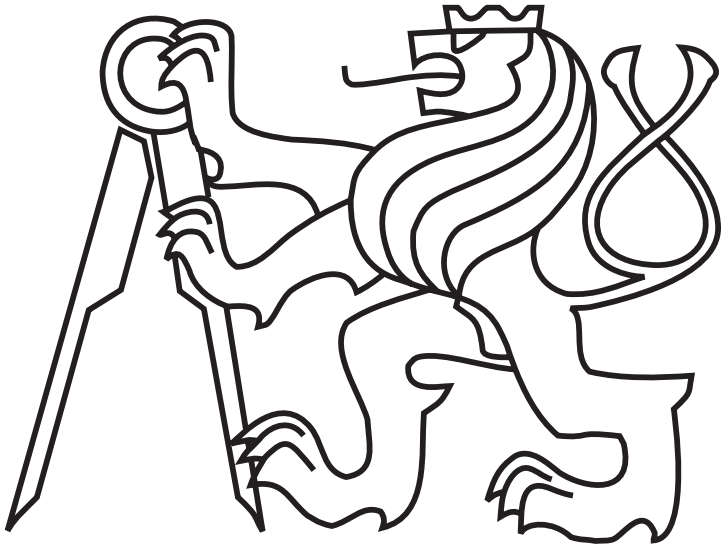
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

Katedra řídicí techniky



BAKALÁŘSKÁ PRÁCE

Klasické i moderní řízení spínaných měničů

Vedoucí práce: Ing. Zdeněk Hurák, Ph.D.

Praha, 2011

Autor: Petr Adámek

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Petr Adámek**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Klasické i moderní řízení spínaných měničů**

Pokyny pro vypracování:

1. Navrhněte a realizujte 2 - 3 základní zapojení spínaných měničů (buck, boost, buck-boost, ...), a to ve formě uživatelsky příjemných (výukových) laboratorních přípravků (napojení na převodníkovou kartu na PC pro snadné monitorování a řízení).
2. Na hotových přípravcích bude předvedeno použití základních a případně i pokročilejších metod pro zpětnovazební návrh řízení. Na tuto BP může navazovat DP, která se bude soustředit již výhradně na použití moderních metod řízení pro spínané měniče.

Seznam odborné literatury:

[1] C. Basso, Switch-Mode Power Supplies Spice Simulations and Practical Designs, McGraw-Hill Professional, 2008.
[2] R.W. Erickson and D. Maksimovic, Fundamentals of Power Electronics, Springer, 2001.
[3] A. Pressman, K. Billings, and T. Morey, Switching Power Supply Design, 3rd Ed., McGraw-Hill Professional, 2009.

Vedoucí: Ing. Zdeněk Hurák, Ph.D.

Platnost zadání: do konce zimního semestru 2011/2012

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 11. 10. 2010

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne 23. 5. 2011 .....

.....

podpis

Na tomto místě bych rád poděkoval Ing. Zdeňku Hurákovi, Ph.D. za odborné vedení a cenné rady při vypracovávání této bakalářské práce. Dále bych chtěl poděkovat dalším pracovníkům katedry řídicí techniky za technickou podporu. V neposlední řadě bych rád poděkoval svým rodičům za podporu během studia.

Abstrakt

Úkolem této práce je sestavit několik spínaných DC/DC měničů a navrhnout pro ně klasické regulátory. DC/DC měniče představují v této práci dva základní měniče, snižující měnič a zvyšující měnič. Klasické regulátory jsou dvoupolohový regulátor a číslicový PID regulátor. K řízení měničů je použit mikroprocesor ATmega8, který komunikuje pomocí sériové linky s počítačem. Zátěž měniče představuje proměnný lineární odpor, který simuluje změnu zatížení měniče. Výsledkem práce je porovnání vlivu různých typů regulátorů na výstupní napětí měniče při různých hodnotách zátěže.

Klíčová slova

Snižující měnič, zvyšující měnič, dvoupolohový regulátor, PID regulátor, PSD regulátor.

Abstract

The task of this bachelor thesis is built several switching DC / DC converters and propose them classical controllers. DC/DC converters represent two basic converters of this work - balk converter and boost converter. Classical controllers are two - position controller and digital PID controller. Classical and modern control for switching power converters is taken using the microprocessor ATmega8, which communicates through a serial line with a computer. The linear potentiometer represents load of converter. Results of this work describe the influence of various types of controllers on output voltage of converter during various loads.

Key words

Buck converter, boost converter, two - position controller, PID controller, PSD controller.

Obsah

1 Úvod

1.1 Cíl a motivace

Cílem této bakalářské práce je realizovat dvě základní zapojení DC/DC měničů, ve formě výukových přípravků. Tyto přípravky bude možné napojit pomocí převodníku k počítači, který bude sloužit jako nadřazená jednotka řídicímu mikroprocesoru. Počítač slouží pro výběr a nastavení regulátoru a zároveň monitoruje činnost měniče. Hlavním úkolem je porovnat vliv různých regulátorů na kvalitu a přesnost regulace.

1.2 Napěťové měniče DC/DC

Napěťové měniče můžeme rozdělit do několika kategorií. Za hlavní rozdělení můžeme považovat měniče s lineárními prvky a spínané měniče.

Měniče s lineárními prvky jsou měniče, které jsou používány k snižování vstupního napětí. Projevují se nižší účinností než spínané měniče. Nevýhodou lineárních měničů je neschopnost zvýšit vstupní napětí a nemožnost konverze napětí vzhledem ke společnému vodiči (zemi). Hlavní výhodou těchto napěťových měničů je jednoduchost zapojení a nízká cena.

Spínané měniče obsahují ve svém zapojení alespoň jeden spínaný prvek. Vlastnosti měniče jsou závislé na zapojení. Spínané měniče mohou vstupní napětí snižovat, zvyšovat nebo konvertovat vůči společnému vodiči. Nutnou podmínkou k vlastní činnosti měniče je stejnosměrné vstupní napětí, které je pomocí spínacího prvku převedeno na obdélníkový signál, který je usměrněn a vyfiltrován na výstupní napětí. Výstupní filtr musí po dobu, po kterou je výstupní filtr odpojen od napájecího zdroje, dodávat energii do zátěže. Tato energie se ve výstupním filtru akumuluje v době, kdy je výstupní filtr připojen ke zdroji napětí.

$$P_{lin} = U_s I_s \tag{1}$$

$$P_{imp} = U_s I_s k \tag{2}$$

$$k = \frac{T_1}{T_1 + T_2} = \frac{T_1}{T} \leq 1 \tag{3}$$

, kde P_{lin}- ztrátový výkon lineárních měničů, P_{imp}- ztrátový výkon impulsních měničů, U_s- napětí na akčním členu, I_s- proud protékající akčním členem, k - pracovní činitel (střída $0 \leq k \leq 1$)[1].

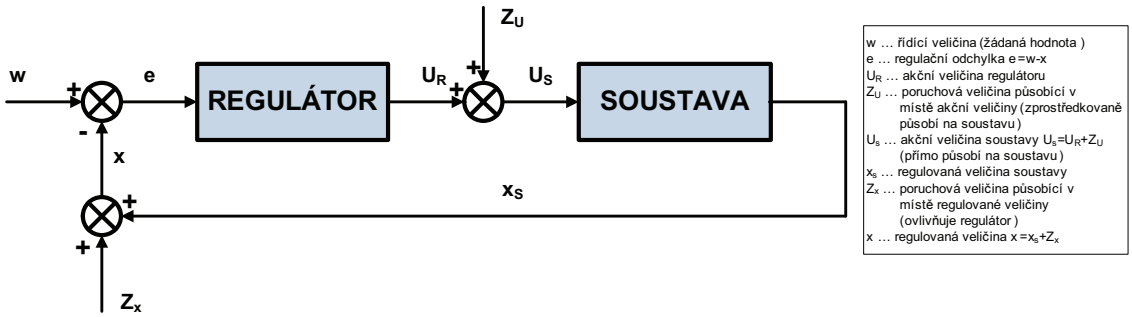
Porovnáním ztrát u lineárních a spínaných měničů je zřejmá výhoda spínaných měničů. U lineárních měničů je k udržení výstupního napětí potřeba akční člen, který je realizován řízeným rezistorem, protože se jedná o spojitý měnič, je na tomto rezistoru stálý úbytek napětí a stálý ztrátový výkon (1). Naopak u spínaných měničů je akční člen realizován pomocí spínače (tranzistoru), který je provozován v saturačních hodnotách. V sepnutém stavu má tranzistor malý vnitřní odpor a úbytek napětí na spínacím

prvku je menší než u akčního členu lineárních měničů. Navíc je spínač sepnut pouze část řídicího cyklu což má za následek další zmenšení ztrátového výkonu (2) na akčním členu.

1.3 Řízení spínaných měničů

Řízení spínaného měniče probíhá v prostřednictvím spínacího prvku, kterým je nejčastěji tranzistor typu MOSFET v závislosti na frekvenci spínání.

Pokud máme stabilní vstupní napětí, zátěž se mění v malém rozsahu a máme požadavek na stálé výstupní napětí, tak můžeme pro řízení použít frekvenci spínání s pevně stanovenou střídou. Výstupního napětí měniče je funkcí frekvence spínání a vstupního napětí v závislosti na typu zapojení. Nebo můžeme zavést zápornou zpětnou vazbu (Obrázek 1), ve které je definován požadavek na výstupní napětí a pomocí regulátoru měníme frekvenci spínání nebo střídu při stálé frekvenci.



Obrázek 1: Regulační obvod.

1.4 Použití spínaných měničů

V dnešní době se se spínanými měniči běžně potkáváme například ve spínaných zdrojích, v aplikacích využívající stejnosměrné napájení, v aplikacích, které dobíjejí akumulační články. Dále se s měniči můžeme setkat například u fotovoltaických článků[2], kde je potřeba zvýšit napětí získané napětí. DC/DC měniče mají své uplatnění i ve výkonové elektronice, kde mohou sloužit jako zdroje pro stejnosměrné motory.

1.5 Struktura bakalářské práce

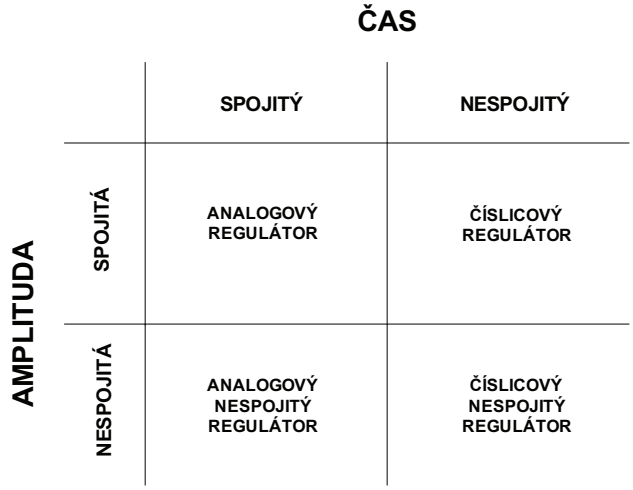
Práce je rozdělena hierarchicky rozdělena do 8 kapitol, v první kapitole jsme se dozvěděli základní rozdělení měničů a možnosti jejich řízení a jejich použití. Druhá kapitola se věnuje základnímu principu nespojitých regulátorů, které je možné použít k řízení měničů. Třetí kapitola pojednává stručně o možné realizaci. Ve čtvrté a páté kapitola se práce věnuje podrobně jednotlivým měničům, jejich popisu a návrhu regulátorů. Šestá kapitola vyhodnocuje výsledky bakalářské práce. Sedmá kapitola je

věnována zdrojům, ze kterých bylo čerpáno v této práci. V osmé kapitole jsou uvedeny návrhy plošných spojů a zdrojové kódy programů.

2 Nespojité regulátory

2.1 Rozdělení regulátorů z hlediska spojitosti

Regulátory můžeme rozdělit do několika kategorií (Obrázek 2). Spojité regulátory jsou pouze ty, které pracují ve spojitém čase a pracují se spojitými signály, představiteli takových regulátorů jsou PID regulátory tvořené operačními zesilovači. Číslicovými regulátory rozumíme regulátory, které jsou spojité v amplitudě, ale jsou nespojité v čase. Číslicové regulátory musejí být tvořeny A/D a D/A převodníky. Regulátory, které jsou spojité v čase a nespojité v amplitudě označujeme jako nespojité regulátory, tyto regulátory jsou tvořeny analogovými částmi stejně jako spojité regulátory, ale pracují s nespojitými veličinami. Nejčastěji se akční zásah regulátoru mění skokově. Regulátory, jenž jsou nespojité jak v čas tak i v amplitudě, označujeme za nespojité regulátory, u nichž se výpočty provádějí v číslicové části. Regulátor opět obsahuje A/D a D/A převodníky, při čemž alespoň jeden ze signálů se kterým pracuje se mění skokově.

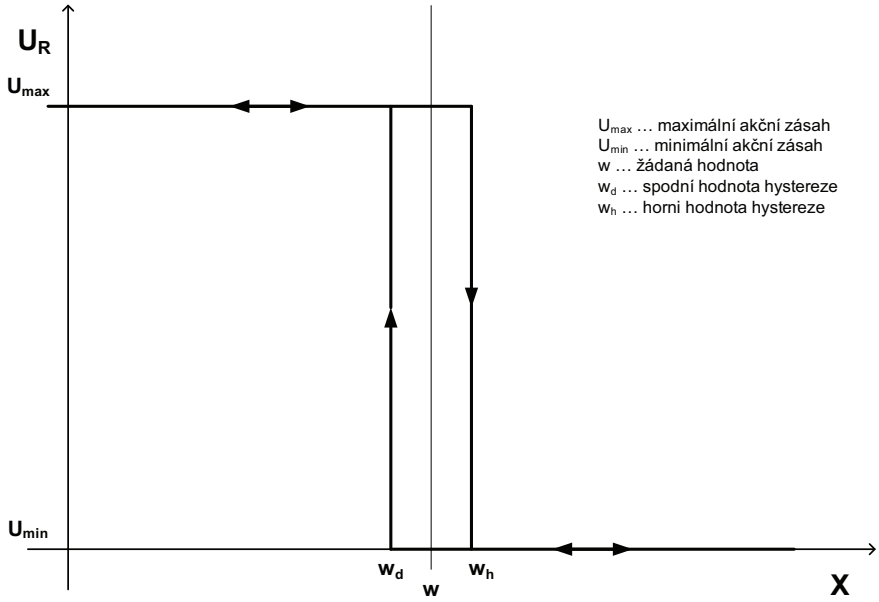


Obrázek 2: Výběr regulátoru podle spojitostí.

Řízení měniče neprobíhá spojitě, proto regulátor tohoto systému řadíme mezi nespojité regulátory. Tento nespojitý regulátor je charakteristický tím, že jeho výstup (akční veličina) nezávisí spojitě na vstupu (regulované veličině). Akční veličina se tedy nemění spojitě, ale nabývá pouze omezeného počtu hodnot. Změna z jedné hodnoty na druhou probíhá skokem, akční člen tedy může nabývat pouze dvou nebo více pevných poloh. Podle počtu těchto poloh rozdělujeme tyto regulátory na dvoupolohové, třípolohové a vícepolohové. Nespojité regulátory patří pro svou jednoduchou konstrukci a cenovou dostupnost mezi nejrozšířenější regulátory. Akční člen měniče (MOSFET) může nabývat pouze dvou hodnot stavu, kdy je spínací prvek sepnut, a stavu, kdy je spínací prvek vypnut. Tudíž se jedná pouze o dvoupolohovou regulaci.

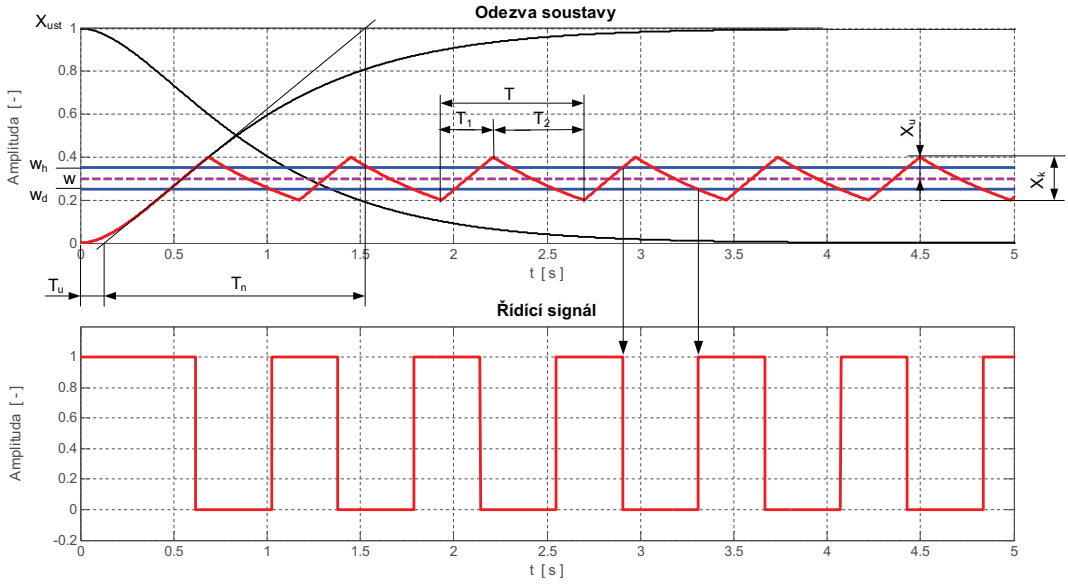
2.2 Dvoupolohový regulátor

Nejjednodušším nespojitým regulátorem je dvoupolohový regulátor. Klesne-li skutečná hodnota X pod hranici žádané hodnoty w, nabude akční zásah U_R maximální hodnoty (spínač sepne). Naopak pokud skutečná hodnota překročí hodnotu žádanou, nabude akční zásah minimální hodnoty (spínač rozepne). V takovém to regulátoru by však docházelo k častému spínání, což by mělo za následek rychlé opotřebení spínacího prvku, proto se dvoupolohová regulace doplňuje o pásmo hystereze. Zvětšení pásma hystereze má bohužel za následek zvětšení nepřesnosti regulované veličiny. Vlastnosti dvoupolohové regulace můžeme vyjádřit pomocí statické charakteristiky (Obrázek 3).



Obrázek 3: Statická charakteristika dvoupolového regulátoru.

Soustava měniče je tvořena dvěma akumulátory energie, tudíž hovoříme o dvoukapacitní soustavě. U jednokapacitní soustavy má na řízení výstupu vliv pouze hystereze (viz předchozí odstavec). U dvoukapacitní soustavy se kromě hystereze uplatní především vlastnosti soustavy. Výstup dvoukapacitní soustavy (Obrázek 4) nekolísá pouze v pásmu hystereze, ale výstup nabývá většího rozsahu než je hystereze. Tento jev je způsoben tím, že se regulovaná veličina nezačne okamžitě zmenšovat, i když dosáhne hodnoty w_h a akční veličina se vypne (je nulová), ale dále se zvětšuje. Je to způsobeno zpožděním v soustavě, které je dáno velikostí doby průtahu T_u. Teprve po uplynutí této doby se začne regulovaná veličina zmenšovat a při dosažení hodnoty w_d se sice akční veličina znovu zapne, ale zmenšování regulované veličiny pokračuje dál v důsledku zpoždění v soustavě. Z výkladu je zřejmé, že podstatný vliv na šířku pásma kmitání regulované veličiny X_k, a tím i na kvalitu regulačního pochodu, má v tomto případě regulovaná soustava, především její doba průtahu T_n. Hystereze regulátoru se naopak příliš neuplatní, neboť ke kmitání regulované veličiny dojde i tehdy, když je hystereze nulová[3].



Obrázek 4: Dvoupolohová regulace s dvoukapacitní soustavou.

Pro odvození vztahů, které popisují vlastnosti dvoupolohové regulace dvoukapacitní soustavy, vycházíme z podobnosti trojúhelníků. Tímto způsobem můžeme dopočítat frekvenci kmitů (6) a rozkmit regulované veličiny (7). Danné vztahy platí pro předpoklad $X_{ust} \geq 2w$.

$$\frac{T_1}{X_k} = \frac{T_n}{X_{ust}} \Rightarrow T_1 = T_n \frac{X_k}{X_{ust}} \tag{4}$$

$$\frac{T_2}{X_k} = \frac{T_n}{w} \Rightarrow T_2 = T_n \frac{X_k}{w} \tag{5}$$

$$f = \frac{1}{T} = \frac{1}{T_1 + T_2} = \frac{X_{ust} w}{T_n X_k \left(X_{ust} + w\right)} \tag{6}$$

$$\frac{T_u}{X_u} = \frac{T_n}{w} \Rightarrow X_u = \frac{T_u}{T_n} w$$

$$X_k = h + 2X_u = h + 2\frac{T_u}{T_n} w = h + X_{ust} \frac{T_u}{T_n} \tag{7}$$

Průběh dvoupolohové regulace s dvoukapacitní soustavou (Obrázek 4) je teoretický, ve skutečnosti jsou přechody více zaobleny a rozkmit je proto menší. Ze vzorce (7) si můžeme všimnout, že zvětšující se pásmo hystereze h a zvětšení doby průtahu T_u má za následek zvětšení rozkmitu X_k. Naopak zvětšení doby náběhu T_n rozkmit X_k zmenšuje.

Způsoby zvyšování kvality regulace:

- zmenšení hystereze - rychlejší spínání (zvýšení četnosti spínání)
- zkrácení doby průtahu - odstranit vliv rychlosti samovolného působení energie (u tepelných soustav např. promíchávání)

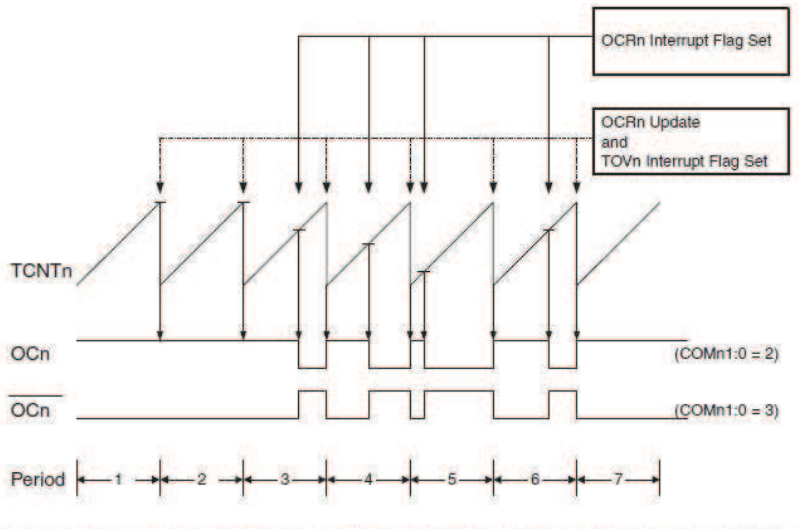
- zvětšení doby náběhu - úprava soustavy, která neovlivní dobu průtahu, ale jen zvětší setrvačnou kapacitu soustavy; zvětšíme zátěž soustavy
- použití vhodného akčního členu

2.3 Impulsní regulátor

Do skupiny nespojitých regulátorů můžeme zahrnout i impulsní regulátory. Impulsní regulátor je charakteristický tím, že na jeho výstupu jsou impulsy, jejichž střída nebo frekvence je závislá na regulační odchylce. Výstup impulsního regulátoru se chováním blíží analogovému (spojitému), protože pracovní frekvence bývá řádově vyšší než maximální frekvence soustavy, uplatní se tedy jen střední hodnota výstupu.

Impulsní regulátor získáme zavedením zpožďující zpětné vazby dvoupolového regulátoru. Jak již bylo řečeno mění se střída nebo frekvence signálu, nejčastějším řízením je řízení pomocí PWM.

Při návrhu PWM regulace postupujeme stejně jako při návrhu klasických regulátorů. Akční veličina regulátoru musí být převedena na PWM (Obrázek 5). Výstupem regulátoru je střída PWM signálu, která je na PWM signál převedena pomocí pilovitého průběhu.



Obrázek 5: Převod střídy na PWM signál v mikroprocesoru ATmega8.[4]

2.4 Číslicový PID regulátor

Číslicový PID regulátor můžeme odvodit ze spojitého PID regulátoru (8) několika metodami. Jednou metodou je nahradit integrál v integrační složce pomocí lichoběžníkové metody za sumu regulačních odchylek a derivaci nahradit pomocí diference dvou po sobě následujících kroků, které vydělíme periodou vzorkování T_{vz}. Takovýto regulátor nazýváme PSD (9).

$$u(t) = k_P \left[e(t) + \frac{1}{T_I} \int_0^t e(\tau)\, d\tau + T_D \frac{de(t)}{dt} \right] \tag{8}$$

$$u(k) = k_P \left\{ e(k) + \frac{T_{vz}}{2 \cdot T_I} \sum_{j=0}^{k} e(k) + \frac{T_D}{T_{vz}} \left[e(k) - e(k-1) \right] \right\} \tag{9}$$

Klasický PID regulátor je v dnešní době v praxi nejpoužívanějším regulátorem, jeho obdoba PSD regulátor je používána ve většině řídicích systémů. Jeho nevýhodou ovšem je, že obsahuje sumu všech regulačních odchylek z tohoto pohledu je mnohem výhodnější použít jeho přírůstkový tvar. Přírůstkový tvar získáme posunutím celé rovnice PSD regulátoru o jeden krok zpět (10), odečtením rovnic (9) a (10) získáme přírůstkový tvar PSD regulátoru (11).

$$u(k-1) = k_P \left\{ e(k-1) + \frac{T_{vz}}{2T_I} \sum_{j=0}^{k-1} e(k-1) + \frac{T_D}{T_{vz}} \left[e(k-1) - e(k-2) \right] \right\} \tag{10}$$

$$\begin{aligned} u(k) - u(k-1) &= k_P \left\{ e(k) + \frac{T_{vz}}{2T_I} \sum_{j=0}^{k} e(k) + \frac{T_D}{T_{vz}} \left[e(k) - e(k-1) \right] \right\} \\ &\quad - k_P \left\{ e(k-1) + \frac{T_{vz}}{2T_I} \sum_{j=0}^{k-1} e(k-1) + \frac{T_D}{T_{vz}} \left[e(k-1) - e(k-2) \right] \right\} \end{aligned}$$

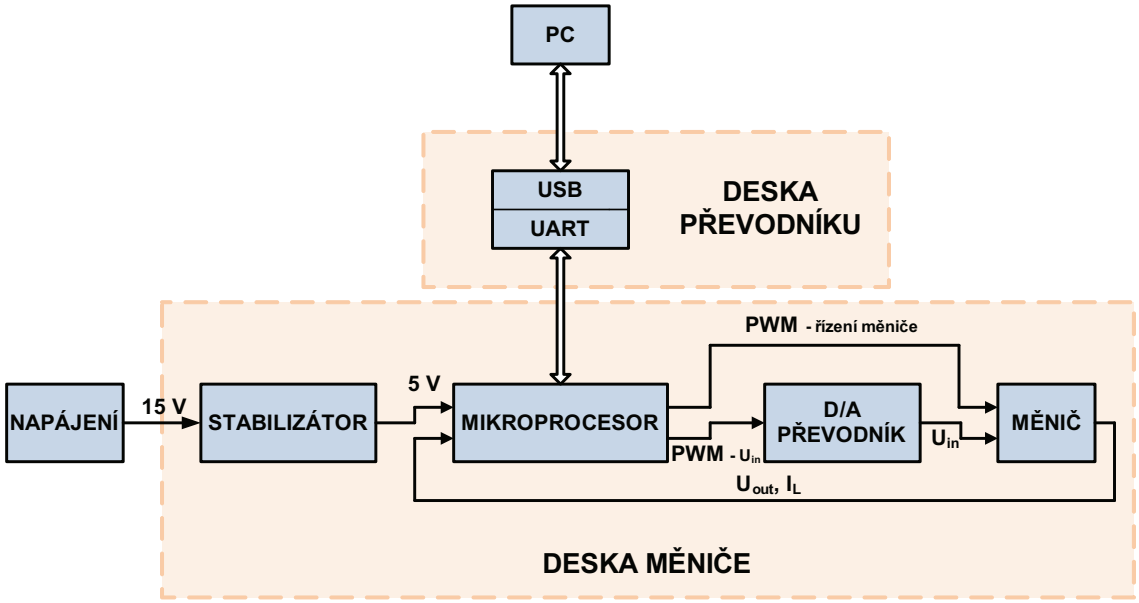
$$u(k) - u(k-1) = k_P \left\{ e(k) - e(k-1) + \frac{T_{vz}}{2T_I} e(k) + \frac{T_D}{T_{vz}} \left[e(k) - 2 \cdot e(k-1) + e(k-1) \right] \right\}$$

$$u(k) = k_P \left\{ e(k) - e(k-1) + \frac{T_{vz}}{2T_I} e(k) + \frac{T_D}{T_{vz}} \left[e(k) - 2e(k-1) + e(k-2) \right] \right\} + u(k-1) \tag{11}$$

3 Hardwarová a softwarová realizace

3.1 Komunikace

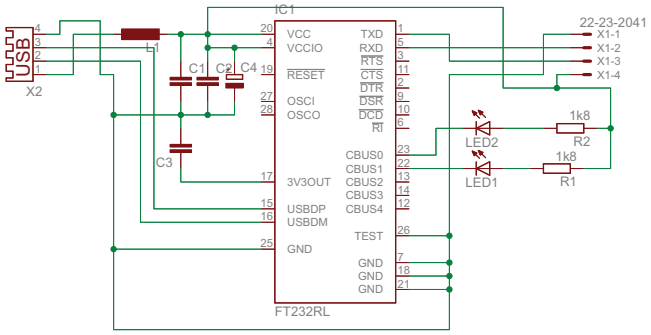
Abychom nemuseli řídicí mikroprocesor programovat pokaždé, když chceme změnit typ regulace nebo konstanty regulátoru. Nebo nechceme-li výsledné průběhy a data zjišťovat pomocí dalších přístrojů, které bychom připojili k obvodu, zavedeme pro pohodlnější zobrazení výsledků komunikaci mezi řídicím mikroprocesorem a počítačem (Obrázek 6).



Obrázek 6: Ideové schéma zapojení.

Mikroprocesor ATmega8 je vybaven sériovou linkou (USART), kterou pomocí převodníku USB/UART (Obrázek 7) můžeme připojit k USB portu počítače. USB port je zvolen kvůli snadné dostupnosti v dnešních počítačích. Po připojení převodníku k počítači je vytvořen virtuální sériový port COM, který je přístupný jako běžný COM port.

Při komunikaci mezi mikroprocesorem a počítačem jsme pouze omezeni přenosovou rychlostí, kterou jsme nastavili na 9600 Bd. Tato rychlost nám sice nezajistí získání všech dat, které vzorkujeme, ale pokud si data pošleme vždy po ukončení předchozího přenosu, získáme tím přibližné průběhy dat, které by měly být jako orientační zcela dostačující. Pro přesnější průběhy je možné k měniči připojit další přístroje, které dokáží získat více dat.



Obrázek 7: Převodník USB/UART - schéma zapojení.

3.2 Popis bloků

Protože přípravky mají sloužit zkušebním a výukovým účelům realizujeme celý prvek podle ideového schématu (Obrázek 6). Tato idea je založena na parametrech, které značně ovlivňují chod měniče. Mezi tyto parametry budeme řadit vstupní napětí měniče U_{in} a zátěž R_Z.

Blok napájení, je pevné napětí 15V, které slouží k napájení mikroprocesoru přes blok stabilizátoru a napájení operačního zesilovače v bloku D/A převodníku.

Blok stabilizátor je tvořen jedinou součástkou, stabilizátorem napětí, jehož výstupem je napětí 5V, které slouží pro napájení mikroprocesoru a jako referenční napětí A/D převodníku.

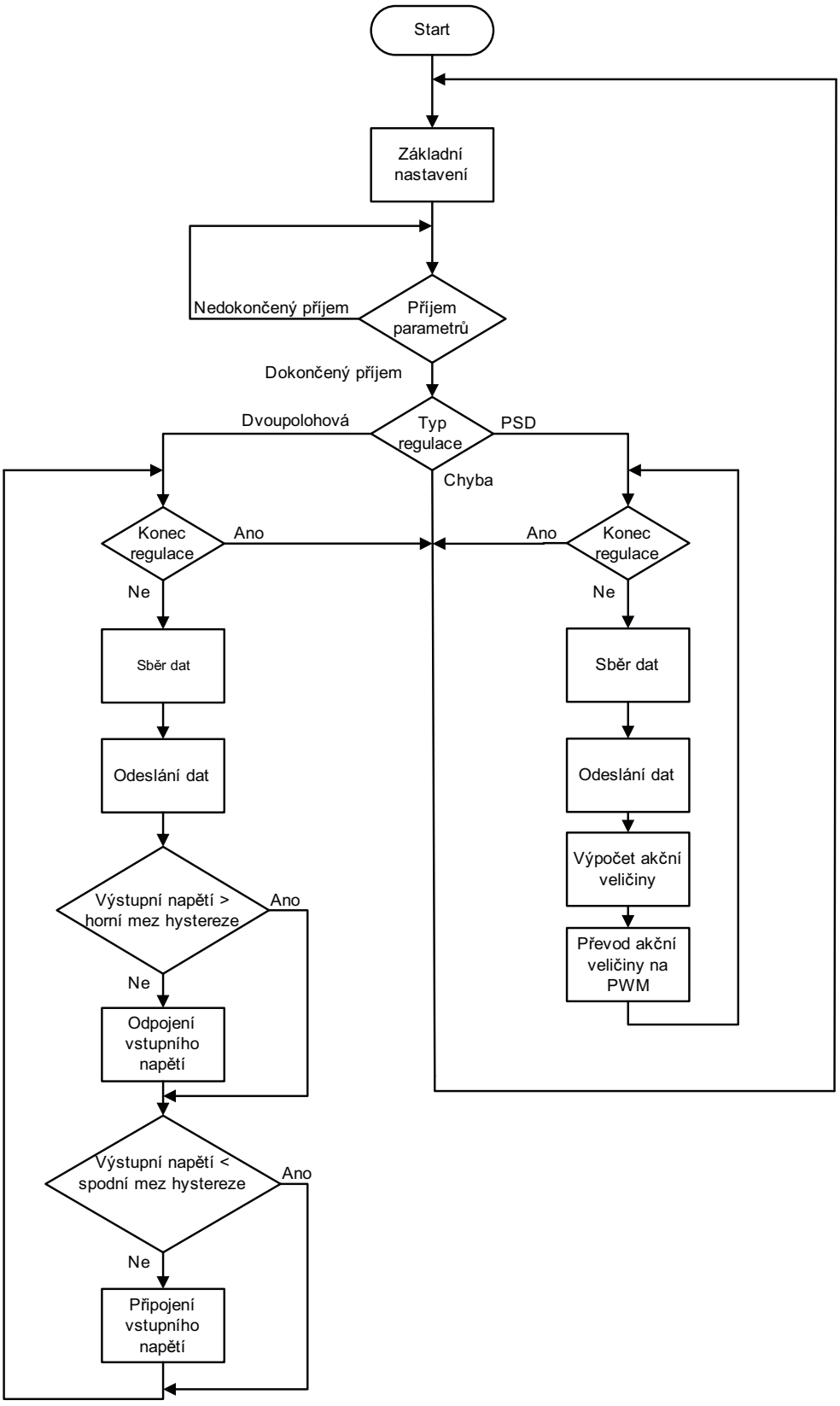
Blok mikroprocesoru je tvořen mikroprocesorem ATmega8, který umožňuje generovat PWM signály, převádět analogové signály na digitální a v neposlední řadě také komunikaci s PC. Jeden PWM signál slouží jako vstupní napájení U_{in} a druhé PWM slouží k řízení samotného měniče. Veličiny měniče, které chceme sledovat jsou převedeny pomocí integrovaného A/D převodníku. Všechny hodnoty, které sledujeme jsou odesílány do počítače pomocí sériového rozhraní mikroprocesoru (viz 3.1 Komunikace).

Blok D/A převodníku je realizován jako aktivní DP filtr 1. řádu, který převádí PWM signál na vstupní napětí měniče. Tento převodník využívá skutečnost, že výstupem převodníku je střední hodnota napětí vstupního PWM signálu, které může být adekvátně zesílena zesilovačem.

Blok měniče bude podrobě popsán v následujících kapitolách.

3.3 Řídicí mikroprocesor

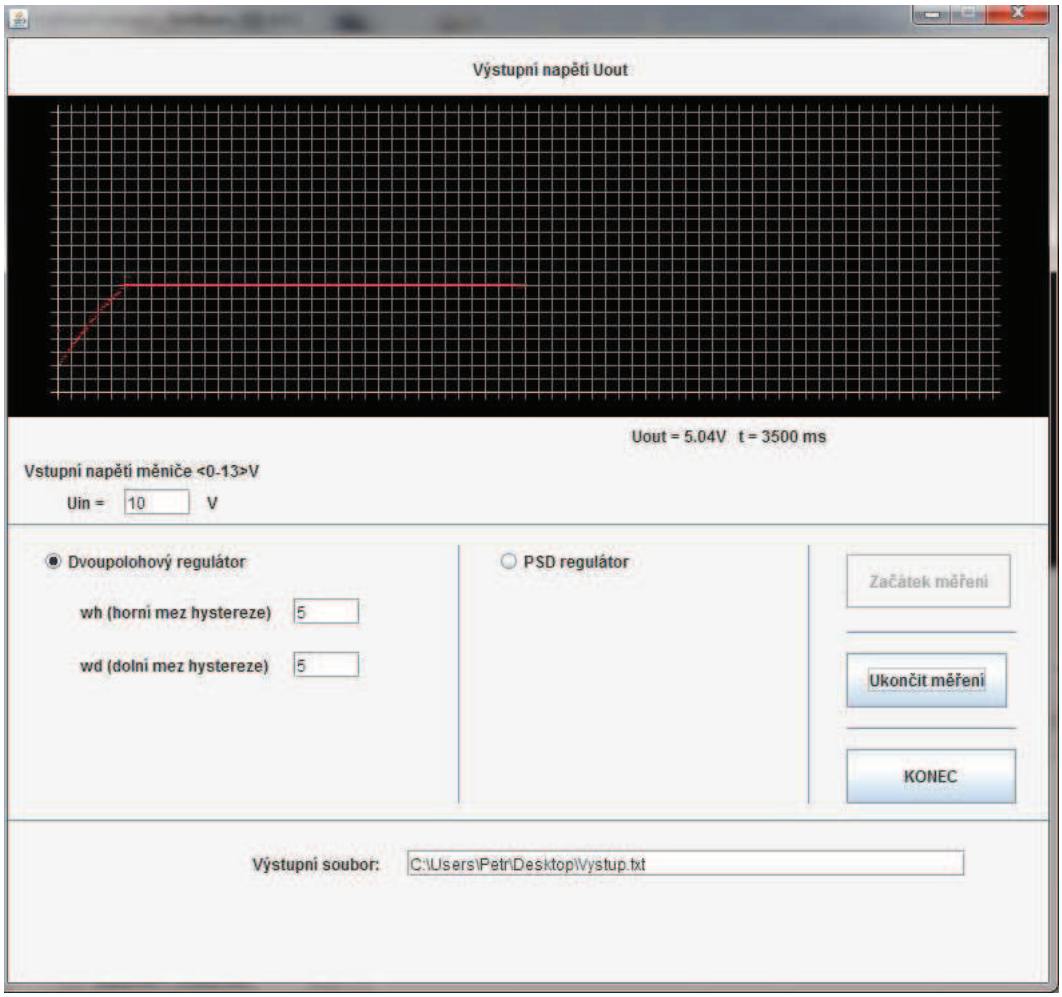
Řídicí mikroprocesor musí obstarat nejen řízení měniče, ale zároveň musí komunikovat s nadřazenou jednotkou, kterou v našem případě představuje počítač. Program mikroprocesoru můžeme jednoduše popsat vývojovým diagramem (Obrázek 8).



Obrázek 8: Vývojový diagram.

3.4 Počítač

Počítač slouží k zadání typu regulace a parametrů regulátoru, je iniciátorem spojení, které vyvolá měření. Dále zobrazuje příchozí data do grafu, který slouží k přibližné orientaci naměřených dat, která jsou po skončení měření uložena do souboru. S uloženými daty v souboru je možné pracovat v další programech (Matlab, Excel).



Obrázek 9: Ukázka programu.

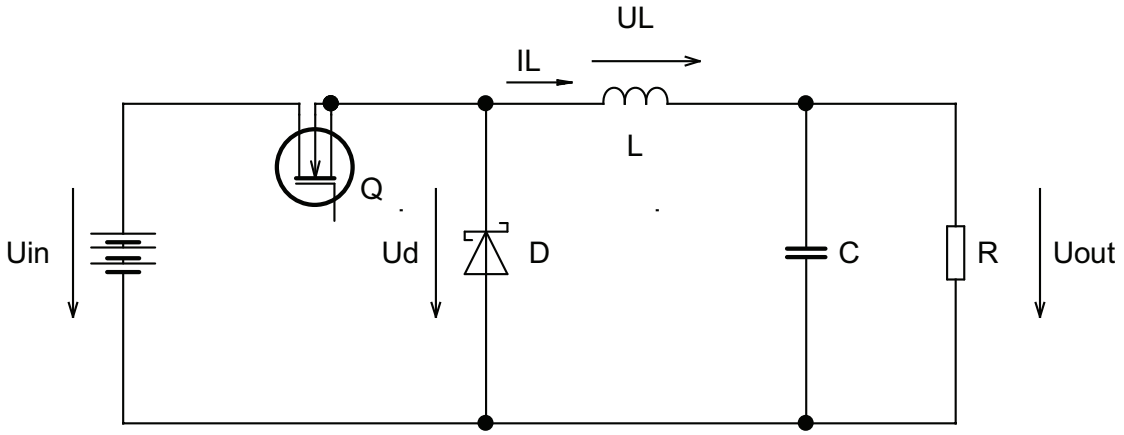
4 Snižující měnič

4.1 Popis funkce

Snižující měnič může být též nazýván step down nebo měnič typu buck (Obrázek 10) a řadíme jej mezi spínané měniče. Měnič můžeme popsat dvěma režimy. Režimu, kdy je spínací prvek sepnut, a režim, kdy je spínací prvek rozepnut. Spínacím prvkem rozumíme tranzistor Q.

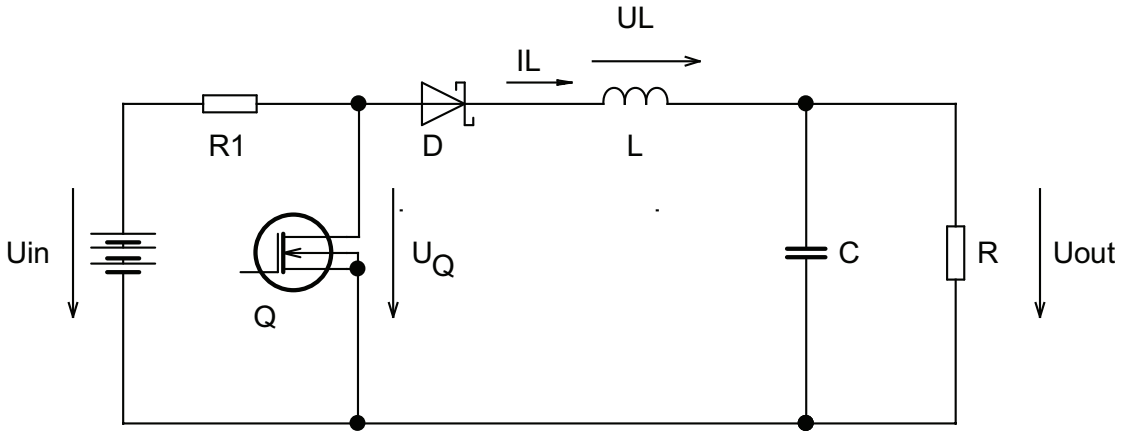
Pokud je tranzistor sepnut, je napájecí zdroj připojen k obvodu a proud je do zátěže dodáván přes cívku L. V tomto režimu se cívka chová jako spotřebič a je na ní úbytek napětí U_L. Zároveň se nabíjí kondenzátor C proudem I_C a napětí U_C se zvyšuje. Velikost výstupního napětí může dosáhnout maximálně vstupního napětí U_{in}.

Režim, kdy je tranzistor vypnut, má za následek odpojení vstupního napětí. Cívka se začne chovat jako zdroj a snaží se udržet směr a velikost proudu I_L, polarita napětí U_L se tedy obrátí. Zároveň se k tomuto proudu přičte proud kondenzátoru C, který se tak začne vybíjet. Výstupní napětí v tomto režimu klesá. Budeme-li zvyšovat kapacitu kondenzátoru C, snížíme tím zvlnění výstupního napětí[5].



Obrázek 10: Snižující měnič - základní zapojení.

Nevýhodou zapojení (Obrázek 10) je plovoucí spínací prvek, abychom se mu vyhnuli, můžeme použít pozměněné zapojení (Obrázek 11). V novém zapojení fungují obdobné principy, které jsou popsány výše. Rozdílem mezi zapojeními je spínací logika, pokud je spínací prvek sepnut dojde k odpojení napájecího zdroje, naopak při rozepnutém spínacím prvku je napájecí zdroj k obvodu připojen.



Obrázek 11: Snižující měnič - upravené zapojení.

4.2 Odvození stavového popisu

Při odvozování stavového popisu budeme vycházet z upraveného zapojení snižujícího měniče (Obrázek 11). Úbytek napětí na diodě, kterou budeme považovat za ideální, zanedbáme. Stavové rovnice určíme pomocí diferenciálních rovnic, které popisují snižující měnič. Jako stavový popis zvolíme proud protékající cívkou a druhou veličinou výstupní napětí (napětí na kondenzátoru $u_C = u_{out}$) $[\ i_L \quad u_C\ ]^{\mathrm{T}}$. Vstup měniče nahradíme pomocí úbytku napětí na spínacím prvku Q, v ideálním případě dosahuje napětí na tranzistoru dvou hodnot $\{0; U_{in}\}$.

1. **Spínací prvek Q sepnut**

$$u_Q = 0 \tag{12}$$

$$i_L = \frac{1}{L}\int (u_Q - u_C)\,dt + i_0 \tag{13}$$

$$\dot{i}_L = \frac{1}{L}\,(u_Q - u_C) \tag{14}$$

$$-\,i_L + \frac{u_C}{R} + C\frac{du_C}{dt} = 0 \tag{15}$$

$$\dot{u}_C = \frac{i_L}{C} - \frac{u_C}{RC} \tag{16}$$

2. **Spínací prvek Q rozepnut**

$$u_Q = U_{in} \tag{17}$$

$$i_L = \frac{1}{L}\int (u_Q - u_C)\,dt + i_0 \tag{18}$$

$$\dot{i}_L = \frac{1}{L}\,(u_Q - u_C) \tag{19}$$

$$-\,i_L + \frac{u_C}{R} + C\frac{du_C}{dt} = 0 \tag{20}$$

$$\dot{u}_C = \frac{i_L}{C} - \frac{u_C}{RC} \tag{21}$$

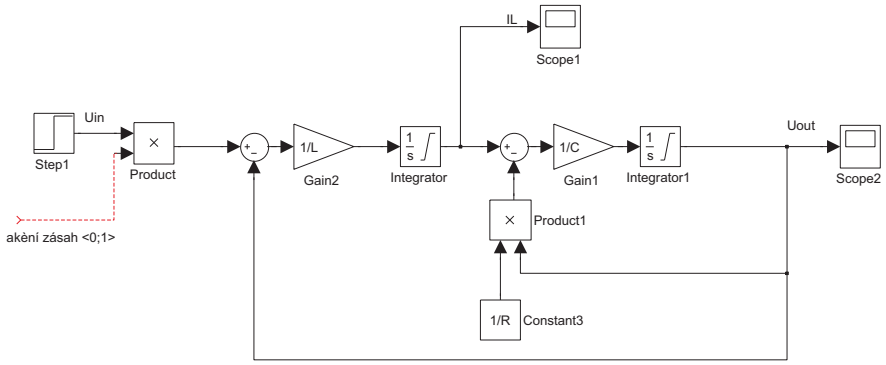
- Stavový popis - Stavový popis vytvoříme porovnáním rovnic (14) a (19), (16) a (21). Tyto rovnice jsou identické, až na velikost napětí U_Q , ačkoliv popisují dva stavy měniče, můžeme je tedy využít pro vytvoření stavového popisu pomocí matic (22). U_Q můžeme přepsat pomocí střední hodnoty (24) (viz 4.3 Řízení měniče).

$$\begin{bmatrix} \dot{i}_L \\ \dot{u}_C \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \cdot \begin{bmatrix} i_L \\ u_C \end{bmatrix} + \begin{bmatrix} -\frac{1}{L} \\ 0 \end{bmatrix} \cdot [u_Q]$$
$$[u_{out}] = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i_L \\ u_C \end{bmatrix} + [0] \cdot [u_Q] \tag{22}$$

- Přenos soustavy

$$G(s) = \frac{U_{out}(s)}{U_Q(s)} = \frac{R}{CLRs^2 + Ls + R} \tag{23}$$

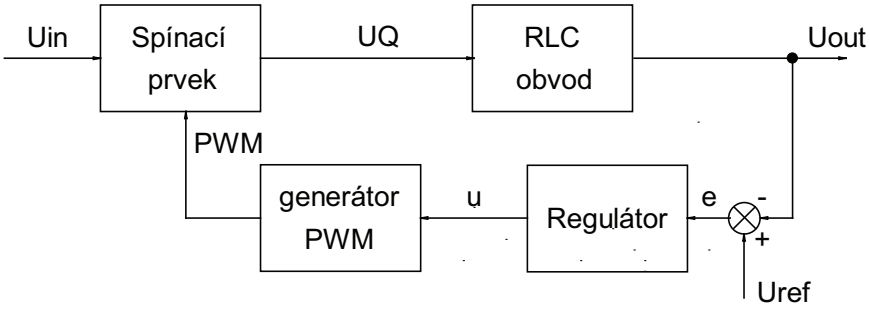
- Model soustavy



Obrázek 12: Simulinkové schéma snižujícího měniče.

4.3 Řízení měniče

Řízení měniče může probíhat pomocí nespojitých regulátorů, tedy dvoupolohovým regulátorem nebo impulsovým regulátorem.

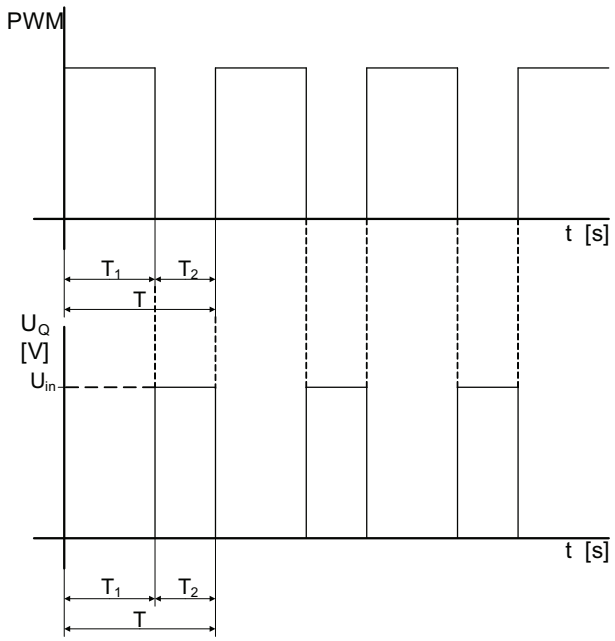


Obrázek 13: Blokové schéma měniče a řídicí části.

Při použití impulsového regulátoru s PWM je napětí na spínacím prvku U_Q možno popsat pomocí střední hodnoty vstupního napětí (24). Napětí na tranzistoru má obdélníkový průběh, výpočet střední hodnoty se tak omezí pouze na časový úsek, kdy se na spínacím prvku objeví vstupní napětí. Tento časový úsek odpovídá času $T_2 = T - T_1$.

$$U_Q = \frac{1}{T}\int_{T_1}^{T} U_{in}dt = \frac{1}{T} \cdot U_{in}\,(T - T_1) = \frac{T - T_1}{T} \cdot U_{in} = (1 - k) \cdot U_{in} \tag{24}$$

Tímto způsobem přejdeme z dvoustavového výstupu regulátoru na spojitý a následně provedeme návrh pro spojitou soustavu i spojitý regulátor, který převedeme na diskrétní (viz 4.4 Číslicový regulátor).



Obrázek 14: Ideální průběh napětí U_Q.

4.4 Návrh měniče

a) **Měnič**

Nejprve zvolíme prvky a veličiny, které budou pevně stanoveny.

$$U_{in} = 12V$$

$$U_{out} = 5V$$

$$f = 100kHz$$

$$I_Z = 0,5A$$

$$C = 10\mu F$$

Nyní stanovíme střídu k a z ní čas T_2, po který je připojen zdroj U_{in}.

$$U_{out} = (1-k)\cdot U_{in} \Rightarrow k = 1 - \frac{U_{out}}{U_{in}} = \frac{7}{12}$$

$$k = \frac{T-T_2}{T} \Rightarrow T_2 = T - k\cdot T = \frac{1-k}{f} = \frac{5}{12\cdot 10^5}s$$

$$I_1 = I_C + I_Z = 0,5A \ldots v\, ustáleném\, stavu\, I_C = 0$$

$$L = \frac{(U_{in}-U_{out})\cdot T_2}{I_1} = 58,3\mu H \approx 68\mu H$$

Tímto způsobem postupujeme při návrhu snižujícího měniče, protože však chceme použít číslicovou regulaci, musíme tedy dodržet vzorkovací větu (25), u které budeme vycházet ze šířky pásma.

$$f_{vz} = \frac{30\omega_{BW}}{2\pi} \tag{25}$$

Dosadíme-li do přenosu (22) vypočtené hodnoty prvků a z přenosu následně určíme šířku pásma získáme následující hodnoty (pro $R_Z = 1000\Omega$):

$$G(s) = \frac{10^{10}}{6,8s^2 + 680s + 10^{10}}$$

$$\omega_{BW} \approx 60\cdot 10^3\, rad\cdot s^{-1}$$

$$f_{vz} \approx 287\, kHz$$

Tato frekvence vzorkování je příliš vysoká, proto zvolíme součástky tak, abychom mohli vzorkovat s nižší frekvencí. Toho dosáhneme, pokud zvýšíme akumulační prvky:

$$R_Z = 1000\Omega$$

$$L = 470\mu H$$

$$C = 3300\mu F$$

$$G(s) = \frac{10^7}{15,51s^2 + 4,7s + 10^7}$$

$$\omega_{BW} \approx 1,3\cdot 10^3\, rad\cdot s^{-1}$$

$$f_{vz} \approx 6210\, Hz$$

S touto vzorkovací frekvencí jsme schopni již dále pracovat, proto zvolíme tyto parametry pro návrh a realizaci (Tabulka 1).

Tabulka 1: Parametry měniče

Veličina	Hodnota
L	$470\mu H$
C	$3300\mu F$
R_Z	$(0,33 \div 10,33)\, k\Omega$
U_{in}	$12V$
U_{out}	$5V$
f_{vz}	$10kHz$

b) **Regulátor**

Regulátory měniče navrhneme s určitými požadavky, chceme aby výstupní napětí bylo co nejméně zvlněné a zároveň chceme, aby odezva systému byla co možná nejrychlejší.

- Dvoupolohový regulátor

  U dvoupolohové regulace dosáhneme nejmenšího výstupního zvlnění, pokud je rozkmit X_k co možná nejmenší. Vyjdeme-li ze vzorce (7), tak výrazného zmenšení dosáhneme snížením pásma hystereze, které můžeme zvolit nulové (26). Další parametry jsou závislé na vlastnostech měniče, které bychom změnili pouze úpravou parametrů (Tabulka 1).

$$w_h - w_d = h = 0 \tag{26}$$

$$w_h = w_d = w \tag{27}$$

Regulátor pracuje na principu „porovnej a nastav", což znamená, že každý vzorek signálu je porovnán, při nulové hysterezi, se žádanou hodnotou. Pokud je žádaná hodnota větší než navzorkovaný signál musí dojít k připojení zdroje napětí, pokud je však žádaná hodnota menší než navzorkovaný signál dojde k odpojení zdroje napětí.

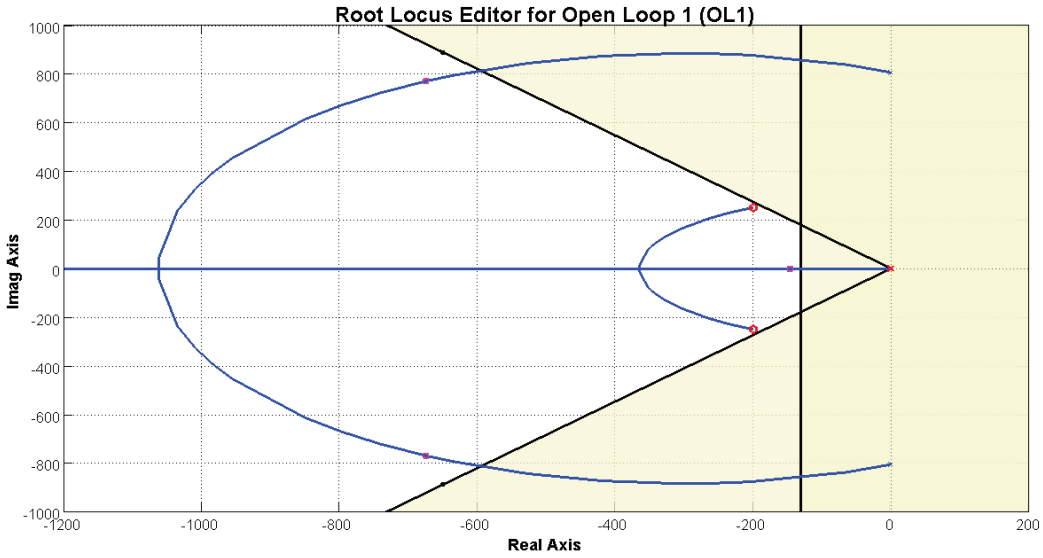
- Číslicový PID regulátor s převodem na PWM - PSD regulátor

  Číslicovým PID regulátorem rozumíme PSD regulátor, který navrhneme jako spojitý PID. Konstanty PID regulátoru pak přepočítáme na PSD regulátor, jehož akční veličinu upravíme pomocí PWM generátoru na impulsní regulátor. U spojitého regulátoru upravíme přenos soustavy měniče (23) za pomoci střední hodnoty napětí na tranzistoru (24), získáme tedy nový spojitý přenos (29).

$$G_1(s) = \frac{U_{out}(s)}{U_Q(s)} = \frac{U_{out}(s)}{(1-k)\,U_{in}} \tag{28}$$

$$G_1(s) = \frac{U_{out}(s)}{(1-k)} = \frac{U_{in}}{LCs^2 + \frac{L}{R}s + 1} \tag{29}$$

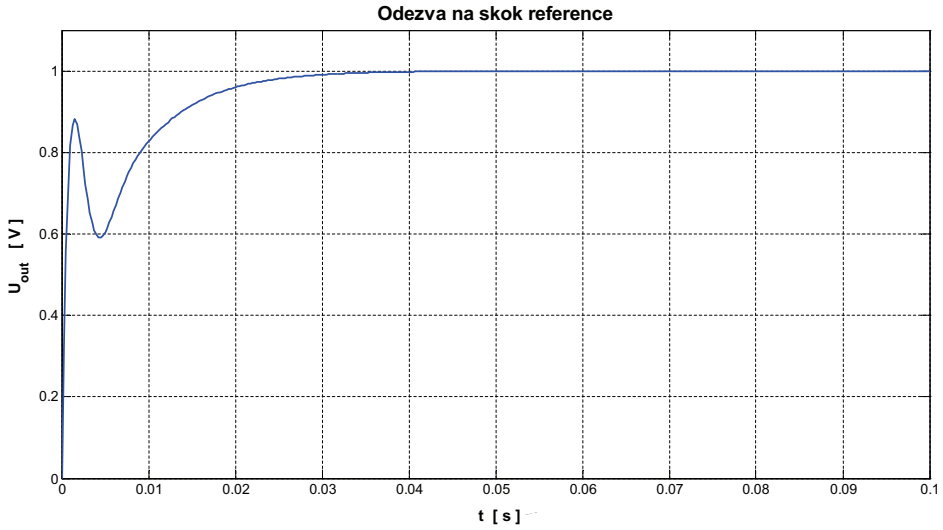
Přenos (29) je tedy výstupní napětí podělené upravenou střídou signálu, pro tento přenos navrhneme PID regulátor. PID regulátor navrhneme metodou GMK v Matlabu nástrojem Sisotool. Požadavek na PID regulátor je maximálně 10% překmit s dobou ustálení do 30ms s odchylkou ±2%.



Obrázek 15: Umístění pólů pomocí GMK.

$$R(s) = k_P + \frac{k_I}{s} + k_D s = 0,08 + \frac{20}{s} + 0,0001922s \tag{30}$$

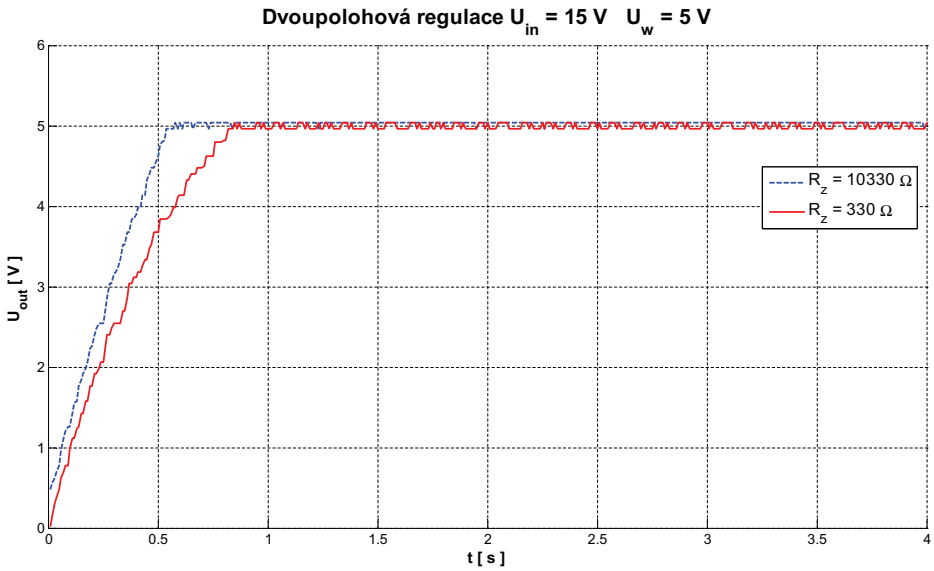
$$\begin{aligned} u(k) &= 0,08\left\{e(k) - e(k-1) + 0,0125e(k) + 24,025\left[e(k) - 2e(k-1) + e(k-2)\right]\right\} \\ &\quad + u(k-1) \end{aligned} \tag{31}$$



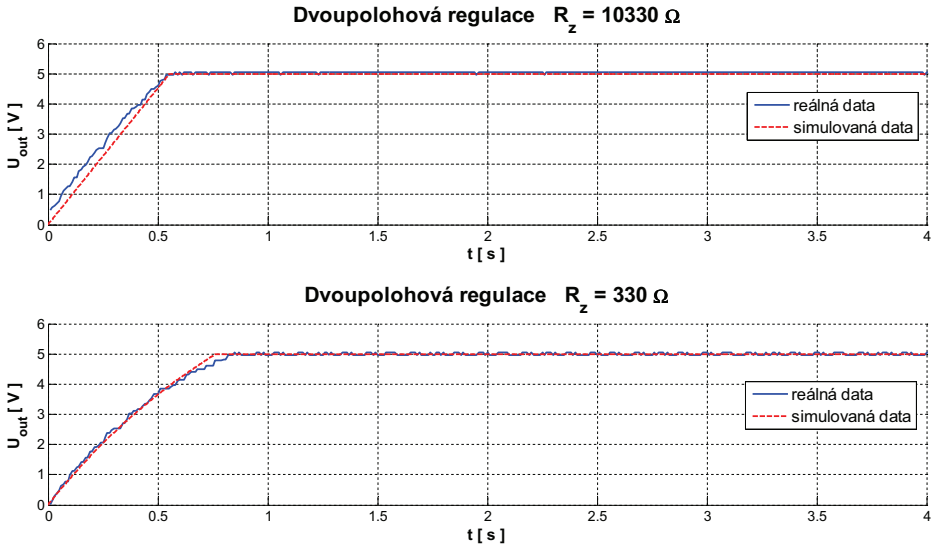
Obrázek 16: Odezva na skok reference.

4.5 Ověření funkčnosti regulátorů

a) Dvoupolohový regulátor

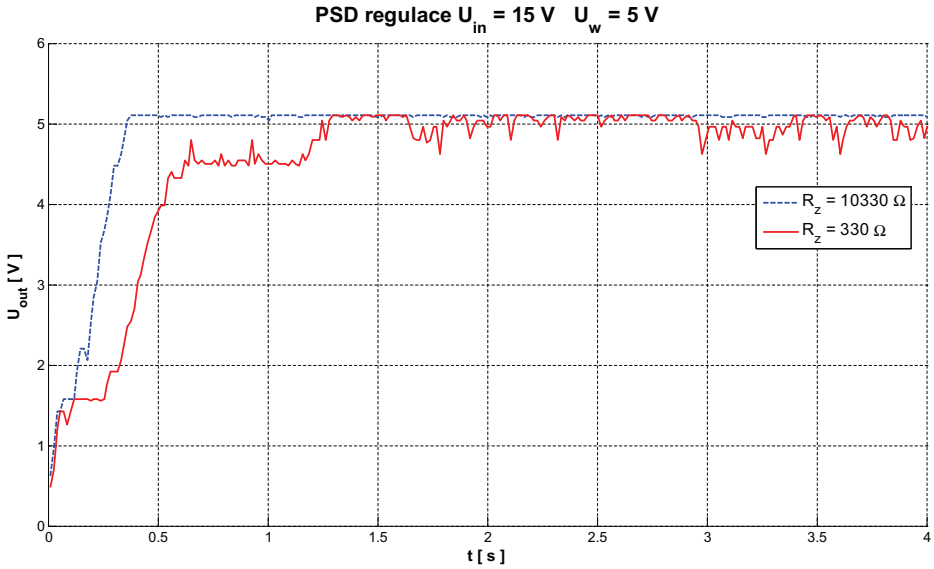


Obrázek 17: Porovnání dvoupolohového regulátoru pro konstantní vstupní napětí při různé zátěži.

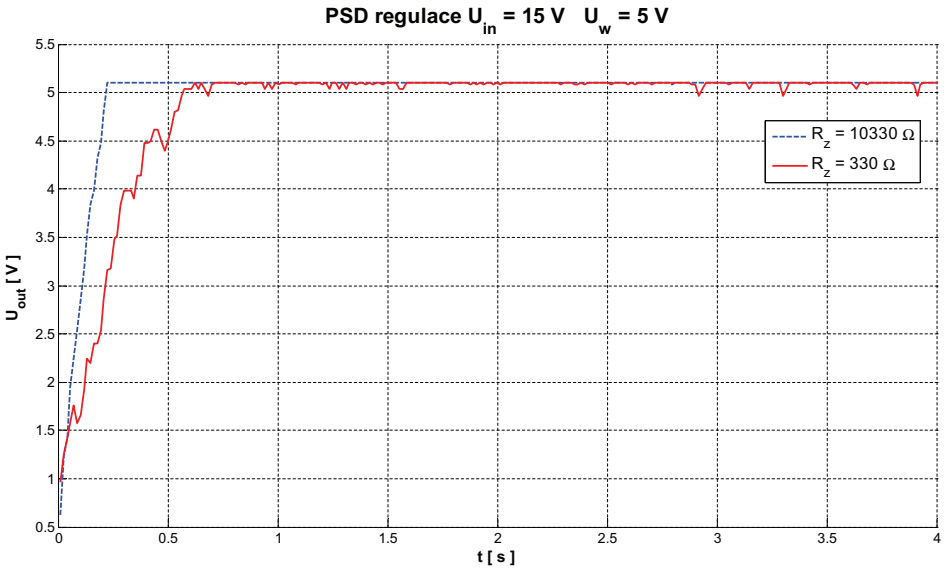


Obrázek 18: Porovnání simulovaných a reálných dat.

b) **Číslicový PID regulátor s převodem na PWM - PSD regulátor**



Obrázek 19: Porovnání PSD regulátoru pro konstantní vstupní napětí při různé zátěži $k_P = 0,08$, $k_I = 20$, $k_D = 0,0001922$.



Obrázek 20: Porovnání PSD regulátoru pro konstantní vstupní napětí při různé zátěži $k_P = 0,8$, $k_I = 10$, $k_D = 0,002$.

4.6 Vyhodnocení regulace

Při nižší zátěži (Obrázek 17) dochází k rychlejšímu vybíjení kondenzátoru C, což má za následek rychlejší pokles výstupního napětí, které se regulátor snaží udržet na hodnotě 5V s nulovým pásmem hystereze. Naopak u vyšší zátěže je setrvačnost soustavy větší, a proto nedochází k tak častému spínání tranzistoru. Zároveň u vyšší zátěže dochází k rychlejšímu akumulování energie a tím k rychlejší době náběhu. Při porovnání simulovaného průběhu a reálného průběhu (Obrázek 18) dochází jen k malé odchylce těchto dvou průběhů.

U PSD regulátoru si všimněme, že kvalita regulace závisí na zvolených konstantách regulátoru. U vypočtených konstant (Obrázek 19) je regulace lepší pro vyšší odpor, pro nižší hodnotu odporu dochází k značnému kolísání napětí. U ručně zvolených konstant je výstupní napětí stabilní pro obě hodnoty odporů (Obrázek 20).

Porovnáme-li dvoupolohovou a PSD regulaci zjistíme, že dvoupolohová regulace je přesnější, rychlost regulace je závislá u obou typů regulátoru na vlastnostech soustavy. Hlavní výhodou dvoupolohového regulátoru je nenáročnost návrhu a cena regulátoru, ovšem regulace je závislá především na vlastnostech soustavy. U PSD regulátoru je správná funkčnost regulátoru je zajištěna vhodnou volbou konstant regulátoru.

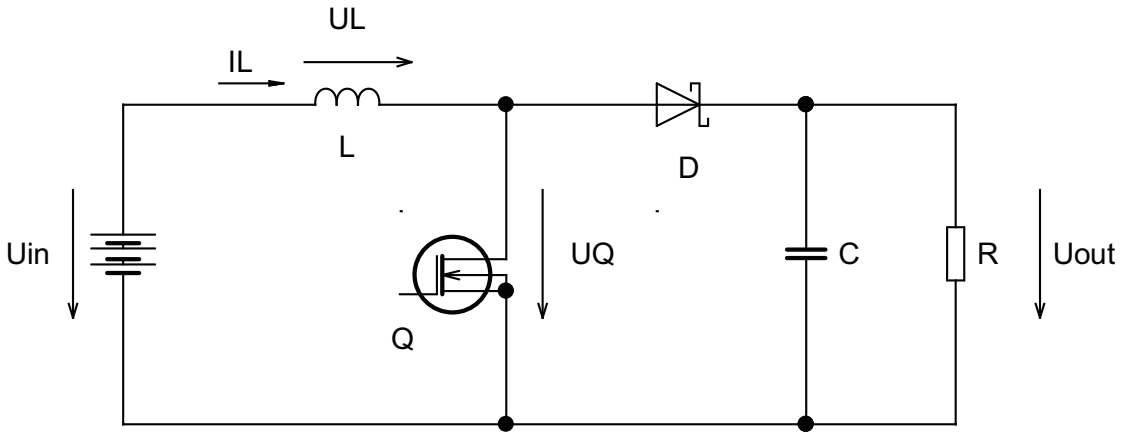
5 Zvyšující měnič

5.1 Teoretický úvod

Zvyšující měnič je označovaný jako step up nebo měnič typu boost. Měnič pracuje opět ve dvou režimech. Režim, kdy je spínací prvek je sepnut, a režim, kdy je spínací prvek je rozepnut. Spínacím prvkem je tranzistor Q.

Při sepnutém spínacím prvku se cívka L chová jako spotřebič a dioda D je polarizována v závěrném směru. Pokud je tedy tranzistor sepnut, je proud do zátěže dodáván kondenzátorem C, který se vybíjí a výstupní napětí klesá.

S rozepnutím spínače Q přejde cívka L do stavu, kdy se chová jako zdroj, zachovává směr proudu, ale obrací polaritu napětí. Napětí na tranzistoru je součtem napětí zdroje U_{in} a napětím cívky U_L. Dioda D je polarizována v propustném směru. Kondenzátor C je dobíjen proudem I_C a roste na něm napětí, tudíž roste i výstupní napětí U_{out}[5].



Obrázek 21: Zvyšující měnič.

5.2 Stavový popis

Při odvození stavového popisu vyjdeme ze základního zapojení zvyšujícího měniče (Obrázek 21). Úbytek napětí na diodě, kterou budeme považovat za ideální, zanedbáme. Stavové rovnice určíme pomocí diferenciálních rovnic, které popisují zvyšující měnič. Jako stavový popis zvolíme proud protékající cívkou a druhou veličinou výstupní napětí (napětí na kondenzátoru $u_C = u_{out}$). Vstup měniče nahradíme pomocí úbytku napětí na spínacím prvku Q, v ideálním případě dosahuje napětí na tranzistoru buď nulové hodnoty nebo součtu napětí zdroje a napětí na cívce $U_{in} + U_L$. Pro sestavení přenosů můžeme použít stavového popisu, jako stavový vektor zvolíme $[\ i_L \quad u_C\ ]^{\mathrm{T}}$.

1. **Spínací prvek Q sepnut**

$$u_Q = 0 \tag{32}$$

$$i_L = \frac{1}{L}\int (u_{in} - u_Q)\, dt + i_0 \tag{33}$$

$$\dot{i}_L = \frac{1}{L}(u_{in} - u_Q) \tag{34}$$

$$\frac{u_C}{R} + C\frac{du_C}{dt} = 0 \tag{35}$$

$$\dot{u}_C = -\frac{u_C}{RC} \tag{36}$$

2. **Spínací prvek Q rozepnut**

$$u_Q = u_C \tag{37}$$

$$i_L = \frac{1}{L}\int (u_{in} - u_Q)\, dt + i_0 \tag{38}$$

$$\dot{i}_L = \frac{1}{L}(u_{in} - u_Q) \tag{39}$$

$$-i_L + \frac{u_C}{R} + C\frac{du_C}{dt} = 0 \tag{40}$$

$$\dot{u}_C = \frac{i_L}{C} - \frac{u_C}{RC} \tag{41}$$

- Stavový popis - Stavový popis získáme porovnáním rovnic (34), (36), (39) a (41). Chybějící členy, které mizejí v závislosti na režimu, ve kterém se právě měnič nachází, vynásobíme buď 1 nebo 0, čímž si do stavového popisu zavedeme akční zásah a získáme popis (42).

$$\begin{gathered}\begin{bmatrix} \dot{i}_L \\ \dot{u}_C \end{bmatrix} = \begin{bmatrix} \frac{u_{in}}{L} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{RC} \end{bmatrix} \cdot \begin{bmatrix} i_L \\ u_C \end{bmatrix} + \begin{bmatrix} -\frac{u_C}{L} \\ \frac{i_L}{C} \end{bmatrix} \cdot \begin{bmatrix} u \end{bmatrix} \\ [u_{out}] = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i_L \\ u_C \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot [u]\end{gathered} \tag{42}$$

, kde u představuje akční zásah regulátoru $u\epsilon\{0;1\}$, je-li $u = 0$ jedná se o stav 1. Spínací prvek sepnut , je-li $u = 1$ jedná se o stav 1. Spínací prvek rozepnut.

Zatímco u snižujícího měniče byl stavový popis lineární a nebylo jej potřeba linearizovat, tak u zvyšujícího měniče je stavový popis nelineární a musíme linearizovat v pracovním bodě. Linearizací získáme stavový popis (43), ze kterého sestavíme přenos soustavy pro určitý pracovní bod.

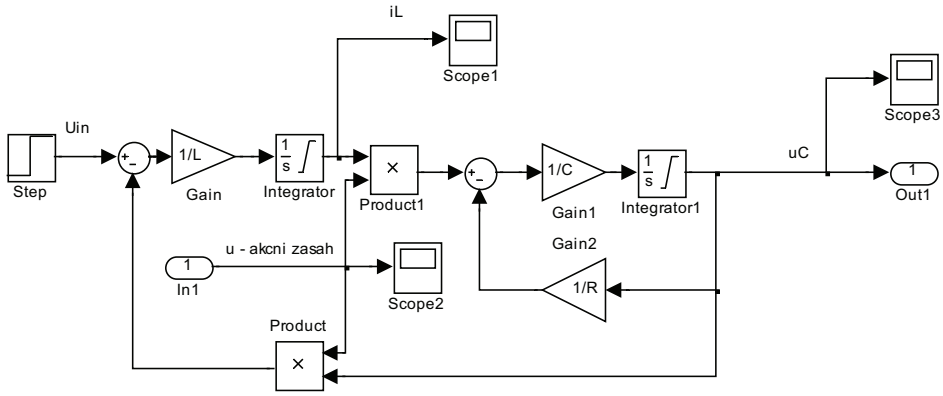
- Linearizovaný stavový popis

$$\begin{gathered}\begin{bmatrix} \Delta\dot{i}_L \\ \Delta\dot{u}_{out} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{u_0}{L} \\ \frac{u_0}{C} & -\frac{1}{RC} \end{bmatrix} \cdot \begin{bmatrix} \Delta i_L \\ \Delta u_C \end{bmatrix} + \begin{bmatrix} -\frac{u_{C0}}{L} \\ \frac{i_{L0}}{C} \end{bmatrix} \cdot \begin{bmatrix} \Delta u \end{bmatrix} \\ [\Delta u_{out}] = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta i_L \\ \Delta u_C \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot [\Delta u]\end{gathered} \tag{43}$$

- Přenos linearizované soustavy

$$G(s) = \frac{\Delta U_{out}(s)}{\Delta U(s)} = \frac{LR \cdot i_{L0} \cdot s - R \cdot u_0 \cdot u_{C0}}{CLR \cdot s^2 + L \cdot s + R \cdot u_0^2} \tag{44}$$

- Model soustavy



Obrázek 22: Simulinkové schéma zvyšujícího měniče.

5.3 Řízení měniče

Řízení měniče opět neprobíhá spojitě, u tohoto měniče by bylo využití dvoupolohového regulátoru obtížnější než u snižujícího měniče, protože při dvoupolohové regulaci by mohlo docházet k chybovým stavům a činnost měniče by nemusela být správná. Zatímco u snižujícího měniče jsme porovnávali jen výstupní napětí, u zvyšujícího měniče bychom museli kontrolovat i napětí na cívce, což by vedlo ke složitějším podmínkám. Z tohoto důvodu pro řízení měniče využijeme pouze impulsový regulátor. Navrhneme opět spojitý regulátor, který následně diskretizujeme na PSD regulátor a převedeme na PWM regulaci.

5.4 Návrh měniče

a) **Měnič**

Při návrhu zvyšujícího měniče bychom postupovali obdobně jako u snižujícího měniče. Protože nám však jde o princip řízení, které je závislé na vzorkování, musíme jej vzít v úvahu při návrhu. Zvolíme-li parametry měniče (Tabulka 2), stejné jako u snižujícího měniče, vyjde nám vzorkovací frekvence $3\,kHz$.

Minimální vzorkovací frekvenci vypočítáme z linearizovaného přenosu (46), do kterého dosadíme hodnoty (Tabulka 2). Počáteční podmínky zjistíme pomocí simulace, pokud za u_0 dosadíme střídu signálu k, protože výstupní napětí je závislé na vstupním napětí a střídě řídicího signálu (45).

$$U_{out} = \frac{U_{in}}{1-k} \tag{45}$$

$$u_0 = k = \frac{U_{out} - U_{in}}{U_{out}} = \frac{10 - 5}{10} = 0,5$$

$$u_{C0} = 10\,V$$

$$i_{L0} = 0,02\,A$$

$$G\left(s\right) = \frac{9,4 \cdot 10^{-3} \cdot s - 5000}{1,551 \cdot 10^{-3} \cdot s^2 + 470 \cdot 10^{-6} \cdot s + 250} \tag{46}$$

$$\omega_{BW} \approx 624\,rad \cdot s^{-1}$$

$$f_{vz} \approx 3\,kHz$$

Tabulka 2: Parametry měniče

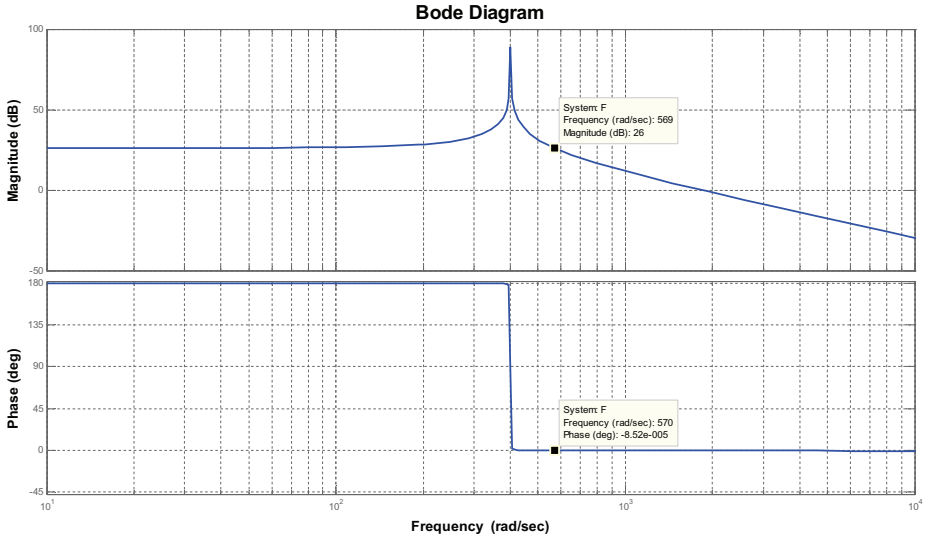
Veličina	Hodnota
L	$470\mu H$
C	$3300\mu F$
R_Z	$(0,33 \div 10,33)\,k\Omega$
U_{in}	$5V$
U_{out}	$10V$
f_{vz}	$10kHz$

b) **Regulátor**

Regulátor opět navrhujeme s určitými požadavky na zvlnění a rychlost regulace.

- Dvoupolohový regulátor
- Číslicový PID regulátor s převodem na PWM - PSD regulátor

Impulsní PSD regulátor navrhneme podobně jako u snižujícího měniče, k nastavení a určení konstant využijeme linearizovaný přenos (44).



Obrázek 23: Amplitudová a fázová charakteristika v logaritmických souřadnicích.

$$k_D = \frac{1}{\omega_D \sqrt{2} |P(j\omega)|} = \frac{1}{570\sqrt{2} \cdot 19,72} = 6,3 \cdot 10^{-5} \tag{47}$$

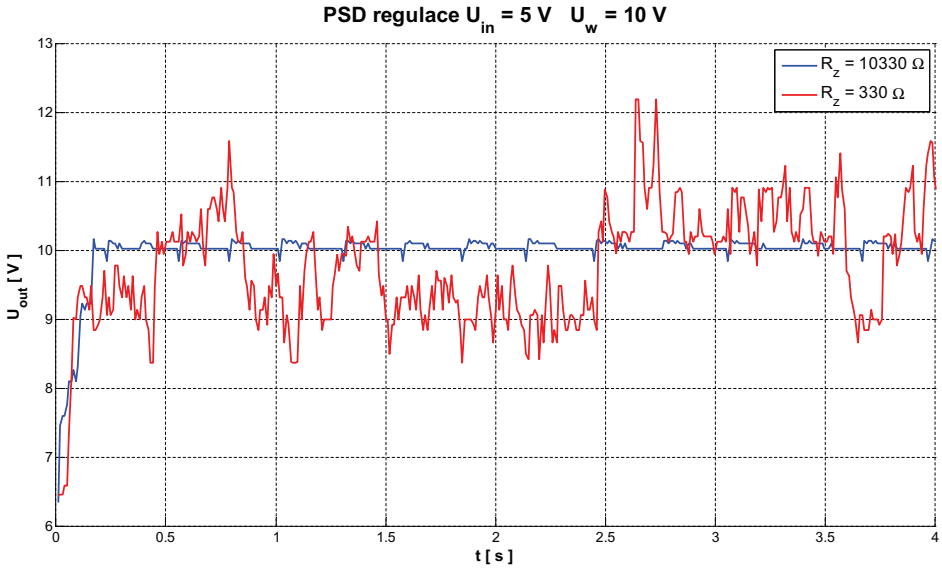
$$k_P = 1,1 \omega_D k_D = 1,1 \cdot 570 \cdot 6,29 \cdot 10^{-5} = 3,9 \cdot 10^{-3} \tag{48}$$

$$k_I = 0,1 k_D \omega_D^2 = 0,1 \cdot 6,29 \cdot 10^{-5} \cdot 570^2 = 0,20 \tag{49}$$

$$R(s) = k_P + \frac{k_I}{s} + k_D s = 0,0039 + \frac{0,20}{s} + 0,000063 s \tag{50}$$

$$\begin{aligned} u(k) &= 0,0039 \{e(k) - e(k-1) + 0,00000098 e(k) + 160 [e(k) - 2 \cdot e(k-1) + e(k-2)]\} \\ &\quad + u(k-1) \end{aligned} \tag{51}$$

5.5 Ověření funkčnosti regulátorů



Obrázek 24: Porovnání PSD regulátoru pro konstantní vstupní napětí při různé zátěži $k_P = 0,004$, $k_I = 0,2$, $k_D = 0,000063$.

5.6 Vyhodnocení regulace

PSD regulátor regulátor pro zvyšující měnič, který jsme navrhli pomocí frekvenční metody pro zvolenou fázovou bezpečnost, reguluje lépe pro vyšší hodnotu zátěže (Obrázek 24). Měnič jsme museli zapojit na laboratorní zdroj napětí, protože ze stabilizovaného zdroje (15V/1,6A), jsme nebyli schopni pokrýt proudový odběr měniče. Výkyvy napětí u nižší hodnoty zátěže jsou způsobeny převážně kolísáním vstupního napětí, které opět nedokázalo pokrýt proudový odběr.

6 Závěr

U obou typů měničů, jsme museli upustit od nastavitelného vstupního zdroje napětí, které bylo nastavováno pomocí PWM výstupu mikroprocesoru a dále převedeno pomocí aktivního filtru typu dolní propust na na vstupní napětí měniče. Aktivní filtr nebyl schopen udržet stabilní napětí, což bylo způsobeno vyšším proudovým odběrem samotného měniče. Tento proudový odběr nedokázal operační zesilovač pokrýt, což mělo za následek pokles výstupního napětí operačního zesilovače (vstupního napětí měniče U_{in}). U snižujícího měniče jsme tento problém vyřešili tak, že jako vstupní napětí měniče bylo použito napájení přípravku ze stabilizovaného síťového napáječe (15V/1,6A). U zvyšujícího měniče jsme museli použít externí laboratorní zdroj, který by pokryl proudový odběr měniče. Problém se vstupním napětím měniče při zachování programově nastavitelného zdroje napětí bychom museli řešit pomocí dalších prvků, které by zachovali velikost napětí a zároveň by dokázali dodávat do měniče větší proud.

U snižujícího měniče je, z hlediska složitosti návrhu, mnohem výhodnější použití jednoduchého dvoupolohového regulátoru. Překmity výstupního napětí od žádané hodnoty jsou u dvoupolohového regulátoru závislé především na vlastnostech soustavy, z naměřených hodnot se zvlnění výstupního napětí pohybuje přibližně v pásmu $\pm 1\%$ od žádané hodnoty. PSD regulátor pro snižující měnič je výstupní napětí silně závislé na nastavení konstant regulátoru. Pro kvalitnější regulaci by bylo vhodnější použít spojitého PID regulátoru s převodem na PWM. Ovšem u spojitých regulátorů je nutno stabilizovat referenční napětí, které slouží jako žádaná hodnota regulátoru a potřebují další napájení pro operační zesilovače jimiž jsou realizovány jednotlivé konstanty regulátoru.

U zvyšujícího měniče se značně projevila stabilita vstupního napětí, které mělo za následek kolísání výstupního napětí měniče převážně u nižší hodnoty zátěže. U vyšší hodnoty zátěže se kolísání vstupního napětí tolik neprojevilo a regulace proběhla pouze s mírným kolísání výstupního napětí. Opět pro kvalitnější regulaci by bylo výhodnější použít spojitého regulátoru.

7 Literatura a zdroje

[1] M. Hlinovský, J. Honců, P. Němeček, O. Vysoký. Elektronické systémy - Návody ke cvičením. Česká technika - nakladatelství ČVUT, 2006

[2] Optimalizace zatěžování solárních měničů energie v reálných provozních podmínkách, ÚEEN FEKT VUT v Brně, 2006
Dostupné na: <http://www.ueen.feec.vutbr.cz/laboratory-of-unconventional-energy-conversion/mereni_info.php>

[3] M. Rampas. Automatizace pro 4. ročník SPŠ

[4] Technická dokumentace ATmega8, 2011
Dostupné na: <http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf>

[5] A. Krejčeřík. DC/DC měniče. Technická literatura BEN, Praha, 2001

[6] M. Gálus. Bakalářská práce - Preregulátory pro spínané zdroje. VUT, 2009

[7] J. Dvořáček. Bakalářská práce - Laserový projektor. ČVUT, 2008

[8] M. Melzer. Bakalářská práce - Fuzzy regulátory při řízení systémů se suchým třením. ČVUT, 2009

[9] Y. Cao. Configurable Simulink Model for DC-DC Converters with PWM PI Control, 2009
Dostupné na: <http://www.mathworks.com/matlabcentral/fileexchange/18833-configurable-simulink-model-for-dc-dc-converters-with-pwm-pi-control?s_v1=18313971_1-5JXGN>

[10] Knihovna pro komunikaci s převodníkem FT232RL
Dostupné na: <http://rxtx.qbang.org/wiki/index.php/Main_Page>

[11] M. Ossman. Kleine DC/DC-Konverter, 2002, 64-68
Dostupné na: <www.souch.cz/dok/e/menice.doc>

[12] D. Novák. Snižující měnič MP2358, 2006
Dostupné na: <http://hw.cz/Novinky/Ostatni/ART1584-Snizujici-menic-MP2358.html>

[13] Technická dokumentace FT232RL, 2010
Dostupné na: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[14] Software pro programování ATmega8, AVR Studio4
Dostupné na: <http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725>

[15] Software pro programování počítačových aplikací, Netbeans
Dostupné na: <http://netbeans.org/>

[**16**] Software pro tvorbu plošných spojů, EAGLE 5.11.0
Dostupné na: <http://www.cadsoft.de>

[**17**] Software pro návrh regulátorů, Matlab
Dostupné na: <http://www.humusoft.cz/produkty/matlab/>

[**18**] J. Babčaník. Spínané zdroje, 2007
Dostupné na: <http://hw.cz/Teorie-a-praxe/ART1876-Spinane-zdroje.html>

[**19**] Programování jednočipů ATmega8
Dostupné na: <http://cs.wikibooks.org/wiki/Programujeme_jedno%C4%8Dipy#Program_po.C4.8D.C3.ADtaj.C3.ADc.C3.AD_impulzy>

[**20**] V. Váňa. Mikrokontroléry ATMEL AVR. Technická literatura BEN, Praha, 2003

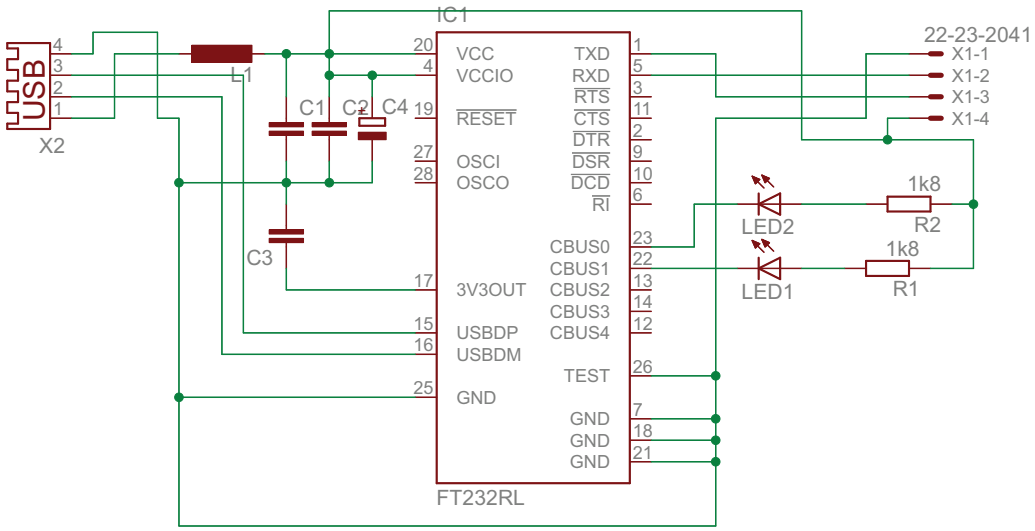
8 Dodatky

8.1 Schémata

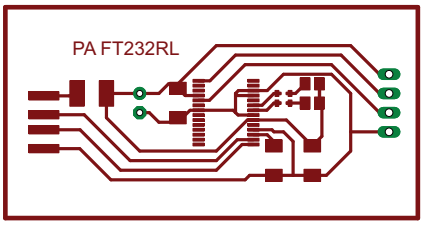
a) Převodník USB/UART

Tabulka 3: Výpis použitých součástek

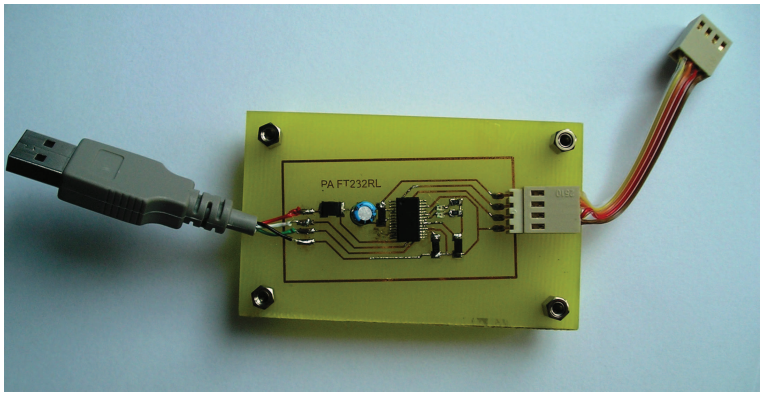
Označení	Hodnota	Popis
$FT232RL$	$470\mu H$	převodník USB/UART
$C1$, $C3$	$10nF$	
$C2$	$10\mu F$	
$C4$	$100\mu F$	
$L1$	$10\mu H$	
$R1$, $R2$	$1,8k\Omega$	ochranný odpor
$LED1$	*zelená*	indikační dioda pro příjem
$LED2$	*červená*	indikační dioda pro odesílání
$X1$		čtyř-pinový konektor se zámkem
$X2$		usb konektor



Obrázek 25: Zapojení převodníku USB/UART.



(a) Plošný spoj.



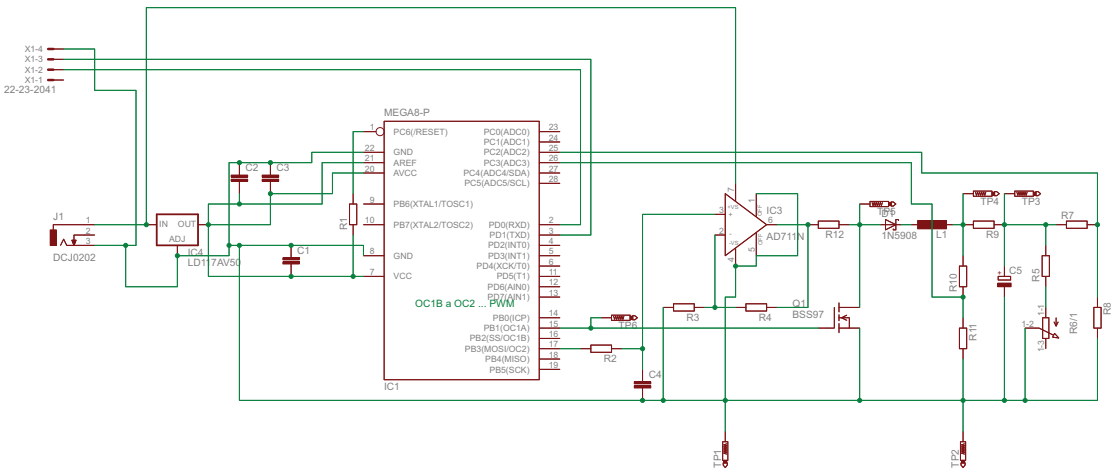
(b) Hotový výrobek.

Obrázek 26: Převodník USB/UART.

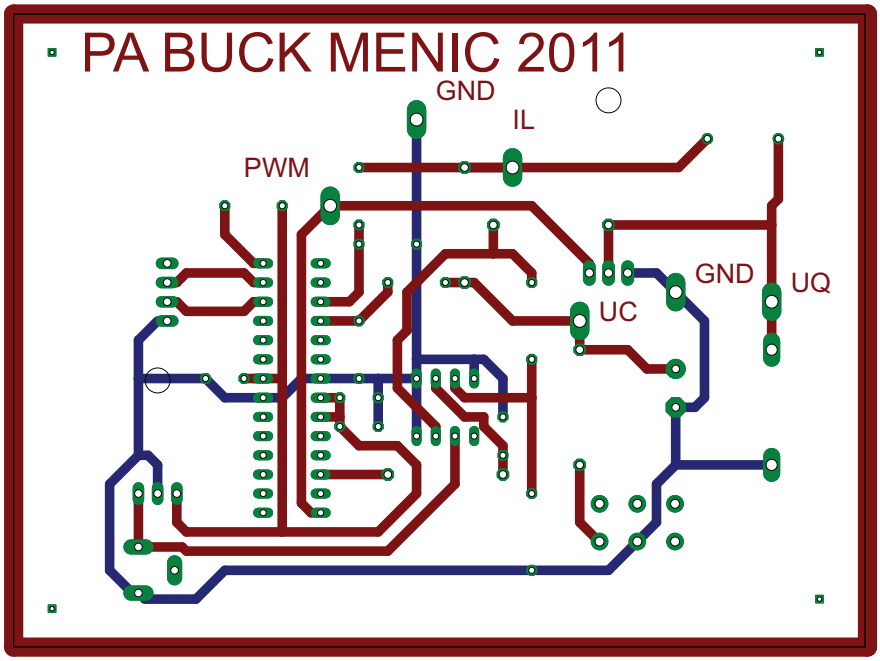
b) Snižující měnič

Tabulka 4: Výpis použitých součástek

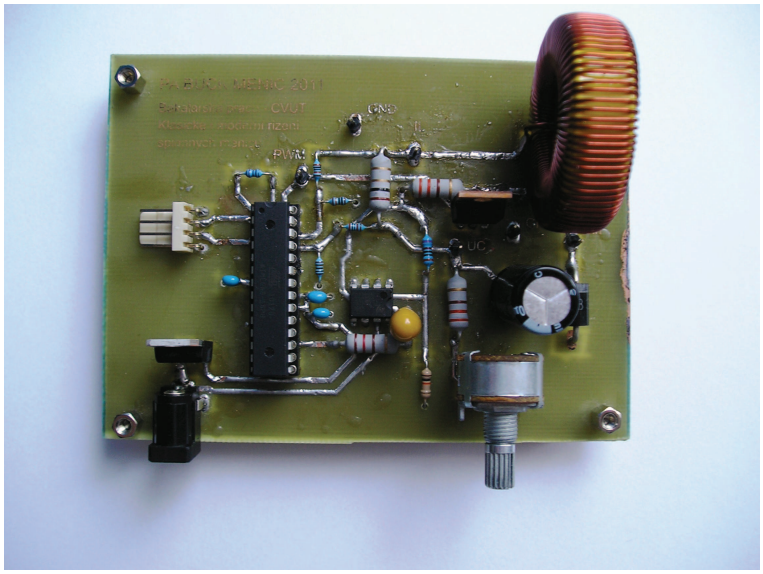
Označení	Hodnota	Popis
MEGA8-P		mikroprocesor ATmega8
$C1$, $C2$, $C3$	$100nF$	
$C4$	$100\mu F$	DP filtr
$C5$	$3300\mu F$	elektrolytický kondenzátor
$IC2$	7805	stabilizátor napětí
$R1$, $R3$	$10k\Omega$	
$R2$	150Ω	DP filtr
$R4$	$30k\Omega$	
$R5$, $R12$	330Ω	předřadné odpory
$R6$	$(0 \div 10)\, k\Omega$	indikační dioda pro příjem
$R7$, $R10$	$300k\Omega$	napěťový dělič
$R8$, $R11$	$100k\Omega$	napěťový dělič
$R9$	1Ω	měření proudu I_L
$L1$	$470\mu H$	toroidní cívka
$D1$		Schottkyho dioda
$Q1$		tranzistor typu MOSFET
$IC3$		operační zesilovač
$J1$		konektor pro síťový napáječ
$X1$		čtyř pinový konektor se zámkem
$TP1 \div TP6$		měřící piny



Obrázek 27: Zapojení snižujícího.



(a) Plošný spoj.



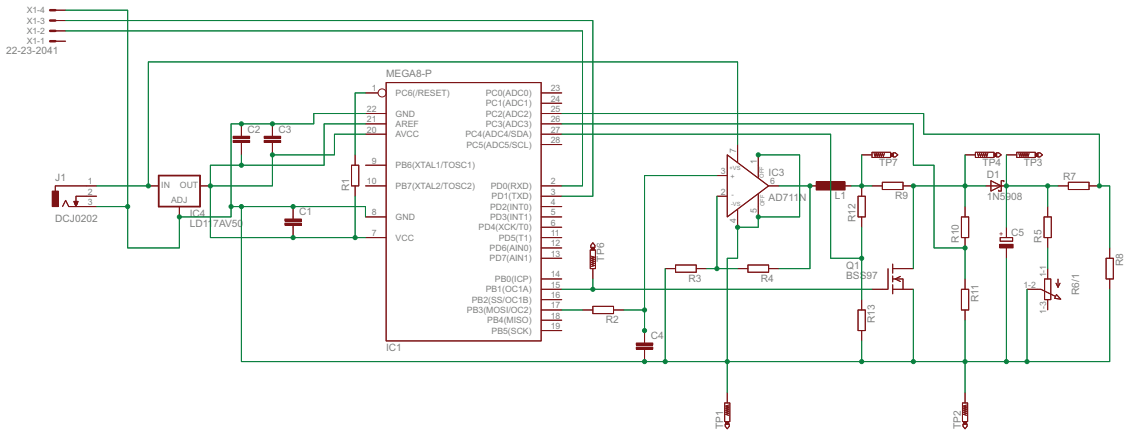
(b) Hotový výrobek.

Obrázek 28: Snižující měnič.

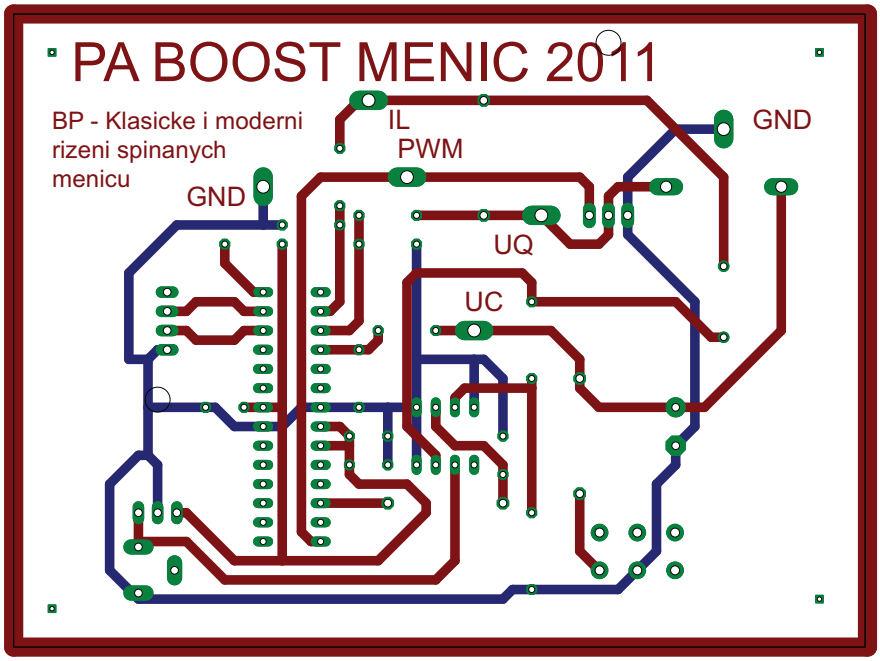
c) Zvyšující měnič

Tabulka 4: Výpis použitých součástek

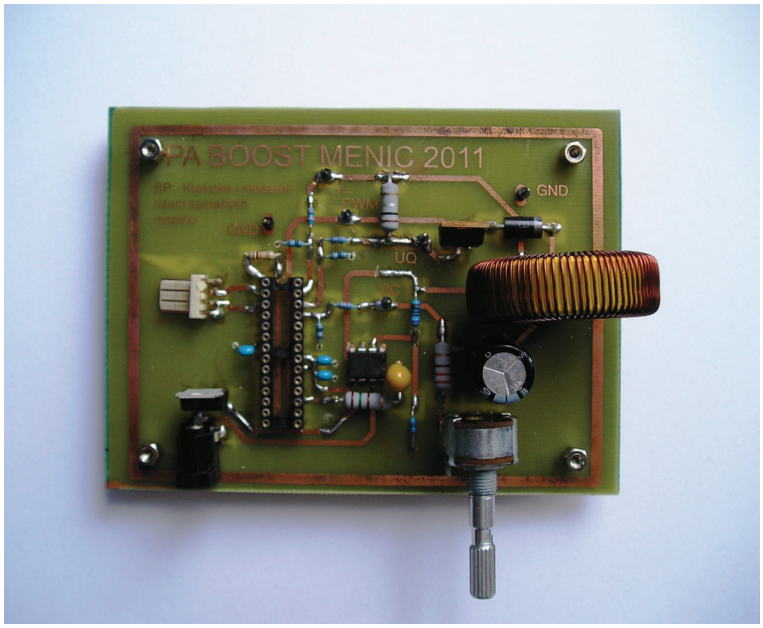
Označení	Hodnota	Popis
MEGA8-P		mikroprocesor ATmega8
$C1$, $C2$, $C3$	$100nF$	
$C4$	$100\mu F$	DP filtr
$C5$	$3300\mu F$	elektrolytický kondenzátor
$IC2$	7805	stabilizátor napětí
$R1$, $R3$	$10k\Omega$	
$R2$	150Ω	DP filtr
$R4$	$30k\Omega$	
$R5$	330Ω	předřadné odpory
$R6$	$(0 \div 10)\, k\Omega$	indikační dioda pro příjem
$R7$, $R10$, $R12$	$300k\Omega$	napěťový dělič
$R8$, $R11$, $R13$	$100k\Omega$	napěťový dělič
$R9$	$0,047\Omega$	měření proudu I_L
$L1$	$470\mu H$	toroidní cívka
$D1$		Schottkyho dioda
$Q1$		tranzistor typu MOSFET
$IC3$		operační zesilovač
$J1$		konektor pro síťový napáječ
$X1$		čtyř pinový konektor se zámkem
$TP1 \div TP7$		měřící piny



Obrázek 29: Zapojení zvyšujícího měniče.



(a) Plošný spoj.



(b) Hotový výrobek.

Obrázek 30: Zvyšující měnič.

8.2 Programy

1. Mikroprocesor

```
////////////////////////////////////////////////////////
//////////////////////KNIHOVNY//////////////////////////
////////////////////////////////////////////////////////

#include <avr/io.h>
#include <avr/pgmspace.h>
#include <stdint.h>
#include <avr/interrupt.h>
#include <stdio.h>


////////////////////////////////////////////////////////
//////////////////////KONSTANTY/////////////////////////
////////////////////////////////////////////////////////
#define MOSFET 1
#define UIN 3
#define PROUD 3
#define NAPETI 2

////////////////////////////////////////////////////////
///////////////DEKLARACE FUNKCI A METOD/////////////////
////////////////////////////////////////////////////////

void initUinAPin (void);
void initPWMregulace(void);
void initADC (void);
void initUART (void);
void initRizeni (void);
void initTimer (void);
void ADCstart (int channel);
void RegulacePID (int kp, int Ti, int Td, int Vref, int Vw);
void RegulaceDvouPolohova (int Vref, int wh, int wd);
unsigned char uartRX (void);
void uartTX (unsigned char data);
int prevodCharNaInt (unsigned char znak);


////////////////////////////////////////////////////////
//////////////////////HLAVNÍ PROGRAM////////////////////
////////////////////////////////////////////////////////

int main(void) {

        initUinAPin();
        initADC();
        initUART();

        int i = -1;
        int j = 0;
        int k = 0;
        unsigned char znak[30], pismeno;
        int kp = 0, ki =0, kd = 0, w = 0, wh = 0, wd = 0, u = 0;



        for(;;){
                i = -1; j = 0; k = 0, pismeno = ' ';

//Prijem znaku dokud neni prenos dokoncen
                while(pismeno != 'K'){
                        i++;
                        znak[i] = uartRX();
                        pismeno = znak[i];

                }

                kp = 0; ki =0; kd = 0; w = 0;
                wh = 0; wd = 0; u = 0;

//Parsovani prijate zpravy
                for (j = 0; j <= i; j++){
                        uartTX(znak[j]);
                        if(znak[0] == 'H'){
                                switch (k){
                                        case 0:
                                                if (znak[j] == 'S'){
                                                        k++;
                                                }
                                                else{
                                                        wh = wh*10
                                                        + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                        case 1:
                                                if (znak[j] == 'U'){
                                                        k++;
                                                }
                                                else{
                                                        wd = wd*10
                                                        + prevodCharNaInt(znak[j]);
```

```
                                                }
                                                break;
                                        case 2:
                                                if (znak[j] == 'K'){
                                                        k++;
                                                }
                                                else{
                                                        u = u*10
                                                        + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                }
                        }
                        if(znak[0] == 'P'){
                                switch (k){
                                        case 0:
                                                if (znak[j] == 'I'){
                                                        k++;
                                                }
                                                else{
                                                        kp = kp*10
                                                        + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                        case 1:
                                                if (znak[j] == 'D'){
                                                        k++;
                                                }
                                                else{
                                                        ki = ki*10
                                                        + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                        case 2:
                                                if (znak[j] == 'W'){
                                                        k++;
                                                }
                                                else{
                                                        kd = kd*10
                                                        + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                        case 3:
                                                if (znak[j] == 'U'){
                                                        k++;
                                                }
                                                else{
                                                        w = w*10
                                                    + prevodCharNaInt(znak[j]);
                                                }
                                                break;
                                        case 4:
                                                if (znak[j] == 'K'){
                                                        k++;
                                                }
                                                else{
                                                        u = u*10
                                                        + prevodCharNaInt(znak[i]);
                                                }
                                                break;
                                }
                        }
                }

                //Potrvrzeni spravnosti
                i=-1;
                pismeno = ' ';
                while(pismeno != 'K'){
                        i++;
                        znak[i] = uartRX();
                        pismeno = znak[i];

                }


                if((znak[1] == 'H' || znak[1] == 'P')&& znak[2] == 'K'){
                        //Vyber typu regulace
                        if(znak[1] == 'P')
                        RegulacePID(kp, ki, kd, 5, w);
                        if(znak[1] == 'H')
                                RegulaceDvouPolohova(5, wh, wd);
                }

        }
        return 0;
}

//Prevod znaku na cislo
int prevodCharNaInt (unsigned char znak){

        int cislo = 0;

        switch (znak){
                case '0': cislo = 0; break;
                case '1': cislo = 1; break;
```

```
                case '2': cislo = 2; break;
                case '3': cislo = 3; break;
                case '4': cislo = 4; break;
                case '5': cislo = 5; break;
                case '6': cislo = 6; break;
                case '7': cislo = 7; break;
                case '8': cislo = 8; break;
                case '9': cislo = 9; break;
        }

        return cislo;
}

//////////////////////////////////////////////////////////
//////////////////////INICIALIZACE//////////////////////
//////////////////////////////////////////////////////////

//inicializace Pinu a vstupniho napeti
void initUinAPin (void) {
        // Nastaveni pinu pro vystup
        DDRB|= (1 << MOSFET) | (1 << UIN);

        // Nastaveni PWM pro generovani vstupniho napeti
        TCCR2 = (1<<CS20)|(1<<WGM21)|(1<<WGM20)|(1<<COM21);
        OCR2 = 0;    // 55~1V~4V !!!OFFSET
        PORTB |= (1<<MOSFET);

}

//Inicializace PWM výstupu pro regulaci
void initPWMregulace(void){
        // INICIALIZACE PORTB I/O
        DDRB |= (1<<MOSFET) | (1<<UIN);      // Nastaveni pinu pro vystup PWM

        TCCR1A = (1<<COM1A1)|(1<<WGM11);
        TCCR1B = (1<<CS10)|(1<<WGM12)|(1<<WGM13);
        ICR1 = 0;          //TOP - ICR1=79
        OCR1A = 20;
        TCCR2 = (1<<CS20)|(1<<WGM21)|(1<<WGM20)|(1<<COM21);
        OCR2 = 124;    //max 252 100=>4V
}


//Inicializace A/D prevodníku
void initADC(void){
        DDRC |= (0<<PROUD)|(0<<NAPETI);

        ADMUX |= (0<<REFS1)|(1<<REFS0)|(0<<ADLAR);

        ADCSRA|=(1<<ADEN);//|(1<<ADIE);
}

//Inicializace PWM výstupu a výstupu pro dvoupolohovou regulaci
void initRizeni(void){
        OCR2 = 124;    //max 252  150~10V ... smernice = 1/15
}

//Inicializace 100us prodlevy
void initTimer(void){
        TCCR0 |= (0<<CS02)|(1<<CS01)|(0<<CS00);
}

//Inicialiyace seriove komunikace
void initUART(void)
{

    /***** inicializace uart  *****/
    UCSRA = 0x00;
    UBRRH = 0x00;  // nastaveni rychlosti pro krystal  1 MHz
    UBRRL = 25;    // (v pripade taktovani 8 MHz zde pouzijte  UBRRL = 0x33; )
    UCSRB = 0x18;  // povolit vysilani a prijem
    UCSRC = 0x86;  // ramec dat:  8 datovych, 1 stop bit, bez parity

}

//////////////////////////////////////////////////////////
//////////////////////START ADC PREVODU/////////////////
//////////////////////////////////////////////////////////

void ADCstart (int channel){
//nastaveni multiplexeru, jaky pin ma pouzit pro prevod
  if(channel==2){
    ADMUX |= (0<<MUX3)|(0<<MUX2)|(1<<MUX1)|(0<<MUX0); //ADC2 ...NAPETI
  }
  if(channel==3){
    ADMUX |= (0<<MUX3)|(0<<MUX2)|(1<<MUX1)|(1<<MUX0); //ADC3 ...PROUD
  }

//Spusteni prevodu
  ADCSRA |= (1<<ADSC);
}


//////////////////////////////////////////////////////////
//////////////////////REGULATORY//////////////////////////
```

```
////////////////////////////////////////////////////////

void RegulaceDvouPolohova(int Vref, int wh, int wd){

        int e0=0;
        unsigned int prevodU = 0, prevodI = 0;
        int odeslano = 0;
        int napeti = 0, proud = 0, akce = 1;
        int poslani = 0, pocet = 0, odeslanych = 0;
        unsigned char znak [16] = {' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};

        initRizeni();
        initTimer();

        PORTB|=(1<<MOSFET);

        TCNT0 = 0;

        while(odeslano != 10000){

//vzorkovaci frekvence 10kHz
                while(TCNT0 < 50){
                }

//Odesilani znaku, pokud je volna linka
                if(poslani == 1 && ( UCSRA & (1<<UDRE))){
                        uartTX(znak[odeslanych]);
                        odeslanych++;
                        if(odeslanych == 16){
                                poslani = 0;
                                odeslanych = 0;
                                odeslano++;
                        }
                }
                TCNT0 = 0;

                ADCstart(2);
                while(ADCSRA & (1<<ADSC));
                prevodU = ADCL;
                prevodU += (ADCH<<8);

                ADCstart(3);
                while(ADCSRA & (1<<ADSC));
                prevodI = ADCL;
                prevodI += (ADCH<<8);

                e0=prevodU*Vref*4; // 8.09V  *3.95

//pripojeni ci odpojeni vstupu
                if(e0 > wh && akce == 0){
                        PORTB|=(1<<MOSFET);
                        akce = 1;
                }
                if(e0 < wd && akce == 1){
                        PORTB&=~(1<<MOSFET);
                        akce = 0;
                }

                pocet++;
                napeti =  + e0;
                proud =  + prevodI*Vref*4 - e0;

//ulozeni dat pokud byla predchozi odeslana
                if(poslani==0){
                        poslani = 1;
                        znak[0]='U';
                        znak[1]=(char)(48+(int)(napeti/10000));
                znak[2]=(char)(48+(int)((napeti%10000)/1000));
                znak[3]=(char)(48+(int)((napeti%1000)/100));
                znak[4]=(char)(48+(int)((napeti%100)/10));
                znak[5]=(char)(48+(int)(napeti%10));
                        znak[6]='I';
                znak[7]=(char)(48+(int)(proud/1000));
                znak[8]=(char)(48+(int)((proud%1000)/100));
                znak[9]=(char)(48+(int)((proud%100)/10));
                znak[10]=(char)(48+(int)(proud%10));
                        znak[11]='S';
                znak[12]=(char)(48+(int)(pocet/100));
                znak[13]=(char)(48+(int)((pocet%100)/10));
                znak[14]=(char)(48+(int)(pocet%10));
                        znak[15]='K';
                        napeti = 0;
                        proud = 0;
                        pocet = 0;
                }

        }

}


//PID (PSD) regulator ... osetrit regulatory ktere nejsou PID
void RegulacePID(int kp, int ki, int kd, int Vref, int Vw){
```

```
        /*
         * uk...aktualni akcni zasah
         * e0...regulacni odchylka e(k)
         * e1...regulacni odchylka e(k-1)
         * e2...regulacni odchylka e(k-2)
        */
        int e0=0,e1=0,e2=0;
        int uk=0;

        unsigned int prevodU = 0, prevodI = 0;
        int napeti = 0, proud = 0;
        int odeslano = 0;
        int poslani = 0, pocet = 0, odeslanych = 0;
        unsigned char znak [16] = {' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};


        initTimer();
        initPWMregulace();

        //Ridici smycka
        while(odeslano != 10000){

//nastaveni vzorkovaci frekvence 10kHz
                while(TCNT0 < 50){
                }

//pokud je mozne odeslat znak, se odesle
                if(poslani == 1 && ( UCSRA & (1<<UDRE))){
                        uartTX(znak[odeslanych]);
                        odeslanych++;
                        if(odeslanych == 16){
                                poslani = 0;
                                odeslanych = 0;
                                odeslano++;
                        }
                }
                TCNT0 = 0;

                ICR1=(int)((uk+500)*0.079);
                e2=e1;
                e1=e0;

                ADCstart(2);
                while(ADCSRA & (1<<ADSC));
                prevodU = ADCL;
                prevodU += (ADCH<<8);
/*
                ADCstart(3);
                while(ADCSRA & (1<<ADSC));
                prevodI = ADCL;
                prevodI += (ADCH<<8);
*/
                napeti=prevodU*Vref*4;
                e0=Vw-napeti;

//vypocet PSD
                uk=(kp*e0-kp*e1+50*ki*e0+kd*10000*(e0-2*e1+e2))/1000+uk;

//omezeni uk
                if(uk>500)
                        uk=500;
                if(uk<-500)
                        uk=-500;

                pocet++;

//              proud =  prevodI*Vref*4 - napeti;
                proud = uk+500;

//ulozeni dat pokud byla predchozi odeslana
                if(poslani==0 && pocet>99){
                        poslani = 1;
                        znak[0]='U';
                        znak[1]=(char)(48+(int)(napeti/10000));
                znak[2]=(char)(48+(int)((napeti%10000)/1000));
                znak[3]=(char)(48+(int)((napeti%1000)/100));
                znak[4]=(char)(48+(int)((napeti%100)/10));
                znak[5]=(char)(48+(int)(napeti%10));
                        znak[6]='I';
                znak[7]=(char)(48+(int)(proud/1000));
                znak[8]=(char)(48+(int)((proud%1000)/100));
                znak[9]=(char)(48+(int)((proud%100)/10));
                znak[10]=(char)(48+(int)(proud%10));
                        znak[11]='S';
                znak[12]=(char)(48+(int)(pocet/100));
                znak[13]=(char)(48+(int)((pocet%100)/10));
                znak[14]=(char)(48+(int)(pocet%10));
                        znak[15]='K';
                        pocet = 0;
                }

        }

}
```

```
////////////////////////////////////////////////////////
//////////////////SERIOVA KOMUNIKACE - UART/////////////
////////////////////////////////////////////////////////


// prijmuti znaku
unsigned char uartRX( void )
{
    /* Wait for data to be received */
   while ( !(UCSRA & (1<<RXC)) ) ;

    /* Get and return received data from buffer */
    return UDR;
}


// odeslani znaku
void uartTX(unsigned char data)
{
    while ( !( UCSRA & (1<<UDRE)) ) ;
    UDR = data;
}
```

2. Počítač

```
/*
 * Razhrani.java
 *
 * Created on 13.4.2011, 14:38:45
 */
package grafickerozhrani;

/*
 *Hlavni trida
 *Trida popisujici graficke rozhrani
*/

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

/**
 *
 * @author Petr
 */
public class Rozhrani extends javax.swing.JFrame {

    public static Graphics g;
    public static Rozhrani r;
    private int vyber;
    private Komunikace kom;

    /** Creates new form Razhrani */
    public Rozhrani() {
        initComponents();
        vyber = 1;
        this.jButton1.setVisible(true);
        this.jButton2.setVisible(true);
        this.jButton3.setEnabled(false);
        this.jButton1.setVisible(true);
        this.jRadioButton1.setSelected(true);
        this.jLabel6.setVisible(false);
        this.jLabel7.setVisible(false);
        this.jLabel8.setVisible(false);
        this.jTextField4.setVisible(false);
        this.jTextField5.setVisible(false);
        this.jTextField6.setVisible(false);
        this.jLabel10.setVisible(false);
        this.jLabel9.setVisible(false);
        this.jTextField7.setVisible(false);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jTextField3 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLayeredPane1 = new javax.swing.JLayeredPane();
        jPanel2 = new javax.swing.JPanel();
        jRadioButton1 = new javax.swing.JRadioButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jSeparator2 = new javax.swing.JSeparator();
        jPanel3 = new javax.swing.JPanel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jRadioButton2 = new javax.swing.JRadioButton();
        jLabel8 = new javax.swing.JLabel();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jTextField6 = new javax.swing.JTextField();
        jLabel9 = new javax.swing.JLabel();
        jTextField7 = new javax.swing.JTextField();
        jLabel10 = new javax.swing.JLabel();
        jSeparator3 = new javax.swing.JSeparator();
        jSeparator1 = new javax.swing.JSeparator();
        jSeparator4 = new javax.swing.JSeparator();
        jSeparator5 = new javax.swing.JSeparator();
        jSeparator6 = new javax.swing.JSeparator();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jTextField8 = new javax.swing.JTextField();
        jLabel13 = new javax.swing.JLabel();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
setResizable(false);
addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowOpened(java.awt.event.WindowEvent evt) {
        formWindowOpened(evt);
    }
});

jButton2.setText("Zacatek mereni");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Ukoncit mereni");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton1.setText("KONEC");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jPanel1.setBackground(new java.awt.Color(0, 0, 0));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 752, Short.MAX_VALUE)
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 241, Short.MAX_VALUE)
);

jTextField3.setText("10");
jTextField3.setPreferredSize(new java.awt.Dimension(50, 20));
jTextField3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField3ActionPerformed(evt);
    }
});

jLabel3.setText("Vstupni napeti menice <0-13>V");

jLabel4.setText("Uin = ");

jLabel5.setText("V");

jPanel2.setMinimumSize(new java.awt.Dimension(260, 129));
jPanel2.setPreferredSize(new java.awt.Dimension(260, 129));
jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jRadioButton1.setText("Dvoupolohovy regulator");
jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton1ActionPerformed(evt);
    }
});
jPanel2.add(jRadioButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0,
    -1, -1));

jLabel1.setText("wh (horni mez hystereze)");
jPanel2.add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 40, -1,
    20));

jLabel2.setText("wd (dolni mez hystereze)");
jPanel2.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 80, -1,
    20));

jTextField1.setText("5");
jTextField1.setPreferredSize(new java.awt.Dimension(50, 20));
jTextField1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1ActionPerformed(evt);
    }
});
jPanel2.add(jTextField1, new org.netbeans.lib.awtextra.AbsoluteConstraints(190, 40,
    -1, -1));

jTextField2.setText("5");
jTextField2.setAutoscrolls(false);
jTextField2.setPreferredSize(new java.awt.Dimension(50, 20));
jPanel2.add(jTextField2, new org.netbeans.lib.awtextra.AbsoluteConstraints(190, 80,
    -1, -1));

jPanel3.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
jLabel6.setText("kp  = ");
jPanel3.add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 30, -1,
    20));

jLabel7.setText("ki   = ");
jPanel3.add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 60, -1,
    20));

jRadioButton2.setText("PSD regulator");
jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton2ActionPerformed(evt);
    }
});
jPanel3.add(jRadioButton2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0,
    -1, -1));

jLabel8.setText("kd  = ");
jPanel3.add(jLabel8, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 90, -1,
    20));

jTextField4.setText("0.08");
jTextField4.setPreferredSize(new java.awt.Dimension(50, 20));
jTextField4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField4ActionPerformed(evt);
    }
});
jPanel3.add(jTextField4, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 30,
    -1, -1));

jTextField5.setText("20");
jTextField5.setAutoscrolls(false);
jTextField5.setPreferredSize(new java.awt.Dimension(50, 20));
jPanel3.add(jTextField5, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 60,
    -1, -1));

jTextField6.setText("0.0001988");
jTextField6.setAutoscrolls(false);
jTextField6.setPreferredSize(new java.awt.Dimension(50, 20));
jPanel3.add(jTextField6, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 90,
    -1, -1));

jLabel9.setText("V");
jPanel3.add(jLabel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(190, 130, 10,
    20));

jTextField7.setText("5");
jTextField7.setAutoscrolls(false);
jTextField7.setPreferredSize(new java.awt.Dimension(50, 20));
jPanel3.add(jTextField7, new org.netbeans.lib.awtextra.AbsoluteConstraints(130, 130,
    -1, -1));

jLabel10.setText("Zadana hodnota w =");
jPanel3.add(jLabel10, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 130, -1,
    20));

jSeparator1.setOrientation(javax.swing.SwingConstants.VERTICAL);

jSeparator4.setOrientation(javax.swing.SwingConstants.VERTICAL);

jLabel11.setText("Vystupni napeti Uout");

jLabel12.setText("Uout = 5 V");

jTextField8.setText("C:\\Users\\Petr\\Desktop\\Vystup.txt");
jTextField8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField8ActionPerformed(evt);
    }
});

jLabel13.setText("Vystupni soubor:");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(24, 24, 24)
        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
            GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(53, 53, 53)
        .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 22, javax.
            swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
            GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(34, 34, 34)
        .addComponent(jSeparator4, javax.swing.GroupLayout.PREFERRED_SIZE, 11, javax.
            swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
```

```
                .addComponent(jSeparator6, javax.swing.GroupLayout.Alignment.TRAILING,
                    javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE)
                .addComponent(jSeparator5, javax.swing.GroupLayout.DEFAULT_SIZE, 107,
                    Short.MAX_VALUE)
                .addComponent(jButton3)
                .addComponent(jButton2)
                .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.
                    MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLayeredPane1)
            .addContainerGap())
        .addComponent(jSeparator2, javax.swing.GroupLayout.DEFAULT_SIZE, 752, Short.
            MAX_VALUE)
        .addComponent(jSeparator3, javax.swing.GroupLayout.DEFAULT_SIZE, 752, Short.
            MAX_VALUE)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel3)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                        279, Short.MAX_VALUE)
                    .addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE, 217,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(95, 95, 95))
                .addGroup(layout.createSequentialGroup()
                    .addGap(32, 32, 32)
                    .addComponent(jLabel4)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel5)
                    .addContainerGap(617, Short.MAX_VALUE))))
        .addComponent(jPanel , javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
            createSequentialGroup()
            .addContainerGap(164, Short.MAX_VALUE)
            .addComponent(jLabel13, javax.swing.GroupLayout.PREFERRED_SIZE, 104, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField8, javax.swing.GroupLayout.PREFERRED_SIZE, 418, javax
                .swing.GroupLayout.PREFERRED_SIZE)
            .addGap(62, 62, 62))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
            createSequentialGroup()
            .addContainerGap(337, Short.MAX_VALUE)
            .addComponent(jLabel11)
            .addGap(315, 315, 315))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(15, 15, 15)
            .addComponent(jLabel11)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jPanel , javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
                GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel12)
            .addGap(10, 10, 10)
            .addComponent(jLabel3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                BASELINE)
                .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.
                    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
                .addComponent(jLabel4)
                .addComponent(jLabel5))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.
                swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
                .addComponent(jLayeredPane1, javax.swing.GroupLayout.Alignment.TRAILING,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 194, Short.MAX_VALUE)
                .addComponent(jPanel , javax.swing.GroupLayout.PREFERRED_SIZE, javax.
                    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
                .addComponent(jPanel , javax.swing.GroupLayout.PREFERRED_SIZE, javax.
                    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
                .addComponent(jSeparator4, javax.swing.GroupLayout.Alignment.TRAILING,
                    javax.swing.GroupLayout.DEFAULT_SIZE, 194, Short.MAX_VALUE)
                .addGroup(layout.createSequentialGroup()
                    .addGap(7, 7, 7)
                    .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(jSeparator5, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
                                10, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(15, 15, 15)
                            .addComponent(jSeparator6, javax.swing.GroupLayout.PREFERRED_SIZE,
                                10, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
                                javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING,
                            javax.swing.GroupLayout.DEFAULT_SIZE, 194, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jSeparator3, javax.swing.GroupLayout.PREFERRED_SIZE, 5, javax.
                    swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    BASELINE)
                    .addComponent(jTextField8, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel13))
                .addGap(79, 79, 79))
        );

        pack();
    }// </editor-fold>

//Ukonceni aplikace
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}

//Spusteni komunikace mezi PC a ATmega8
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        this.jButton3.setEnabled(true);
        this.jButton2.setEnabled(false);
        kom = new Komunikace(((int) (Double.parseDouble(this.jTextField3.getText()) * 1000))
            + "",
                ((int) (Double.parseDouble(this.jTextField4.getText()) * 1000)) + "",
                ((int) (Double.parseDouble(this.jTextField5.getText()) * 1000)) + "",
                ((int) (Double.parseDouble(this.jTextField6.getText()) * 1000)) + "",
                ((int) (Double.parseDouble(this.jTextField7.getText()) * 1000)) + "",
                ((int) (Double.parseDouble(this.jTextField1.getText()) * 1000)) + "",
                ((int) (Double.parseDouble(this.jTextField2.getText()) * 1000)) + "",
                vyber,
                this.jTextField8.getText());
        new Thread(kom).start();
    }

//Dvoupolohova regulace
    private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        vyber = 1;
        this.jRadioButton1.setSelected(true);
        this.jRadioButton2.setSelected(false);
        this.jLabel6.setVisible(false);
        this.jLabel7.setVisible(false);
        this.jLabel8.setVisible(false);
        this.jTextField4.setVisible(false);
        this.jTextField5.setVisible(false);
        this.jTextField6.setVisible(false);
        this.jLabel1.setVisible(true);
        this.jLabel2.setVisible(true);
        this.jTextField1.setVisible(true);
        this.jTextField2.setVisible(true);
        this.jLabel10.setVisible(false);
        this.jLabel9.setVisible(false);
        this.jTextField7.setVisible(false);
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
//PSD regulace
    private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        vyber = 0;
        this.jRadioButton2.setSelected(true);
        this.jRadioButton1.setSelected(false);
        this.jLabel6.setVisible(true);
        this.jLabel7.setVisible(true);
        this.jLabel8.setVisible(true);
        this.jTextField4.setVisible(true);
        this.jTextField5.setVisible(true);
        this.jTextField6.setVisible(true);
        this.jLabel1.setVisible(false);
        this.jLabel2.setVisible(false);
        this.jTextField1.setVisible(false);
        this.jTextField2.setVisible(false);
        this.jLabel10.setVisible(true);
        this.jLabel9.setVisible(true);
        this.jTextField7.setVisible(true);
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```
    }

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
//Ukonceni mereni
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        this.jButton3.setEnabled(false);
        this.jButton2.setEnabled(true);
        kom.stop = false;
    }

//povoleni grafiky pri spusteni aplikace
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        g = this.getGraphics();
        addKeyListener(new KeyAdapter() {

            KeyEvent evt;
        });
    }

    private void jTextField8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
//Vykresleni grafu (mrizky)
    public static void vykresleniGrafu(Graphics G) {
        g.setColor(Color.BLACK);
        g.fillRect(40, 80, 745, 200);
        g.setColor(Color.GRAY);
        for (int i = 40; i < 750; i += 10) {
            g.drawLine(i, 75, i, 295);
        }
        for (int i = 80; i < 300; i += 10) {
            g.drawLine(35, i, 745, i);
        }
        g.setColor(Color.WHITE);
        g.drawLine(35, 290, 745, 290);
        g.drawLine(40, 75, 40, 295);
    }

//Vykresleni bodu, ktery bzl prijat od ATmega8
    public static void vykresleniPrubehu(Graphics G, int x, double y, int t) {
        G.setColor(Color.RED);
        int y2 = (int) (280 - y / 70);
        if (y2 > 14) {
            y2 = 80;
        }
        if (y2 > 280) {
            y2 = 280;
        }
        if (t % 250 == 0) {
            jLabel12.setText("Uout = " + y / 1000 + "V   t = " + t + " ms");
        }
        G.fillOval(x, y2, 2, 2);

    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    public static javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JLayeredPane jLayeredPane1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JSeparator jSeparator5;
    private javax.swing.JSeparator jSeparator6;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
```

```
    private javax.swing.JTextField jTextField7;
    private javax.swing.JTextField jTextField8;
    // End of variables declaration
}
```

```
package grafickerozhrani;

/*
 *Trida pro komunikaci s mikroporcesorem
*/

import gnu.io.*;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Enumeration;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Petr
 */
public class Komunikace implements Runnable {

    String ulozeni;
    String data;
    InputStream in;
    static Enumeration portList;
    static CommPortIdentifier portId;
    String messageString = null;
    static SerialPort serialPort;
    static OutputStream outputStream;
    static boolean outputBufferEmptyFlag = false;
    public boolean stop;

    //Konstruktor
    public Komunikace(String Uin, String kp, String ki, String kd, String w, String wh,
         String wd, int typRegulace, String cesta) {
        stop = true;
        ulozeni = cesta;
        if (typRegulace == 1) {
            messageString = "H" + wh + "S" + wd + "U" + Uin + "K";
        } else {
            messageString = "P" + kp + "I" + ki + "D" + kd + "W" + w + "U" + Uin + "K";
        }
        System.out.println(messageString);
    }

    public void run() {
        //Nalezeni portu USB prevodniku a nastaveni prevodniku
        while (messageString == null) {
        }
        boolean portFound = false;
        String defaultPort = "COM45";
        portList = CommPortIdentifier.getPortIdentifiers();

        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {

                if (portId.getName().equals(defaultPort)) {
                    System.out.println("Found port " + defaultPort);

                    portFound = true;

                    try {
                        serialPort =
                                (SerialPort) portId.open("SimpleWrite", 2000);
                        try {
                            InputStream in = serialPort.getInputStream();
                        } catch (IOException ex) {
                            Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE,
                                 null, ex);
                        }
                    } catch (PortInUseException e) {
                        System.out.println("Port in use.");

                        continue;
                    }

                    try {
                        outputStream = serialPort.getOutputStream();
                    } catch (IOException e) {
                    }

                    try {
                        serialPort.setSerialPortParams(19200,
                                SerialPort.DATABITS_8,
                                SerialPort.STOPBITS_1,
                                SerialPort.PARITY_NONE);
                    } catch (UnsupportedCommOperationException e) {
                    }


                    try {
                        serialPort.notifyOnOutputEmpty(true);
                    } catch (Exception e) {
                        System.out.println("Error setting event notification");
```

```
                            System.out.println(e.toString());
                            System.exit(-1);
                        }

                        //Odeslani parametru do ridiciho mikroprocesoru
                        try {
                            zadaniParametru();
                        } catch (IOException ex) {
                            Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE, null,
                                ex);
                        }

                        try {
                            //Potvrzeni spravnosti dat
                            try {
                                potvrzeni(serialPort.getInputStream());
                            } catch (IOException ex) {
                                Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE,
                                    null, ex);
                            }
                        } catch (InterruptedException ex) {
                            Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE, null,
                                ex);
                        }

//                      TestovaciData t = new TestovaciData(serialPort, outputStream);
//                      new Thread(t).start();

                        //Ocekavani prichodu merenych dat
                        try {
                            try {
                                prijem(serialPort.getInputStream());
                            } catch (InterruptedException ex) {
                                Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE,
                                    null, ex);
                            }
                        } catch (IOException ex) {
                            Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE, null,
                                ex);
                        }
                        try {
                            ulozeniDat();
                        } catch (IOException ex) {
                            Logger.getLogger(Komunikace.class.getName()).log(Level.SEVERE, null,
                                ex);
                        }
                        serialPort.close();
                    }
                }
            }

            if (!portFound) {
                System.out.println("port " + defaultPort + " not found.");
            }
        }

    //Odeslani zpravy s daty o regulatoru
        private void zadaniParametru() throws IOException {
            System.out.println("Writing " + messageString + " to " + serialPort.getName());
            try {
                outputStream.write(messageString.getBytes());
            } catch (IOException e) {
            }


        }

    //Potvrzeni spravnosti regulatoru
        private void potvrzeni(InputStream in) throws IOException, InterruptedException {
            this.in = in;
            byte[] buffer = new byte[1];
            int len = -1;
            boolean dal = true;
            String data1 = "";
            System.out.println("potvrzeni");
            try {
                while ((len = this.in.read(buffer)) > -1 && dal) {
                    data1 = data1 + (new String(buffer, 0, len));
                    if ((data1.length() == messageString.length()) && data1.equals(messageString)
                        ) {
                        outputStream.write(messageString.substring(0, 1).getBytes());
                        outputStream.write(messageString.substring(0, 1).getBytes());
                        outputStream.write("K".getBytes());
                        System.out.println("odeslani OK ");
                        dal = false;
                    } else {
                        outputStream.write("K".getBytes());
                        outputStream.write("K".getBytes());
                        outputStream.write("K".getBytes());
                        System.out.println("odeslani KO ");
                    }
                }
            } catch (IOException e) {
            }
        }
```

```
//Prijem merenych dat
    private void prijem(InputStream in) throws IOException, InterruptedException {
        this.in = in;
        int u = 0, i = 0, s = 0;
        int x = 741, j = -1;
        int t = 0;
        byte[] buffer = new byte[1];
        int len = -1, y = 0;
        long cas = System.currentTimeMillis();
        String data1 = "";


        try {
            data = "";
            while ((len = this.in.read(buffer)) > -1 && stop) {
                data1 = data1 + (new String(buffer, 0, len));
                if (len > 0) {
                    j++;

                }

                System.out.println(data1);

                //Parsovani zpravy a ocekavani charakteristickych znaku
                if (data1.endsWith("U")) {
                    u = j;
                }
                if (data1.endsWith("I")) {
                    i = j;
                }
                if (data1.endsWith("S")) {
                    s = j;
                }
                if (data1.endsWith("K")) {
                    //Vykresleni grafu a ulozeni dat
                    if (x > 739) {
                        x = 40;
                        Rozhrani.vykresleniGrafu(Rozhrani.g);
                    }

                    System.out.println(u + " " + i + " " + s + "\n");
                    if ((u >= 0) && (u + 1 < i) && (i + 1 < s) && (s + 1 < j) && (j < 16)) {
                        y = parsovani(data1, u + 1, i);
                        Rozhrani.vykresleniPrubehu(Rozhrani.g, x, y, t);
                        x++;
                        data = data + t + " " + y + " " + parsovani(data1, i + 1, s) + " " +
                             parsovani(data1, s + 1, j) + "\n";
                        t += 10;
                        System.out.println("cas = " + (System.currentTimeMillis() - cas));
                    }
                    data1 = "";
                    j = -1;
                }
                /*                  if(j==4){
                System.out.println("U" + (int)(data1.charAt(0)) + "" + (int)(data1.charAt(1))
                      + "I"  + (int)(data1.charAt(2)) + "" + (int)(data1.charAt(3)) + "S"  + (
                     int)(data1.charAt(4)) + "K" );
                j=-1;
                }*/
            }
        } catch (IOException e) {
        }
    }

//Parsovani ciselnych hodnot
    private int parsovani(String data, int i1, int i2) {
        int y = 0;
        boolean parsuj = true;
        for (int i = i1; i < i2; i++) {
            if ((int) (data.charAt(i)) < 48 || (int) (data.charAt(i)) > 57) {
                parsuj = false;
            }
        }
        if (parsuj) {
            y = Integer.parseInt(data.substring(i1, i2));
        }
        return y;
    }

//Ulozeni dat do souboru
    private void ulozeniDat() throws IOException {
        FileWriter soubor = new FileWriter(ulozeni);
        soubor.write(data);
        soubor.close();
    }
}
```