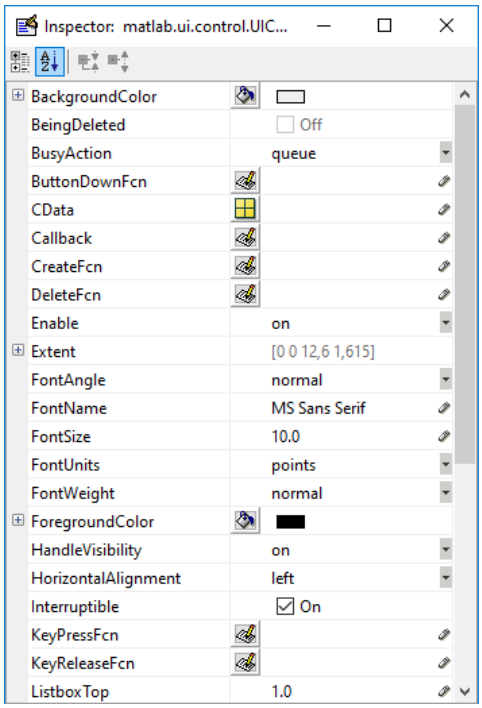
2.2.2 Tvorba GUI

Vytvořit GUI lze za pomocí nástroje GUIDE, který je implementován v MATLABu, nebo je možné potřebný kód napsat ručně za předpokladu, že je dostatečná znalost objektů a funkcí v MATLABu. GUIDE poskytuje soustavu nástrojů, které proces návrhu GUI velmi zjednoduší. Poskytuje nejen sadu návrhových nástrojů, ale také generuje výsledný M-file, ve kterém je obsažen kód pro ovládání, inicializaci a spouštění GUI a také obsahuje *callback funkce*, které se vykonávají, jakmile uživatelé aktivují objekty v GUI.

V nástroji GUIDE byl vytvořen vzhled aplikace, který je zobrazen na obrázku 2.2, kde bylo využito různých prvků jako např. *togglebutton*, *Radio Button*, *Button Group*, *Axes*, *Edit Text* nebo *Static Text*. Použitým prvkům byly nastavil jednotlivé požadované vlastnosti přes Property Inspektor, který je vidět na obrázku 2.3.



Obrázek 2.2 - Tvorba aplikace v GUIDE



Obrázek 2.3 - Property Inspector

2.2.3 Popis Programu

Program je rozepsaný v M-file souboru, který automaticky po uložení navrhnutého prostředí vygenerovalo GUIDE.

Definování Arduina

```
global a;
a = arduino('COM4','uno');
```

Jelikož v tomto případě je propojován MATLAB a Arduino za pomocí podpůrné knihovny, tak musela být nejprve v *OutputFcn* nadefinovat globální proměnná, která představuje samotné Arduino. Tato proměnná obsahuje informaci o portu, na kterém se Arduino nachází a typ Arduina(udává MATLABu informaci o jednotlivých pinech Arduina).

Zadávání hodnot

```
P = str2num(get(hObject,'String'));
if isempty(P)
   msgbox('Zadej pouze číslo!','Pozor','Warn')
   P = 0;
   set(handles.edit1,'String',num2str(P));

else
   if(P > 500)
        P = 500;
        set(handles.edit1,'String',num2str(P));
        msgbox('Maximální hodnota P je 500!','Pozor','Warn')
   elseif(P < 0)
        P = 0;
        set(handles.edit1,'String',num2str(P));
        msgbox('Minimální hodnota P je 0!','Pozor','Warn')
   end
end
```

Tímto způsobem bylo ošetřeno zadávání nesmyslných hodnot do všech polí. Je zde nejprve převod na číslo, protože *Edit Text* pracuje s formátem string. Poté se provede kontrola, jestli nebylo vepsáno písmeno, nebo nesmyslně velká hodnota. Pokud ano, tak vyskočí *MessageBox* s patřičným upozorněním a hodnota v daném poli se změní na odpovídající hodnotu.

Výběr režimu (zkráceně)

```
typ = get(get(handles.typrizeni1,'SelectedObject'),'String');
switch typ
   case 'Manuálně'
      set(handles.edit1,'Visible','Off','Enable','Off');
      set(handles.edit4,'Visible','On','Enable','On');
      set(handles.text5,'Visible','Off');
      set(handles.text6,'Visible','On');
      set(handles.radiobutton6,'Value',get(handles.radiobutton6,'Min'));
   case 'Automaticky'
      set(handles.edit3,'Enable','On','Visible','On');
      set(handles.edit4,'Enable','Off');
      set(handles.edit9,'Visible','On','Enable','On');
      set(handles.text11,'Visible','On');
      set(handles.radiobutton6,'Enable','On');
end
```

V *callbacku* objektu *ButtonGroup* byl pomocí příkazu *switch* zrealizován výběr režimu, který kontroluje, jaký režim je v danou chvíli vybraný. Podle zvoleného režimu zobrazí text a umožní zapisování do polí určených pro daným režim.

Ovládání soustavy

Program pro ovládání je velmi dlouhý, a proto je zde shrnuto pouze to nejpodstatnější.

Jsou zde nadefinované proměnné, se kterými se poté operuje v různých částech programu.

```
typrizeni = 0;
vyskyhladin = 0;
pozadovanevyskyhladin = 0;
vykonycerpadla = 0;
error = 0;
lastTime = now;
atd.
```

Během ovládání běhá program ve smyčce vytvořenou příkazem *while*, kde se neustále vypočítávají potřebné hodnoty pro regulaci a řízení. Podmínkou je realizováno přepínání mezi

jednotlivými typy řízení. V obou případech se načtou do proměnných hodnoty zadané v uživatelském rozhraní a hodnoty ze senzorů.

```
pozadovanavyskahladiny = str2num(get(handles.edit9,'String'));
P = str2num(get(handles.edit1,'String'));
I = P / str2num(get(handles.edit2,'String'));
D = P * str2num(get(handles.edit3,'String'));
vykoncerpadla = str2num(get(handles.edit4,'String'));
```

Načítání hodnot ze senzorů je speciálním příkazem.

```
senzorp = readVoltage(a,'A1');
```

Tento příkaz je dostupný pouze po nainstalování balíčku a vypisuje hodnotu napětí, které je na analogovém vstupu A1 přítomném na platformě Arduino.

U automatického režimu se vypočtou jednotlivé složky potřebné pro zjištění regulačního zásahu a následně se vypočte samotný akční zásah.

```
timeChange = (now - lastTime)*3600*24; %Změna času v sec
    lastTime = now;%Uložení předchozího času
    error = (pozadovanavyskahladinyveV - vyskahladinyveV);
    integrace = integrace + (error * timeChange);
    derivace = (error - lastErr) / timeChange;
    out = P * error + I * integrace + D * derivace;
```

V obou případech je následně opět speciálním příkazem zadán akční zásah.

```
writePWMVoltage(a,'D9',out);
```

Tento příkaz, který je také dostupní po instalaci balíčku, zajistí, aby na patřičném pinu Arduina bylo odpovídající napětí. Toto napětí není spojité, ale je realizováno pomocí PWM signálu.

S každým opakováním je také do grafu vykreslovány hodnoty, které chceme zobrazit (výška hladiny a požadovaná výška hladiny).

```
if get(handles.radiobutton5,'Value') == get(handles.radiobutton5,'Max')
    plotHandle =
plot(handles.axes1,time,vyskyhladin,'Marker','.','LineWidth',.25,'Color','blue');
    set(plotHandle,'YData',vyskyhladin,'XData',time);
end
```

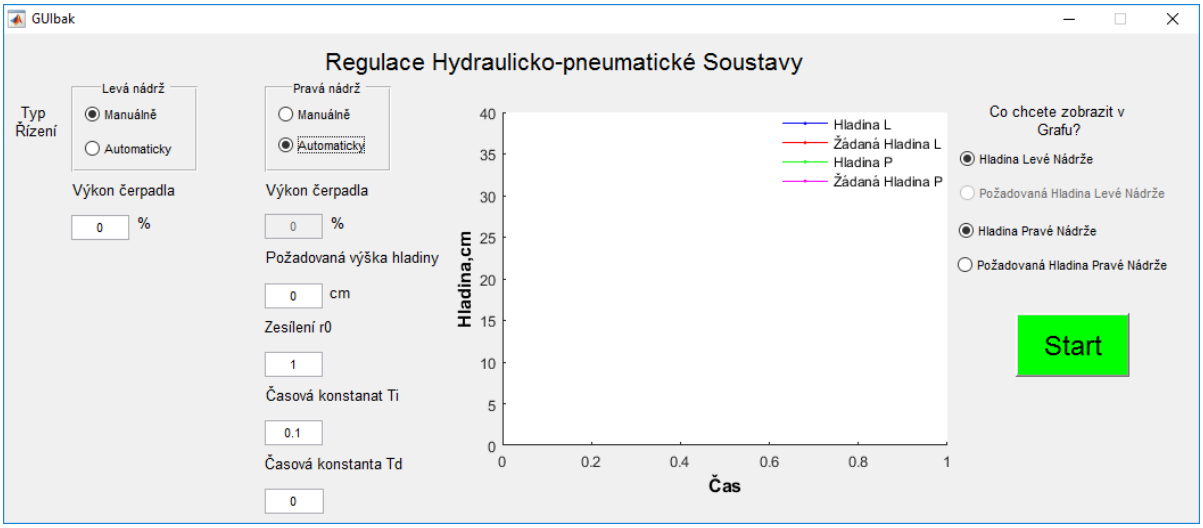
Po ukončení ovládání jsou veškerá data uložena do souboru a na výstupní piny se nastaví nula.

```
%Uložení dat z pravé nadrže
    data_out(:,1) = time_out;
    data_out(:,2) = pozadovanevyskyhladin;
    data_out(:,3) = vyskyhladin;
    data_out(:,4) = vykonycerpadla;
    data_out(:,5) = typriz;
    currentTime = TimeStamp();
    xlswrite(['Data\Data_leva_nadrz_',currentTime,'.csv'],data_out)

    writePWMVoltage(a,'D9',0);
    writePWMVoltage(a,'D10',0);
```

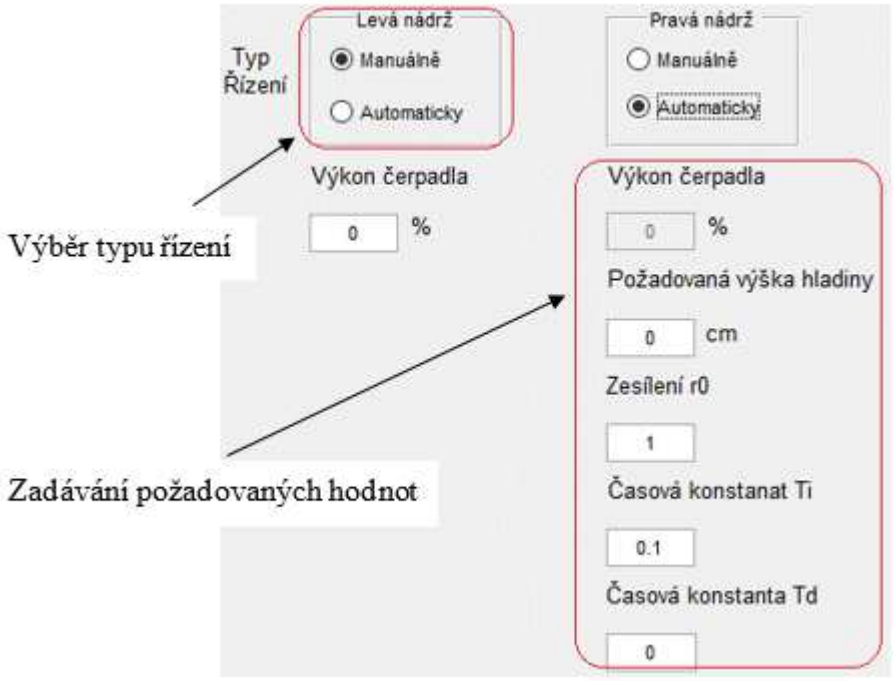
2.2.4 Práce s programem

Po spuštění se uživateli zobrazí základní okno, které je na obrázku 2.4.



Obrázek 2.4 - Základní okno aplikace

V levé horní části jsou přepínače pro výběr typu regulace. Podle vybraného typu řízení jsou pod těmito přepínači zobrazeny i textová pole pro zadávání požadovaných hodnot.

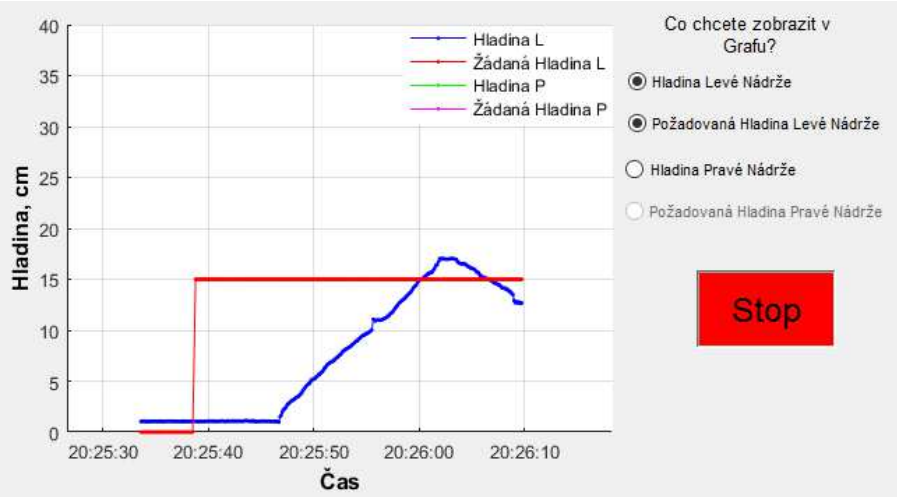


Obrázek 2.5 - Výběr typu řízení a zadávání hodnot

Soustavu se začne ovládat po zmáčknutí tlačítka START. V tuto chvíli se tlačítko změní na tlačítko STOP. Po dalším stisku se zastaví ovládání a uloží se veškerá data. Během ukládání dat se tlačítko změní na Ukládání dat!. Po uložení dat se tlačítko ocitne v prvotním stavu START. Všechny tři podoby tlačítka jsou na obrázku 2.6.

Obrázek 2.6 - Podoby tlačítka

Během ovládání jsou do grafu vykreslovány průběhy, které se volí pomocí přepínačů na pravé straně.

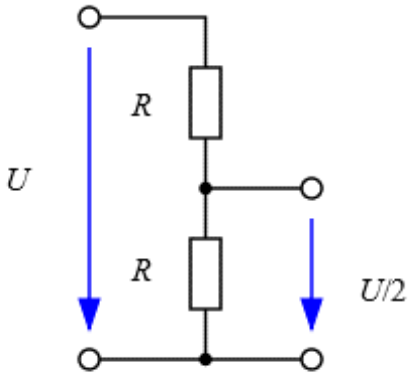


Obrázek 2.7 - Vykreslování průběhů

Během ovládání lze měnit typ řízení. Pokud se provede změna z manuálního na automatický režim, tak do požadované výšky hladiny se zadá aktuální hladina. V opačném případě se nastaví výkon čerpadla.

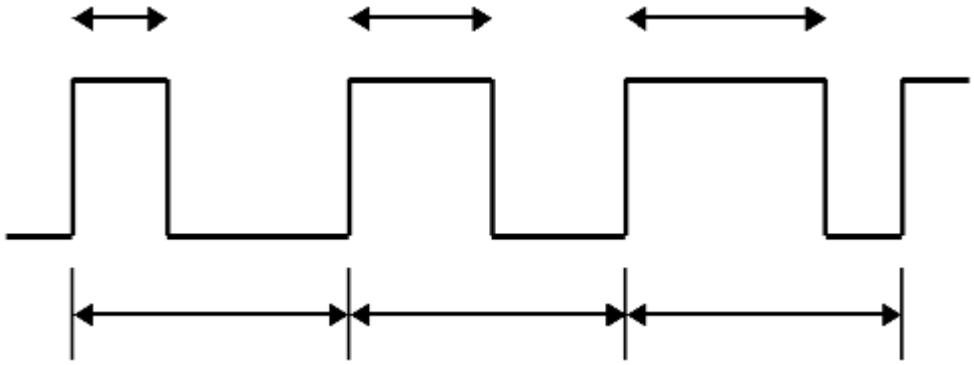
2.3 Arduino a přizpůsobovací obvody

Počítač je připojen k Arduinu USB kabelem, přes který je Arduino napájeno a aplikace komunikuje s Arduinem. Arduino je připojené k soustavě za pomocí podpůrných obvodům, které upravují signály tak, aby bylo možné tuto soustavu ovládat. Analogové vstupy Arduina dokážou zpracovávat napětí pouze do 5V, ale tlakové senzory informují o výšce hladiny signály od 0 do 10V. Z tohoto důvodu do Arduina musí signál přes dělič napětí vytvořený ze dvou stejných odporů, který je na obrázku 2.8. (Havlíček, 2016)



Obrázek 2.8 - Napěťoví dělič

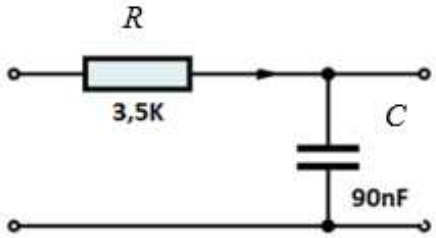
Arduino nemá přímo analogoví výstup, ale je nahrazen PWM signálem. Příklad PWM signálu je na obrázku 2.9.



Obrázek 2.9 - Příklad PWM signálu

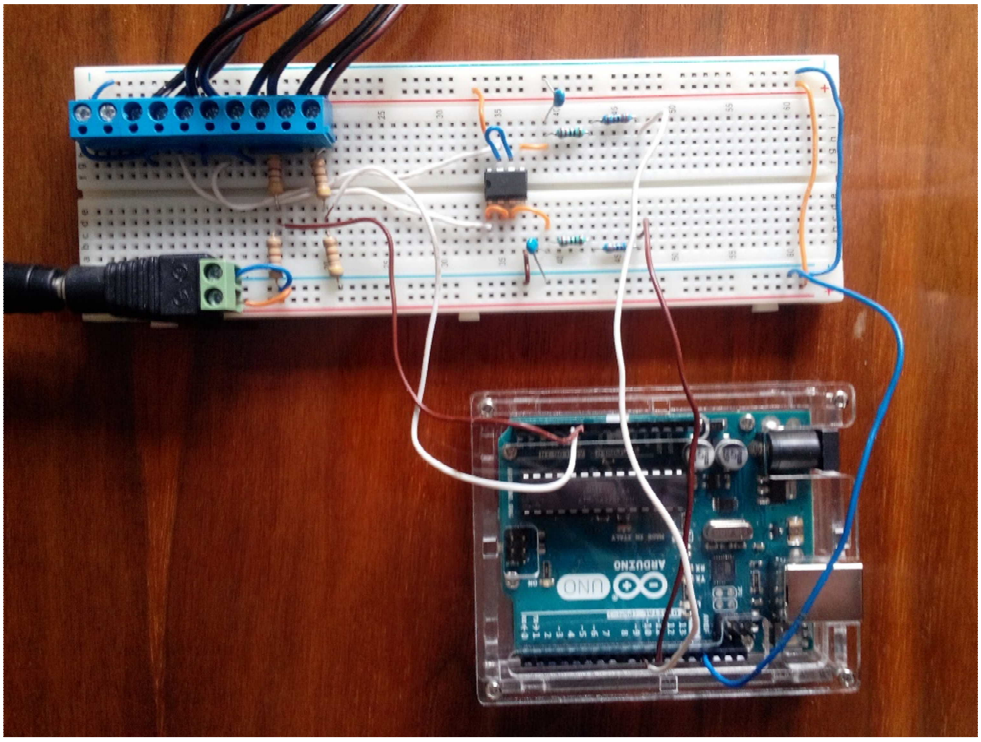
Tento signál je upraven pomocí RC filtru. Frekvence PWM signálu je 490 Hz a velikost kondenzátoru byla zvolena 90nF. Hodnota odporu byla vypočtena pomocí rovnice (2.1). Samotný RC filtr je vidět na obrázku 2.10. (Keim, 2014)

$$f_c = \frac{1}{2\pi RC} \tag{2.1}$$



Obrázek 2.10 - RC filtr

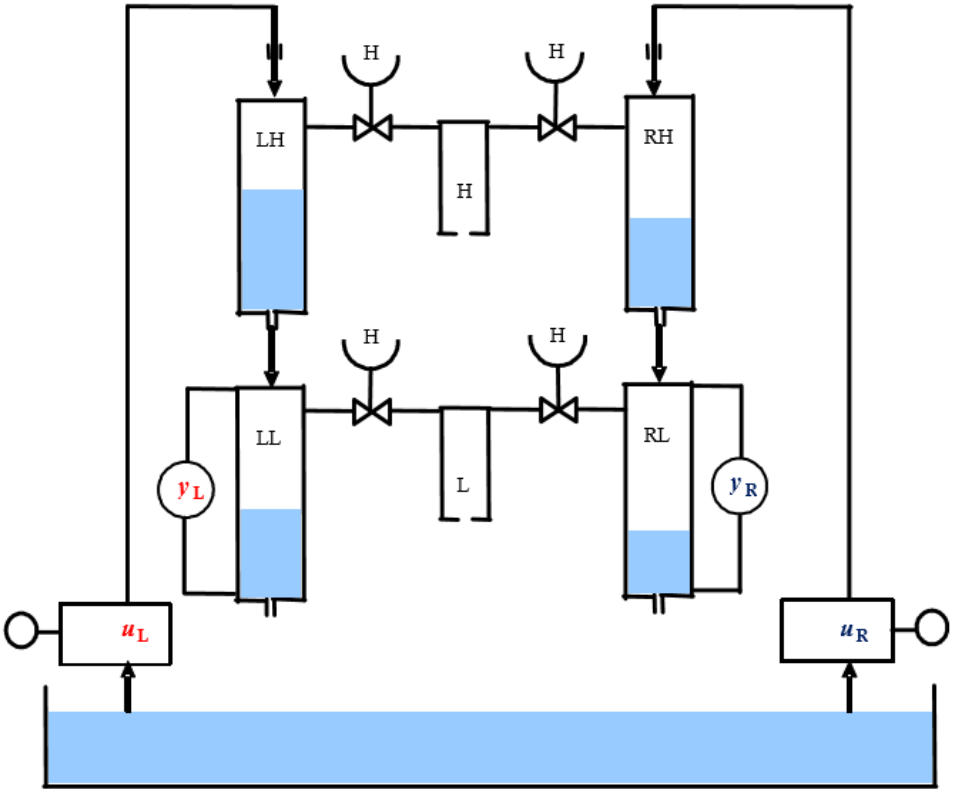
Po připojení k soustavě dojde k poklesu napětí, a proto signál upravený RC filtrem musí být zesílen operačním zesilovačem. Aby operační zesilovač dokázal dodat na výstup 5V, musí mít napájení minimálně 7V. Signál vystupující ze zesilovače už lze poslat do soustavy k ovládání čerpadel. Pomocné obvody jsou implementovány na nepájivém poli. Propojení pomocných obvodů a Arduina je na obrázku 2.11.



Obrázek 2.11 - Arduino s pomocnými obvody

2.4 Hydraulicko-pneumatická soustava

Tato hydraulicko-pneumatická soustava byla navrhnuta a zkonstruována na katedře Řízení procesů Univerzity Pardubice. Je sestavena kombinací hydraulických a pneumatických prvků, které představují čtyři vodní nádrže a dva vzdušníky. Dvě čerpadla umístěna v zásobníku s vodou pumpují vodu do horních nádrží. Z těchto nádrží voda protéká otvorem v dolní části do dolních nádrží. Z dolních nádrží voda vytéká zpět do zásobníku. Celá soustava je znázorněna na obrázku 2.12. (Honc, 2012)



Obrázek 2.12 - Schéma hydraulicko-pneumatické soustavy

Čerpadla, která obstarávají průtok vody, jsou řízeny vstupními signály v rozsahu 0 až 5V. Výšku hladiny měří diferenční tlakové senzory, které podávají informaci výstupním signálem v rozsahu 0 až 10V. Tyto senzory měří rozdíl tlaků nad hladinou a pod hladinou. Pneumatické obvody s manuálně nastavitelnými ventily propojují vzduchové prostory nad hladinou. Otvory ve vzduchových komorách vyrovnávají tlak mezi nádržemi a atmosférou. Manuálními ventily lze nastavit vzájemné ovlivnění mezi nádržemi.

2.5 Identifikační experimenty

V první řadě jsem u regulované soustavy změřil odezvu systému při skokové změny napětí na čerpadlech z 1 na 3V. Tyto průběhy jsem poupravil a normoval tak, abych mohl za pomocí funkcí v MATLABu zjistit model soustavy ve formě obrazového přenosu. Použitý kód vypadá takto:

```
X=fminsearch('hledej3',[1 1 1],[],x,u,y);
Z=X(1);
T1=X(2);
T2=X(3);
S=tf(Z,[T1*T2 T1+T2 1]);
yp=lsim(S,u,x,'zoh');
plot(x,y,'+',x,yp);
```

Funkce hledej3 je definována :

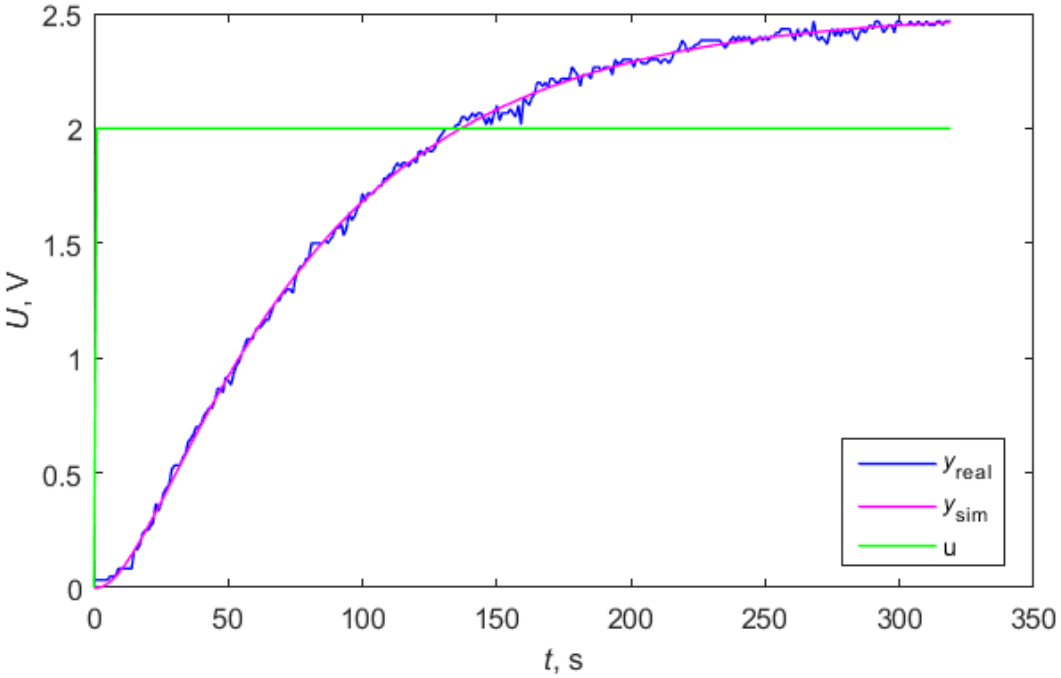
```
function K = hledej3(X,t,u,y)
Z=X(1);
T1=X(2);
T2=X(3);
S=tf(Z,[T1*T2 T1+T2 1]);
yp=lsim(S,u,t,'zoh');
K=(yp-y)'*(yp-y);.
```

Výsledkem jsou dva modely definované rovnicemi (2.2) a (2.3) pro pravou a levou soustavu.

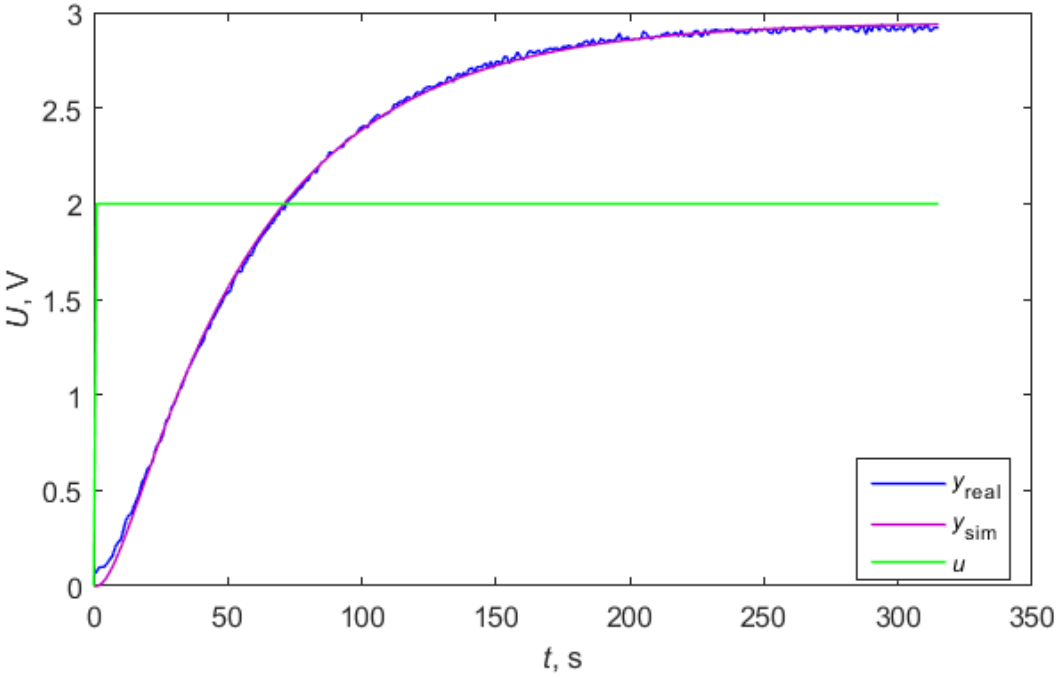
$$S_{\mathrm{L}} = \frac{1{,}254}{1038 \cdot s^2 + 89{,}2 \cdot s + 1} \tag{2.2}$$

$$S_{\mathrm{P}} = \frac{1{,}474}{452{,}7 \cdot s^2 + 62{,}94 \cdot s + 1} \tag{2.3}$$

Na obrázku 2.13 a 2.14 jsou naměřená data proložena odezvou počítanou z obou modelů.

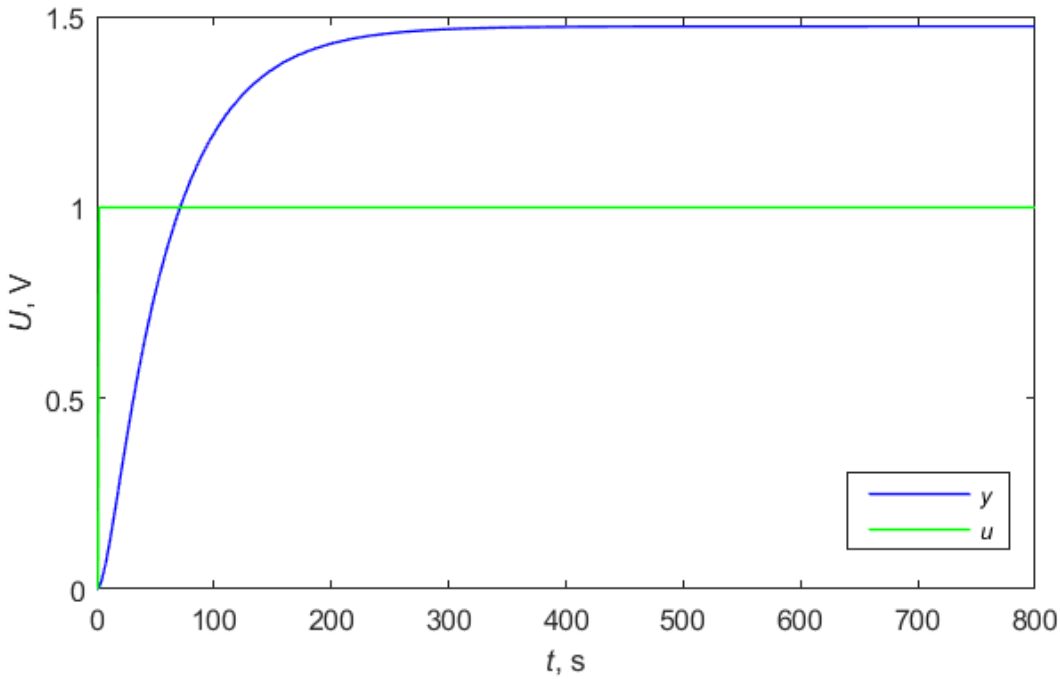


Obrázek 2.13 – Odezva levé nádrže spolu s odezvou modelu

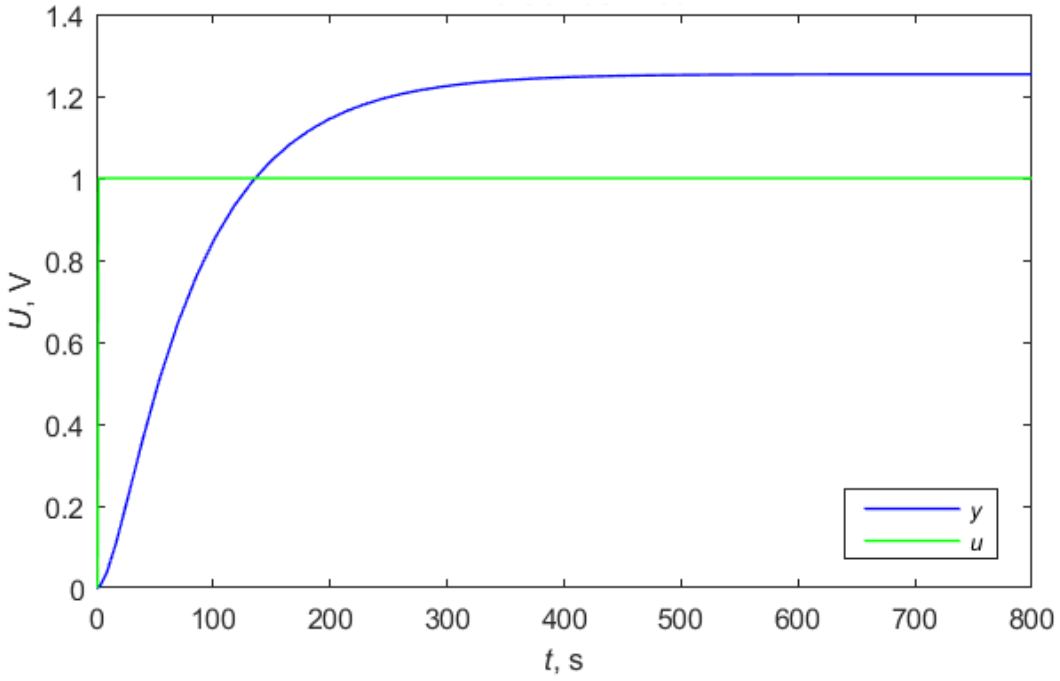


Obrázek 2.14 – Odezva pravé nádrže spolu s odezvou modelu

Pomocí těchto modelů soustav se mohou udělat přechodové charakteristiky soustav. To jsem udělal za pomocí MATLABu příkazem *step*. Výsledné přechodové charakteristiky jsou na obrázku 2.15 a 2.16.



Obrázek 2.15 – Přechodová charakteristika levé nádrže



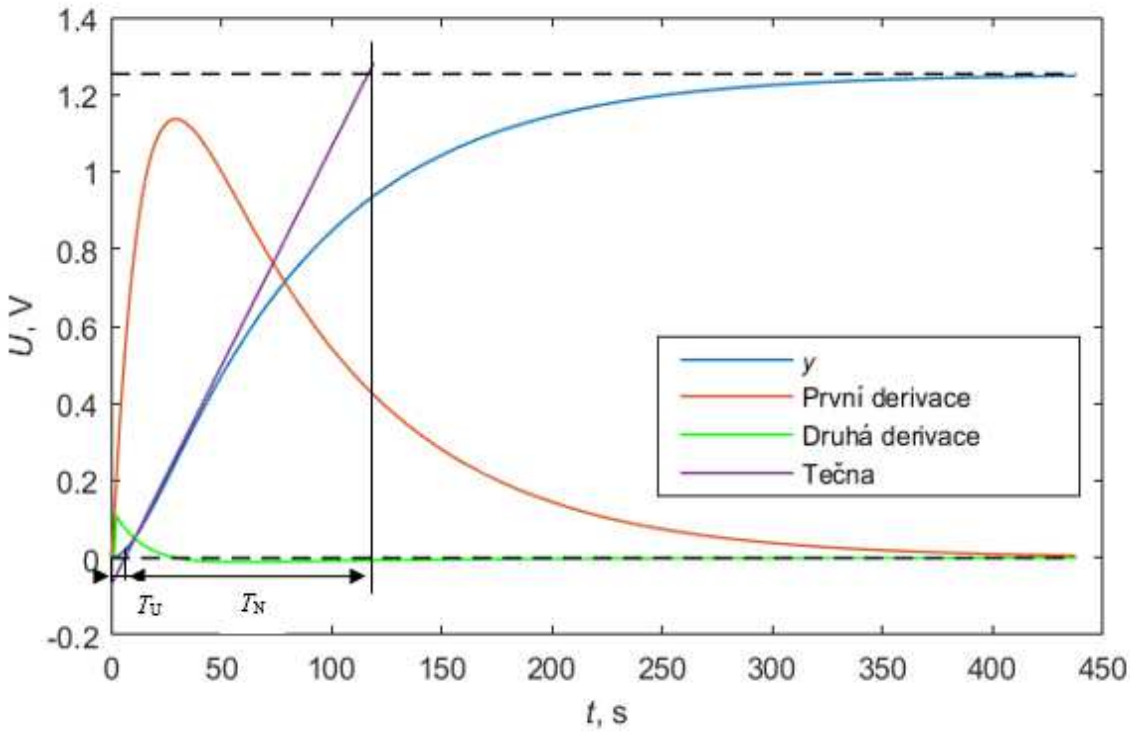
Obrázek 2.16 – Přechodová charakteristika pravé nádrže

S těmito přechodovými charakteristikami se již může pracovat a zjistit jednotlivými metodami složky PID regulátoru.

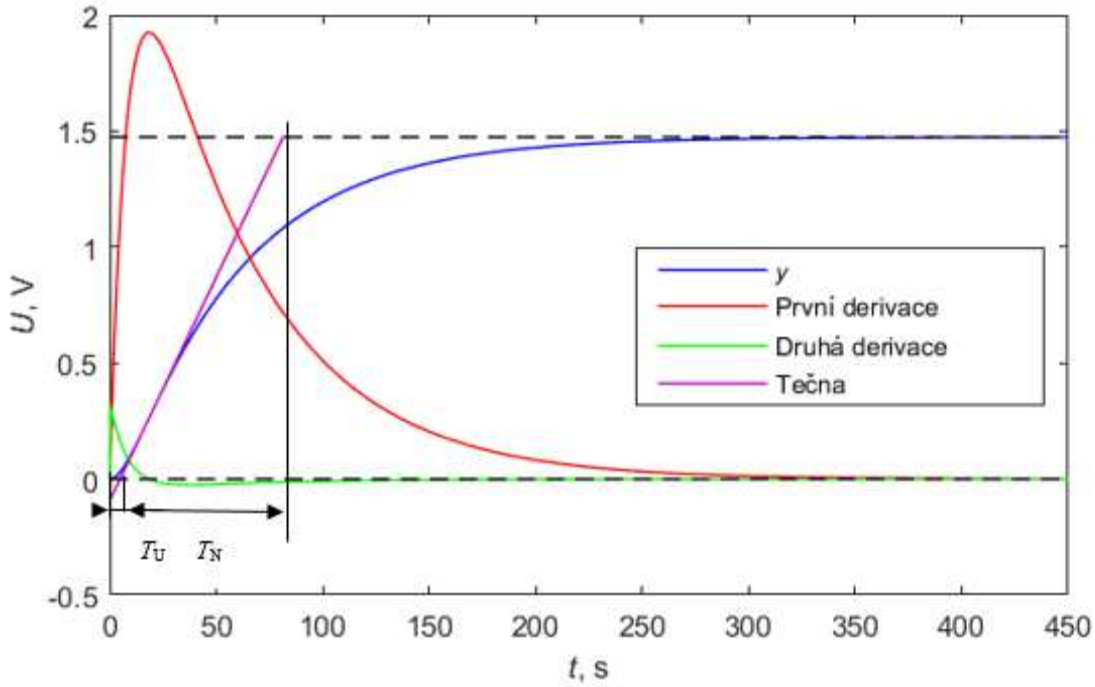
2.6 Návrh parametrů regulátoru

2.6.1 Nastavení parametrů regulátoru metodou Ziegler-Nichols

U obou soustav jsem vypočítal druhou derivaci přechodové charakteristiky a v bodě, kde protíná nulu, udělal tečnu, která mi určila dobu náběhu a dobu průtahu soustavy. Všechny průběhy jsou znázorněny na obrázku 2.17 a 2.18.



Obrázek 2.17 - Zjišťování PID pro levou nádrž metodou Ziegler-Nichols



Obrázek 2.18 - Zjišťování PID pro pravou nádrž metodou Ziegler-Nichols

Doby náběhu a průtahu jsem dosadil do rovnice (1.12) a dostal jsem normalizované dopravní zpoždění. Všechny data jsou v tabulce 2.2.

Tabulka 2.2 – Změřené hodnoty metodou Ziegler-Nichols

Levá				**Pravá**			
T_U	T_N	T_U/T_N	K	T_U	T_N	T_U/T_N	K
6,33	109,66	0,05781	1,254	4,8	76,6	0,062663	1,474

Tyto všechny hodnoty jsem použil do tabulky 1.2 a dostal jsem první parametry PID regulátoru určené metodou Ziegler-Nichols. Vypočtené parametry PID regulátoru metodou Ziegler-Nichols jsou v tabulce 2.3.

Tabulka 2.3 – Hodnoty PID regulátoru metodou Ziegler-Nichols

Ziegler-Nichols							
Levá				Pravá			
	r_0	T_I	T_D		r_0	T_I	T_D
P	13,79434			P	10,82655		
PI	12,41491	19,01835		PI	9,743894	14,4	
PID	16,55321	12,6789	3,169726	PID	12,99186	9,6	2,4

2.6.2 Nastavení parametrů regulátoru Kuhnovou metodou

U Kuhnovy metody jsem nejprve musel zjistit plochu nad křivkou. Tuto plochu jsem vypočítal podle vzorce, který je popsán v rovnici (1.14). Za pomocí rovnice (1.15) jsem vypočítal souhrnnou časovou konstantu pro obě dvě nádrže. Výsledky jsou v tabulce 2.4.

Tabulka 2.4 – Vypočtené hodnoty pro jednotlivé nádrže

Levá			**Pravá**		
K	A_S	T_{Ust}	K	A_S	T_{Ust}
1,254	59,72802	47,63	1,474	92,775034	62,941

Tyto hodnoty jsem dosadil do tabulky 1.3 a 1.4 a získal jsem hodnoty PID regulátorů Kuhnovou metodou pro rychlou i normální regulaci. Parametry PID regulátoru jsou vypsány v tabulce 2.5 a 2.6.

Tabulka 2.5 - Hodnoty PID regulátoru Kuhnovou metodou pro levou nádrž

Typ	Regulátor	r_0	T_I	T_D
Rychle	P	0,797448		
	PI	1,594896	33,341	
	PID	1,594896	38,104	9,24022
Normálně	P	0,797448		
	PI	0,627	23,815	
	PID	1,594896	31,4358	7,95421

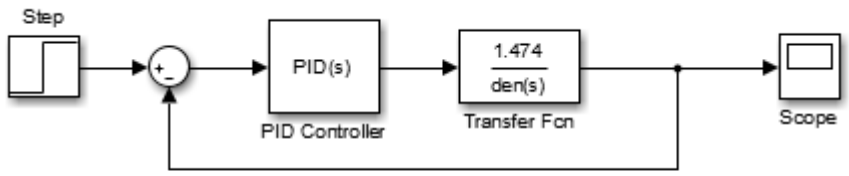
Tabulka 2.6 - Hodnoty PID regulátoru Kuhnovou metodou pro pravou nádrž

Typ	**Regulátor**	r_0	T_I	T_D
Rychle	P	0,67842		
	PI	1,356852	44,05899	
	PID	1,356852	50,35288	12,21057
Normálně	P	0,678426		
	PI	0,339213	31,47055	
	PID	1,356852	41,54113	10,51116

2.7 Regulační experimenty

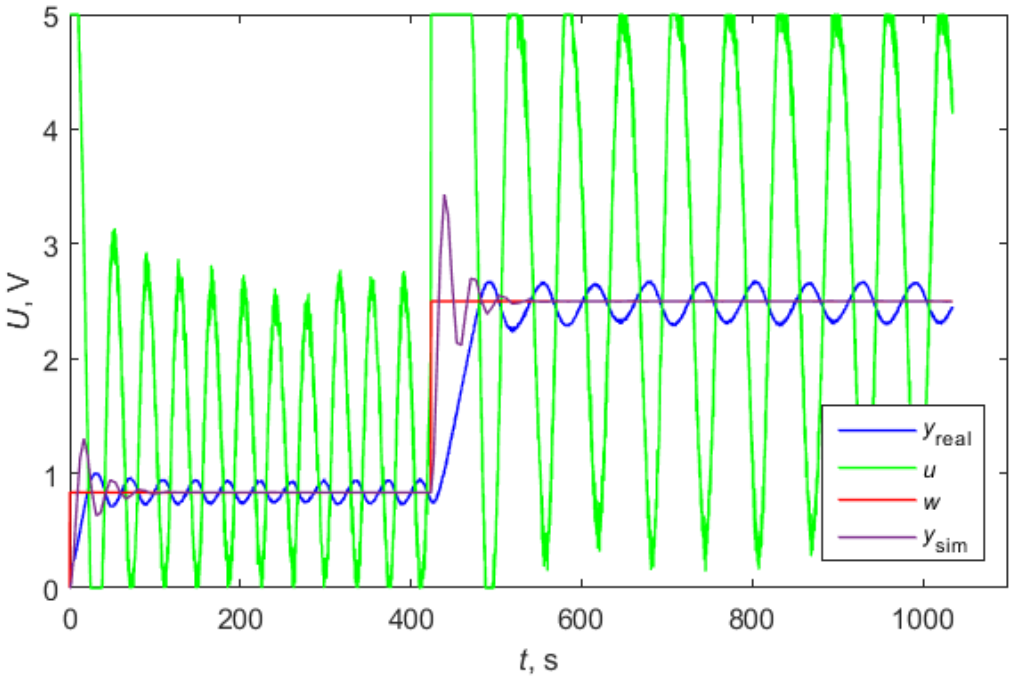
Regulaci hladiny nádrží jsem provedl při změně požadované hodnoty z 5 na 15 centimetrů, což odpovídá skoku napětí žádané hodnoty 0,85 na 2,5V. Při všech experimentech jsem použil PI regulátor. Regulaci jsem provedl na reálných nádržích i jako simulaci v SIMULINKu. Schéma simulace je vidět na obrázku 2.19. Blok *Transfer Function* jsem měnil

podle modelu nádrže, kterou jsem simulovaně reguloval. PID regulátor má nastavená omezení podle reálných zásahů.

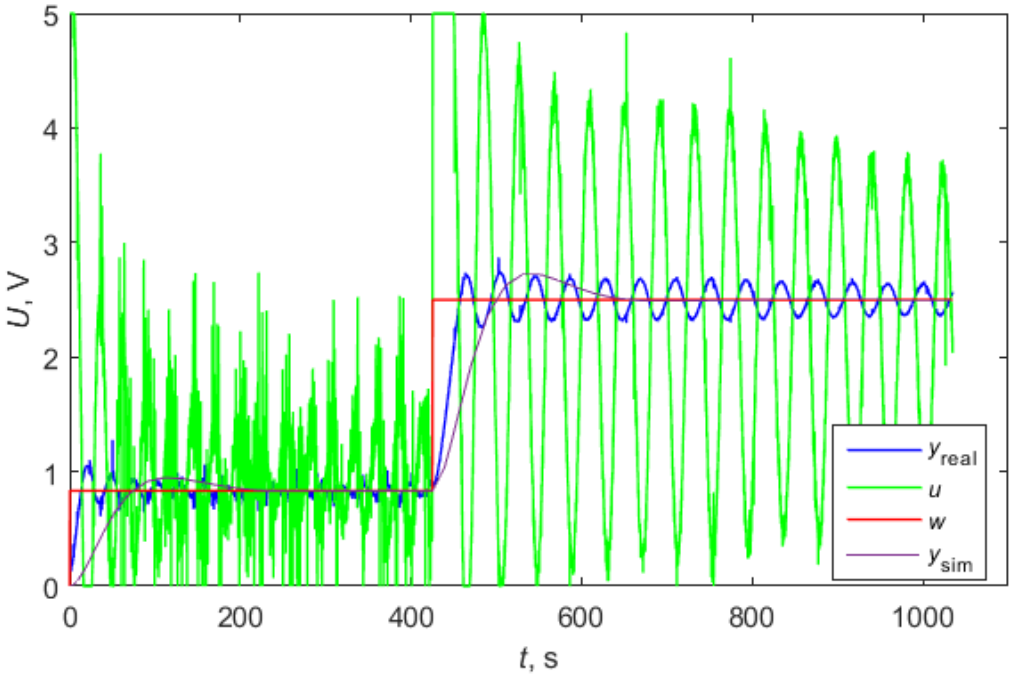


Obrázek 2.19 - Schéma simulované regulace

První regulaci jsem provedl s PI regulátorem nastaveným podle metody Ziegler-Nichols. U těchto regulací se reálná a simulovaná regulace neshodovala. U simulovaných byl jasný velký překmit a ustálení do 250 sekund, ale u reálné regulace byl překmit malý a regulace se neustálila. Proto je tato metoda regulace pro tyto nádrže nevhodná. Průběhy jsou na obrázku 2.20 a 2.21.

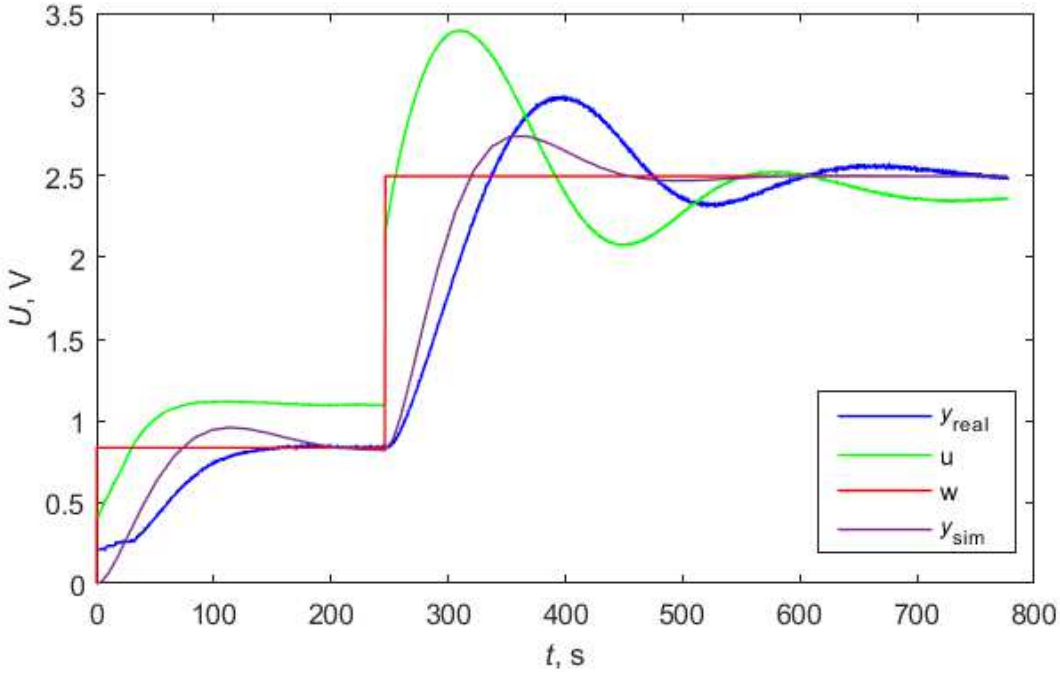


Obrázek 2.20 - Regulace levé nádrže metodou Ziegler-Nichols

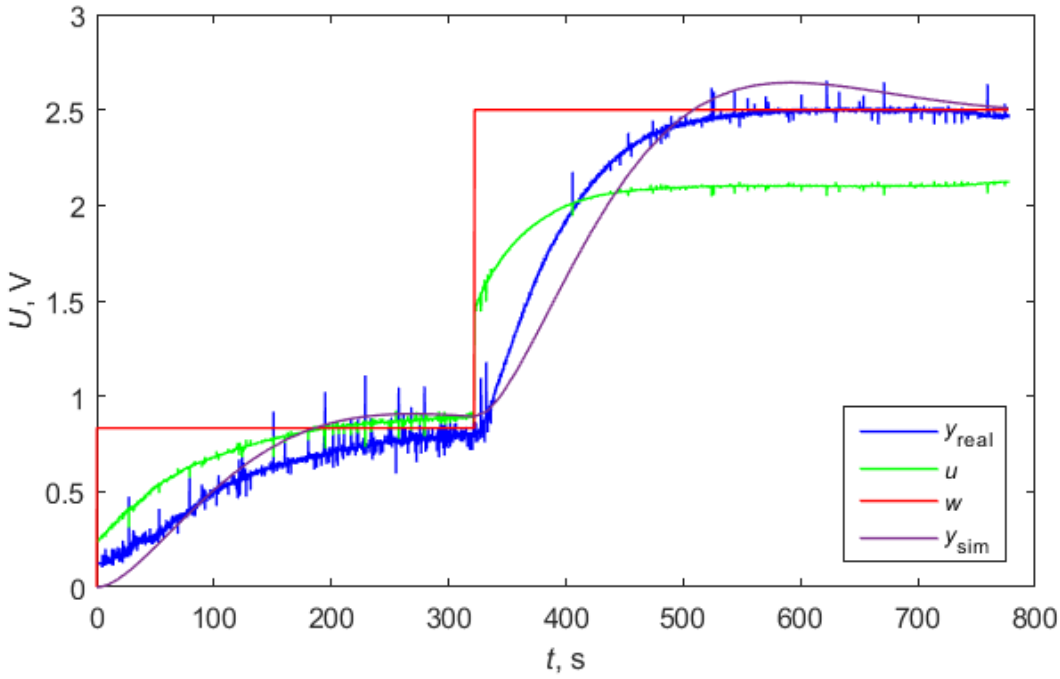


Obrázek 2.21 - Regulace pravé nádrže metodou Ziegler-Nichols

Druhou regulaci jsem provedl s PI regulátorem zjištěným Kuhnovou metodou pro normální rychlost regulace. Průběhy této regulace jsou velice podobné. U levé nádrže byl jeden velký překmit a soustava se po 500 sekundách ustálila. U pravé nádrže nebyl překmit žádný a regulace se ustálila do 300 sekund. Tato metoda nastavení regulátorů je velmi dobrá a přináší velmi dobré výsledky. Průběhy jsou na obrázku 2.22 pro levou nádrž a na obrázku 2.23 pro pravou nádrž.



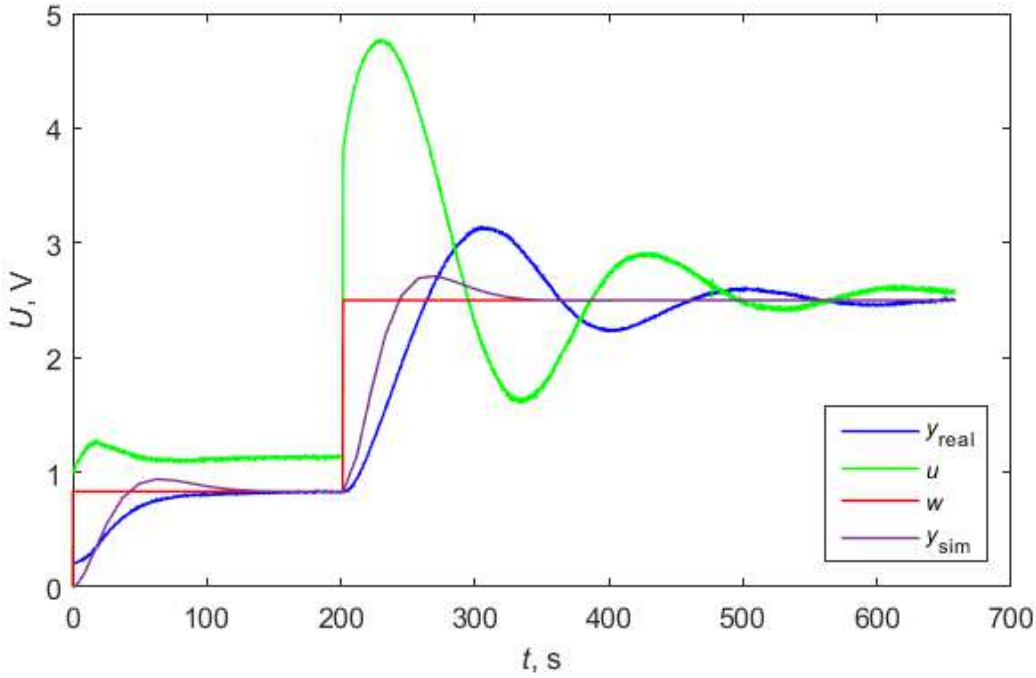
Obrázek 2.22 - Regulace levé nádrže Kuhnovou metodou (normální nastavení)



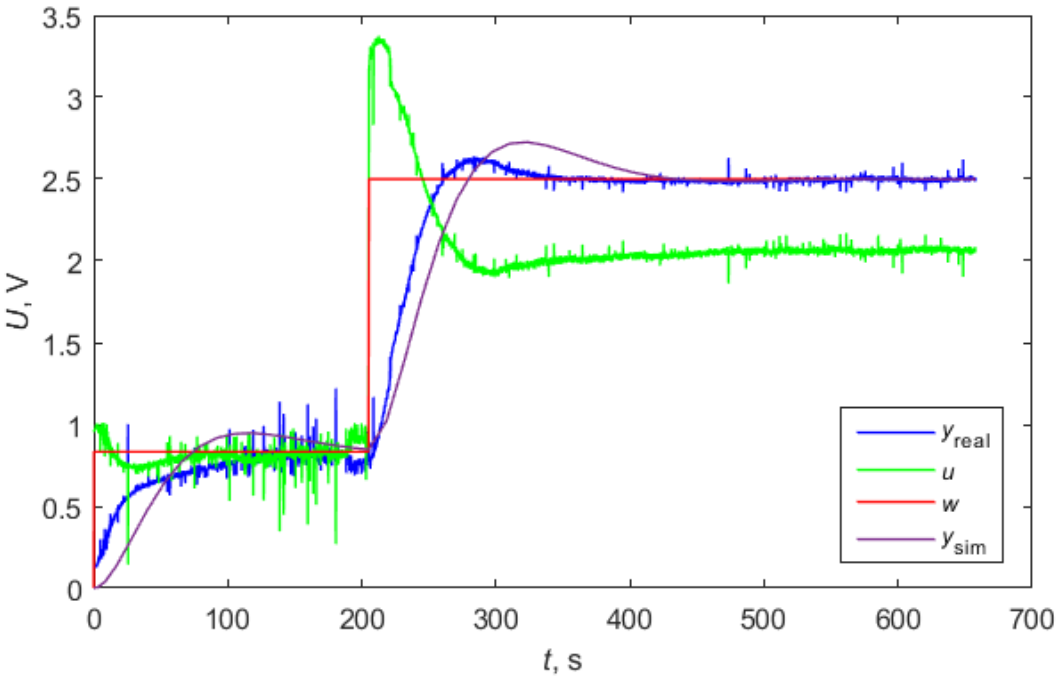
Obrázek 2.23 - Regulace pravé nádrže Kuhnovou metodou (normální nastavení)

Poslední regulaci jsem provedl s PI regulátorem nastaveným Kuhnovou metodou pro rychlou regulaci. U této metody se reálná a simulovaná regulace nijak nelišila. Regulace levé nádrže měla lehký první překmit a následně lehce podkmitla ale regulace se ustálila po 400 sekundách. Pravá nádrž jen lehce překmitla a regulace se ustálila do 200 sekund. Regulace

nádrží touto metodou je nejrychlejší s poměrně i malým překmitem. Průběh pro levou nádrž je na obrázku 2.24. Průběh pravé nádrž je na obrázku 2.25.



Obrázek 2.24 - Regulace levé nádrže Kuhnovou metodou (rychlé nastavení)



Obrázek 2.25 - Regulace pravé nádrže Kuhnovou metodou (rychlé nastavení)

3 ZÁVĚR

Cílem této práce bylo navrhnout a realizovat řídící jednotku pro připojení laboratorní soustavy k počítači pomocí platformy Arduino. Tato platforma je založena na jednočipovém mikropočítači, který má četná aplikační využití.

V teoretické části je uveden popis použitého hardware a způsobu programování jednočipových mikropočítačů a jejich využití. U hardwaru je popsána konstrukce a jednotlivé prvky od procesoru až po paměť. U programování jsou popsány jeho druhy a kdy se který využívá. Dále je zde uvedena teorie regulace systémů a regulační obvody. U PID regulátorů jsou popsány jejich složky a význam při regulaci. Na konci teoretické části je vysvětlen princip číslicového regulátoru a jsou popsány některé metody nastavování PID regulátoru.

V praktické části je popsána platforma Arduino. Jsou zde vypsány jeho parametry, funkce, rozhraní a periferie. Dále je zde popsán software MATLAB. Jsou vyzdviženy jeho přednosti a využití. Je zde popsán postup, jak jsem tvořil program a grafické rozhraní. Je zde popsána funkce a ovládání vytvořeného grafického rozhraní, ve kterém je možné ovládat soustavu manuálně, nebo automaticky. V manuálním režimu se ovládá výkon čerpadel. U automatického režimu je možné nastavit požadovanou výšku hladiny a jednotlivé parametry regulátoru. Je zde uveden návrh a tvorba přizpůsobovacích obvodů k Arduinu, aby bylo možné soustavu ovládat. Je provedena identifikace soustavy při skokové změně napětí na čerpadlech. Z těchto dat byly zjištěny modely soustav a byly pro ně vypočteny přechodové charakteristiky. Z přechodových charakteristik byly vypočítány hodnoty PID regulátorů jednotlivými metodami. Nakonec byla s vypočtenými parametry PI regulátorů řízena hydraulicko-pneumatická laboratorní soustava a výsledky regulace byly porovnány se simulovanými průběhy. Při těchto experimentech bylo zjištěno, že Kuhnova metoda v rychlém režimu dává nejlepší výsledky při regulaci této soustavy.

POUŽITÁ LITERATURA

BALÁTĚ, J. *Automatické řízení*. 2., přeprac. vyd. Praha: BEN - technická literatura, 2004. ISBN 80-7300-148-9.

CVEJN, J. 2015. *Automatizace I: Kap. 10 – Automatická regulace.* Přednáška. Pardubice Osobní sdělení.

CVEJN, J. 2015. *Automatizace I: Kap. 13 – Praktické metody nastavení PID regulátorů*. Přednáška. Pardubice Osobní sdělení.

DUŠEK, F. *MATLAB a SIMULINK: úvod do používání*. Vyd. 2., rozš. Pardubice: Univerzita Pardubice, 2002. ISBN 80-7194-475-0.

HANKOVEC, D. *Jak se naučit programovat? - I. díl* [online]. 2002 [cit. 2017-01-15]. Dostupné z: http://www.dhservis.cz/dalsi_1/popis.htm

HAVLÍČEK, L. 2016. *Mikroprocesorová technika 1. – 2. Přednáška*. Přednáška. Pardubice Osobní sdělení.

HONC, D. a DUŠEK, F. Novel multivariable laboratory plant. In: *26th European Conference on Modelling and Simulation*, Koblenz, Germany, May 29 – June 1 2012, ECMS – European Council for Modelling and Simulation, 2012, pp. 468-473, ISBN 978-0-9564944-4-3

HONC, D. 2016. *Automatizace 2*. Přednáška. Pardubice: Osobní sdělení.

HORÁČEK, O. *Arduino základy – 6. Regulace jasu LED* [online]. 2014 [cit. 2017-01-15]. Dostupné z: https://arduino.cz/arduino-zaklady-6-zhasinani/

KEIM, R. *Low-Pass Filter a PWM Signal into an Analog Voltage* [online]. 2016 [cit. 2017-01-15]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/low-pass-filter-a-pwm-signal-into-an-analog-voltage/

PINKER, J. *Mikroprocesory a mikropočítače*. Praha: BEN - technická literatura, 2004. ISBN 80-7300-110-1.

TAUFER, I.; KOTYK, J.; JAVŮREK, M. *Jak psát a obhajovat závěrečnou práci: bakalářskou, diplomovou, rigorózní, disertační, habilitační. 2.,* dopl. a opr. vyd. Pardubice: Univerzita Pardubice, 2014, 47 s. ISBN 978-80-7395-746-9.

VODA, Z. *Programujeme Arduino* [online]. 2014 [cit. 2017-04-05]. Dostupné z: https://arduino.cz/programujeme-arduino/

VODA, Z. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB pro začátečníky*. 2. vyd. Praha: BEN - technická literatura, 2005. ISBN 80-7300-175-6.

PŘÍLOHY

A – CD

Příloha A

Příloha k bakalářské práci

Řídicí jednotka hydraulicko-pneumatické laboratorní soustavy

Petr Linhart

CD

Obsah