České vysoké učení technické v Praze

fakulta elektrotechnická

katedra mikroelektroniky

Diplomová práce

Výkonový generátor funkcí řízený pomocí PC

Autor: Bc. Adam Půta

Vedoucí práce: Ing. Luboš Jirásek, CSc.

2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Půta** Jméno: **Adam** Osobní číslo: **393089**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Výkonový generátor funkcí řízený počítačem

Název diplomové práce anglicky:

Function High-power Generator Controlled by Computer

Pokyny pro vypracování:

1. Prostudujte dostupnou literaturu týkající se generátorů funkcí a napájecích zdrojů
2. Na základě 1) navrhněte a realizujte generátor funkcí (sinus, obdélník/lichoběžník, trojúhelník), frekvence 10 Hz - 100 kHz, schopný dodávat průběhy 0 V - 100 V. Pokuste se dosáhnout výstupního proudu 10 A.
3. Proveďte ověřovací měření.
4. Zhodnoťte dosažené výsledky.
5. Navrhněte případné další změny zapojení.
6. Zařízení zůstane v majetku zadávajícího pracoviště.
7. Publikování výsledků dosažených v této práci je možné pouze se svolením zadavatele.

Seznam doporučené literatury:

[1] Krejčiřík, A.: Napájecí zdroje I. - III., BEN, Praha 2003 a další vydání.
[2] Aplikační poznámky fy IRF (Infineon)
[3] Balogh, L.: Design and Application Guide for High Speed MOSFET Gate Drive Circuits. (www.ti.com/lit/ml/slup169/slup169.pdf), leden 2014

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Lubor Jirásek CSc., katedra mikroelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **15.02.2017** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **10.09.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání	Podpis studenta

Abstrakt

Cílem této diplomové práce je návrh a realizace výkonového generátoru funkcí, s možností nastavení všech známých parametrů signálu. Od amplitudy, přes fázi, až po frekvenci. Jádro zařízení tvoří snižující měnič, schopný snižovat výstupní napětí od 100V do přibližně 0V. Výstup z generátoru je realizován D/A převodníkem pro vytvarování požadovaného průběhu výstupního signálu. Systém je možné ovládat pomocí grafické aplikace v PC.

Abstract

Goal of this diploma thesis is to develop and build power function generator. We are able to set all output parameters well known from signal processing. These are amplitude, phase and frequency. The core of device is represented by step down converter able to create output signal within the range from 0V to 100V.Output part is responsible for making a requested signal shape. Entire system can be controlled by PC via graphical application.

Klíčové slova

generátor funkcí, tvarování signálu, spínaný zdroj, měření

Key words

function generator, signal shaping, switching power supply, measurement

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci „Výkonový generátor funkcí“ zpracoval sám s přispěním vedoucího práce a používal jsem pouze literaturu uvedenou na konci práce. Souhlasím se zapůjčováním práce a jejím zveřejňováním.
V Praze dne 5. 1. 2018. Bc. Adam Půta

Poděkování

Na úvod mé diplomové práce bych rád poděkoval rodině za projevenou podporu, dále pak vedoucímu práce za cenné připomínky a rady a nakonec také fakultě samotné za možnost používat její laboratoře k vyhotovení práce a měření výstupů z ní vyplývajících.

Obsah

Tabulky

Seznam obrázků

Kapitola 1
Úvod

V moderní elektrotechnice se často neobejdeme bez signálové analýzy, a znalosti odezev různých typů systémů na tyto signály. Od jednoduchých RC, RL, RLC článků, přes široké spektrum filtrů (např. antény), pohonů, až po rozličné typy materiálů, u kterých nás může zajímat, jak se tyto materiály chovají při vystavení specificky modulovaného výkonu, a tím zjištění jejich spektrálních, nebo dalších vlastností. Modulovaným výkonem je myšlen výkonový signál, který má určitý časový průběh. Tento signál může být deterministický, nebo náhodný. V případě deterministického (výkonového, periodického) signálu se setkáváme s parametry, jako je amplituda, fáze, kmitočet a offset. Mezi nejčastěji používané signály patří signály: sinus, pila, trojúhelník, obdélník a některé typy digitálních modulací, jako je například PWM[1], ASK[2], FSK[3] atd.

U našeho generátoru je však možnost dostat na výstupu také konstantní signál. Úkol této diplomové práce je tedy navrhnout a realizovat generátor funkcí (dále GF) signálu schopný nastavovat výše uvedené parametry, a protože výkon generovaného signálu v našem případě může být podstatně větší, než u klasických GF používaných pro nízko výkonové aplikace, mluvíme zde o výkonovém generátoru funkcí (dále VGF). V dnešní moderní době je samozřejmostí možnost komunikace zařízení s počítačem, a proto je součástí této diplomové práce návrh obsluhy VGF pomocí PC.

Z výše uvedených informací tedy plyne, že celý generátor musí být tvořen vstupní částí, zodpovědná za přivedení elektrické energie, dále snižující měnič pro možnost nastavení rozsahu výstupního napětí od přibližně 0 V do 100 V a hodnoty výstupního proudu od přibližně 0A do 10A. Následuje subsystém tvarování výstupního signálu dle požadavku uživatele. Poznamenejme, že výstupní frekvenci je možno zvolit od přibližně 10 Hz do 100 kHz. Nakonec nesmí chybět ovládání celého generátoru pomocí mikroprocesorového systému a komunikační rozhraní pro přenos dat mezi PC a generátorem.

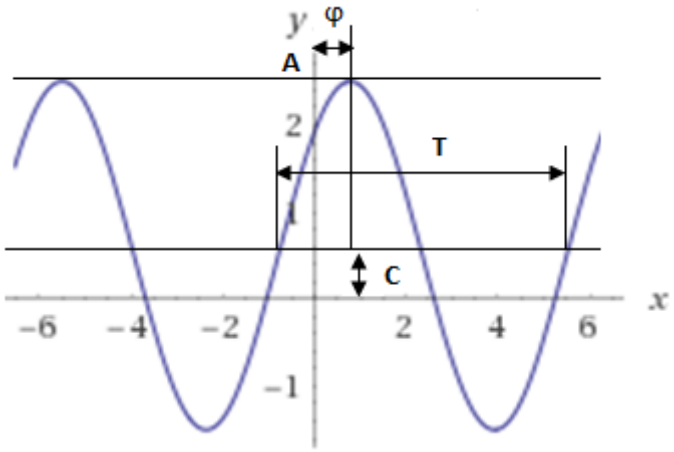
[1] PWM - pulzní šířková modulace
[2] ASK - modulace amplitudovým klíčováním
[3] FSK - modulace frekvenčním klíčováním

Kapitola 2
Generátory funkcí

Generátory funkcí jsou zařízení, schopná generovat na svém výstupu téměř libovolný, v čase proměnný signál. A to ať už se jedná o základní deterministické signály typu sinus, pila, obdélník, přes různé typy digitálních modulací jako je PWM, FSK, PSK, až po generování náhodných (nedeterministických) signálů, nebo signálů, které zadá sám uživatel. Samozřejmě dobrý generátor umí také nastavení offsetu. Základní popis signálu v časové oblasti je následující:

$$y = AS(\omega t + \varphi) + C \tag{2.1}$$

kde A je amplituda signálu, $S(\omega t + \varphi)$ je báze signálu, nebo-li tvar průběhu, φ je fázový posuv, ω je frekvence, a C je offset.



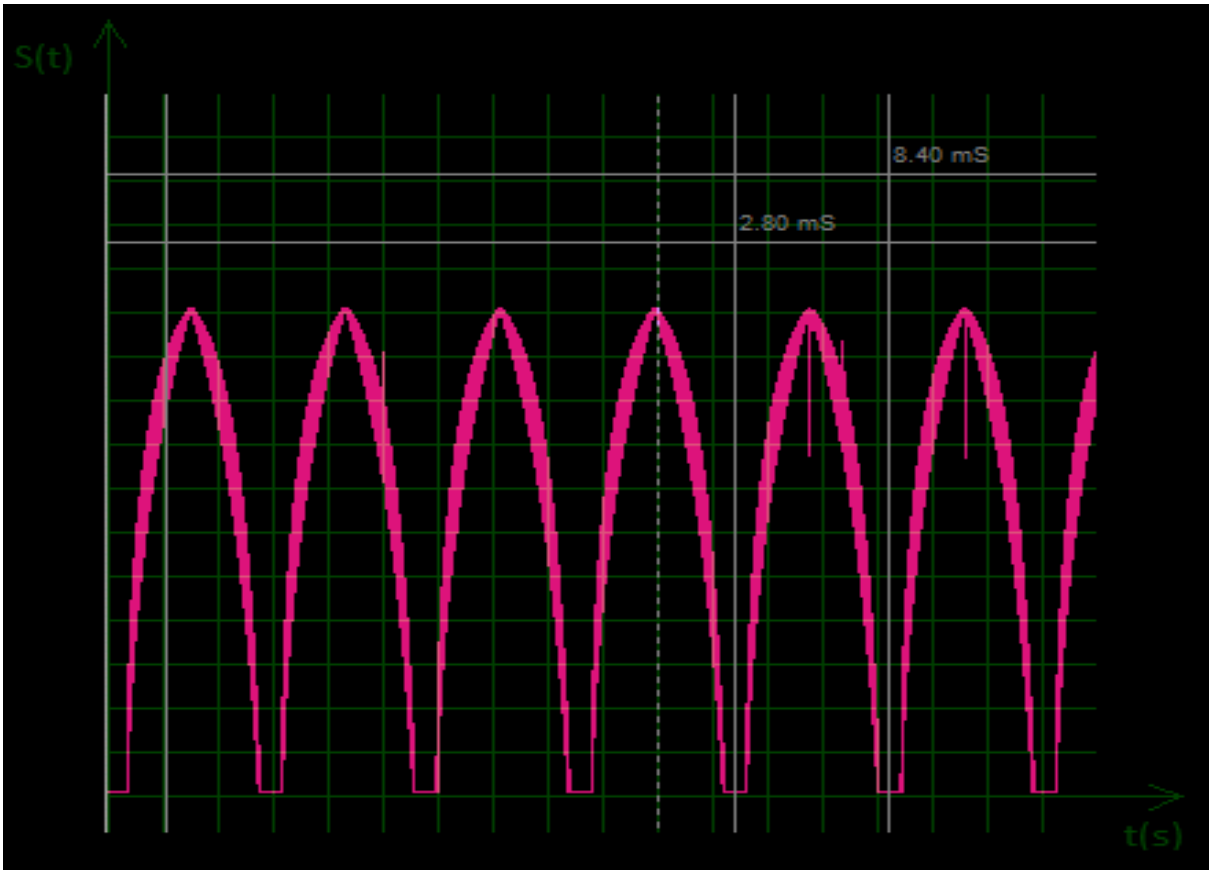
Obr. 2.1: Průběh sinusového signálu s parametry, které lze měnit

2.1 Analogové generování signálu

Generátory funkcí lze obecně rozdělit na analogové a digitální. Analogové využívají buď zapojení s bipolárními tranzistory, nebo síť operačních zesilovačů, a komparátorů ke generování signálu typu trojúhelník, obdélník a sin.Základem je astabilní multivibrátor, který generuje základní signál v podobě obdélníku. Tvarovač signálu, který tvaruje ostatní typy průběhů, je v podstatě zapojení OZ jako integrátor.

2.2 Digitální generování signálu

Již zmíněnou možností je generovat různé typy signálu pomocí digitálních obvodů. První možnost, která mě napadla, byla založená na generování PWM, a tento signál filtrovat. Na obr. 2.2 je vidět ne příliš dobrý výstup signálu z RC filtru, který má být signálem pila, který je generován PWM modulem procesoru.



Obr. 2.2: Signál pila

Tento způsob navíc nedovoluje generovat signály o vyšší frekvenci. Na obrázku 2.2 je vidět perioda signálu 5,6ms. To je frekvence okolo 178Hz. Tento výsledek se dostaví, i když obnovujeme aktivní dobu PWM okamžitě bez prodlev. Obslužné programy procesoru jsou psané v prostředí microC v jazyce C.

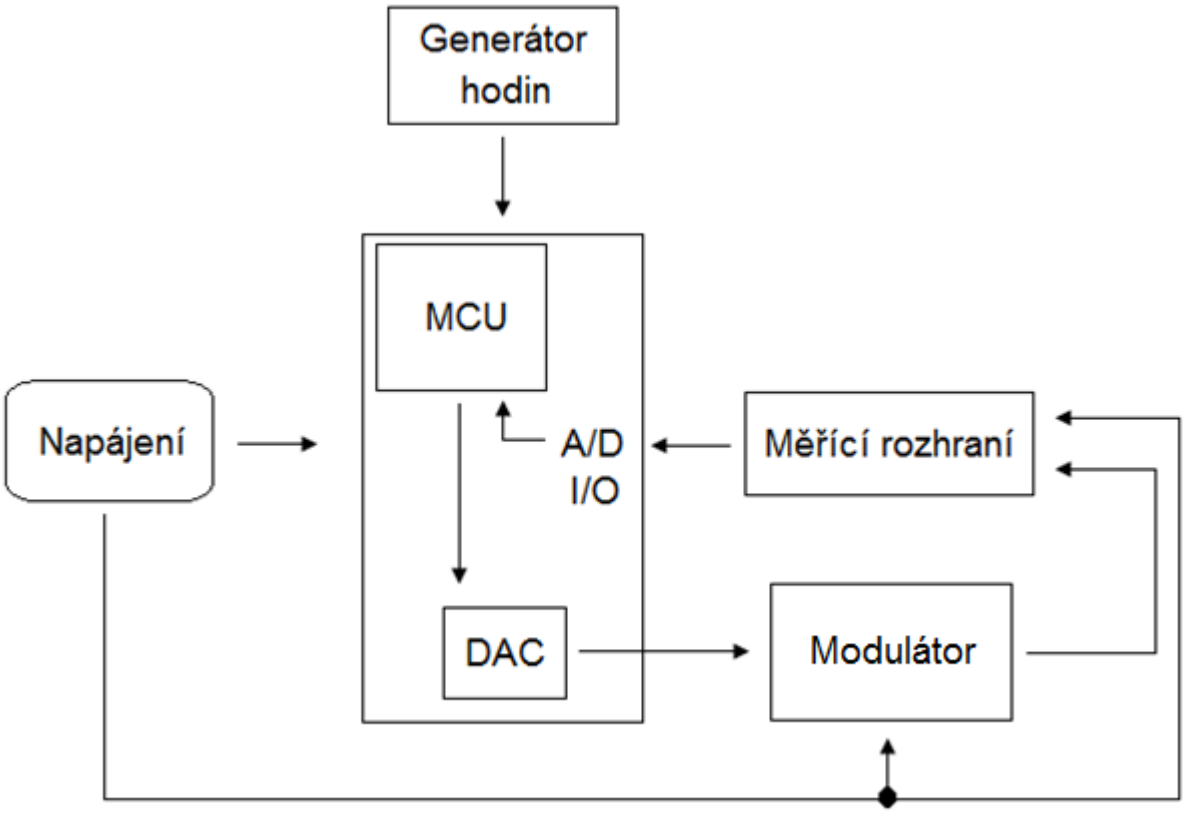
```
void PWM_triangleGen(){
   if(current_duty < 255&&x==0){
   current_duty++;
   PWM1_Set_Duty(current_duty);

   if(current_duty == 254){
   x = 1;
   }
   }
   else{
     current_duty--;
      PWM1_Set_Duty(current_duty);

    if(current_duty==0){

    x=0;
    }
   }
}
```

Obr.2.3: Program pro generování pily pomocí

Lepší je generovat signály pomocí digitálně analogového převodníku. Na obr. 2.4 je zobrazen základní princip syntézy signálu pomocí procesorového systému. Jádro systému tvoří mikropočítač, který generuje požadované tvary signálů pomocí digitálně-analogového převodníku (dále DAC). Nízko výkonový signál je zesílen v bloku modulátoru, který moduluje žádaný výkon na vybraný signál. Měřicí rozhraní převádí výstupní průběh signálu z modulátoru na bezpečnou úroveň.

Pomocí mikropočítače, a jeho DAC modulu generujeme tvary signálu, jejichž vzorky jsou buď uložené v poli proměnných v paměti, nebo se vypočítávají za běhu programu pomocí matematické knihovny.To může být u složitějších průběhů problém, protože to shora omezuje možný kmitočet signálu. Procesoru trvá delší dobu než funkční hodnotu vypočte. K tomu se přičte "doba obnovení" DAC, to součtu dělá spoustu času. V dnešní době se používají specializované obvody jako FPGA, nebo signálové procesory se speciální instrukční sadou[4], které umí signály zpracovávat, a generovat velmi rychle. Pokud je však náš požadovaný maximální kmitočet relativně malý, není to až takový problém.

[4] Instrukční sada je soubor elementárních příkazů, které umí procesor zpracovávat.



Obr. 2.4: Digitální schéma GF

Algoritmus generování číslicově reprezentovaného signálu je vcelku jednoduchý. Následující příklad ukazuje výpočet lineární funkce.

$$S_{[i]} = \sum_{i=0}^{1024} kx_i + q \tag{2.1}$$

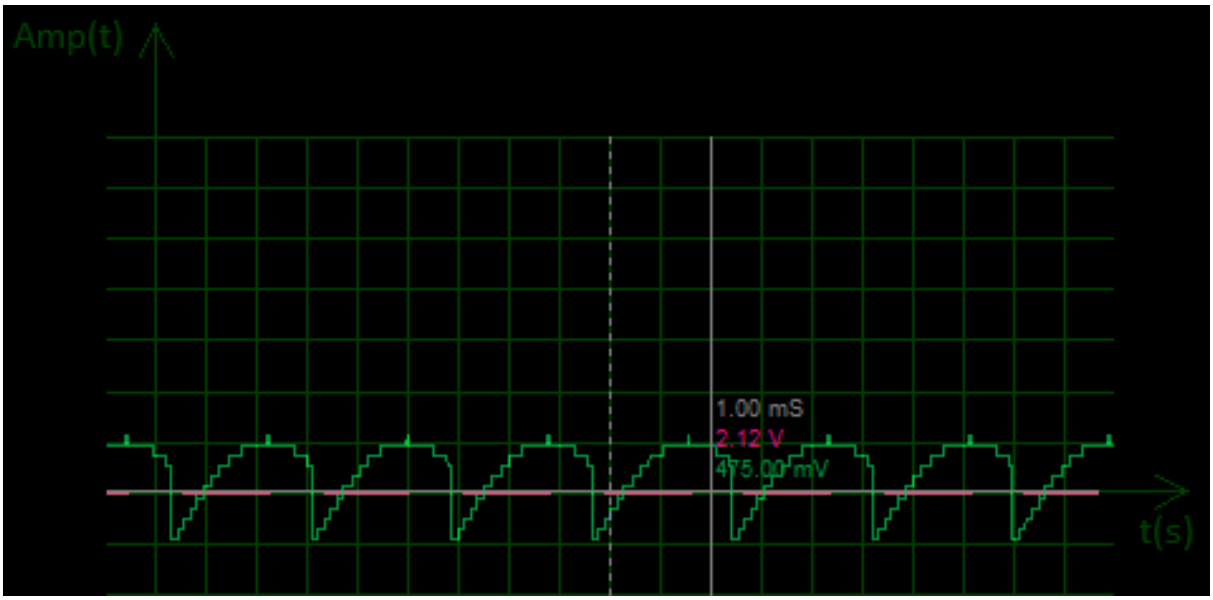
kde i reprezentuje rozsah DAC modulu. Ten je v našem případě 10bitový, a to znamená, že máme k dispozici 2^n napěťových úrovní. N zde reprezentuje číslo 10. Podobně vypočítáme i ostatní typy signálů.

Protože bylo pokusem zjištěno, že matematické knihovny jsou velmi pomalé, rozhodl jsem se generovat signál tak, že například trojúhelník generuji klasickým "For" cyklem. U sinu je situace vůbec nejhorší, protože knihovna jménem sinE3 pracuje tak, že výsledek sinu vynásobí tisícem a zaokrouhlí na nejbližší integer. To s připočtením doby vybavení hodnoty na digitální analogový převodník znamená, že každá hodnota se změní až po cca. desítkách mikrosekund. To značně omezuje frekvenční rozsah. Sinus tedy generuji tak, že jsem si nejprve v programu Visual studiu 2017 od Microsoftu v jazyku C++ vypočetl hodnoty sinu od 0 do 180 stupňů, zaokrouhlil na celé číslo. Tyto

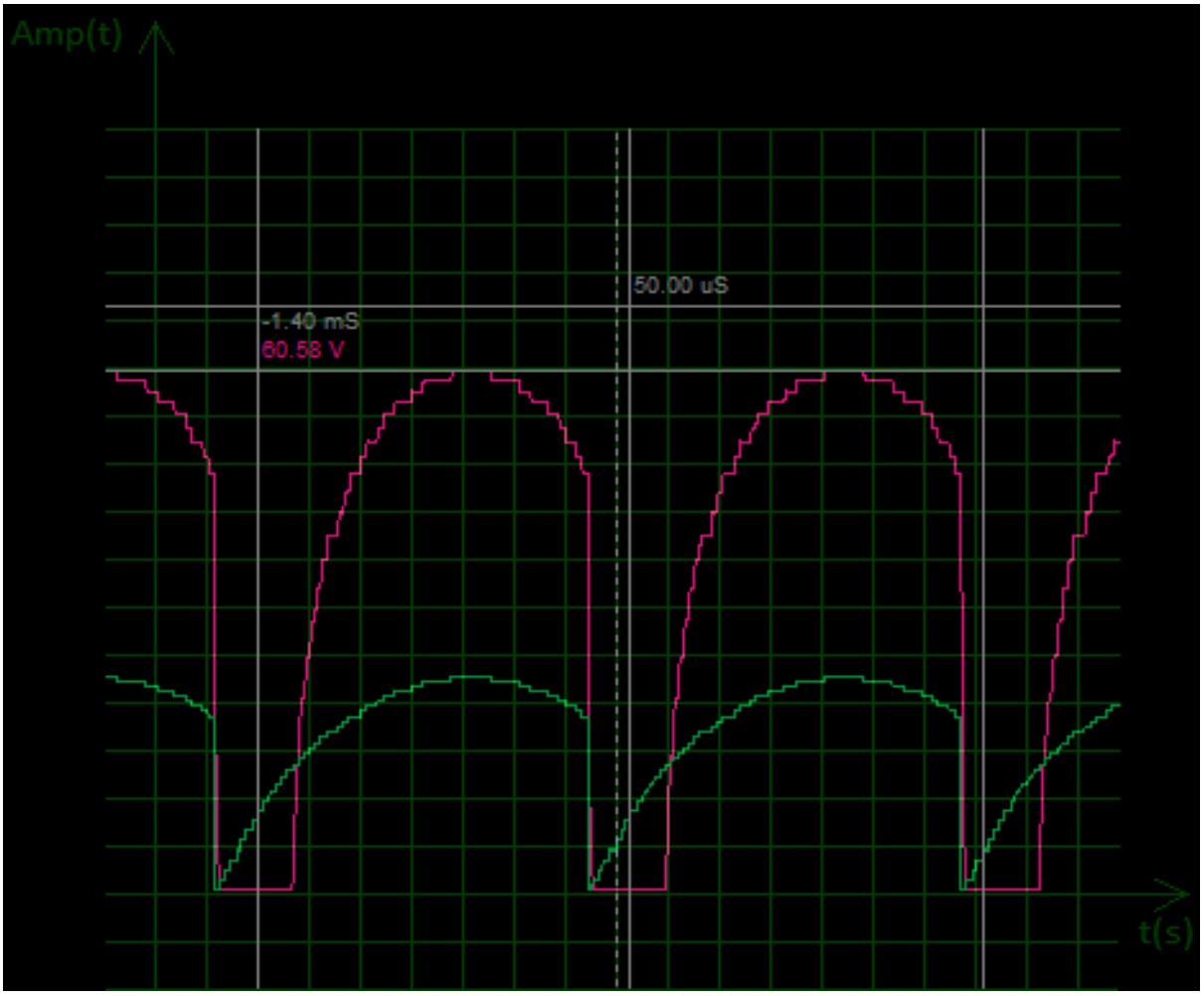
hodnoty jsou nejprve uložené v souboru.txt, a následně zkopírované do podprogramu procesoru pro tvorbu sinu. Při výběru například sinusovky se tyto hodnoty jednoduše ukládají do registru DACxREFH, který ovládá DAC procesoru. Je dobré také myslet na to, že chceme-li měnit amplitudu a frekvenci signálu, rozhodně mezi jednotlivé vzorky signálu musíme vložit čekací smyčky a proměnou pro amplitudu. Na obrázku 2.5 je vidět program pro výpočet a parametrizaci funkce sinus. "For" cyklus provede výpočty na základě proměnné "result". V ní jsou obsažen parametr "range". Ten definuje počet vzorků. Jak je vidět na obrázku, je nastaven na hodnotu 90. Čím nižší hodnota parametru "range" tím bude mít funkce sinus menší počet vzorků. Díky tomu se pak dá zvýšit frekvence signálu generovaného procesorem uvnitř VGF. Nemusí se totiž do DAC posílat tolik vzorků, a tím pádem syntéza signálu trvá kratší dobu. Hodnotu "result" můžeme převést na datový typ Int. "out_file" nám vytvoří soubor .txt, kam řetězec nahrajeme. Hodnotu "x" poté dělíme proměnou "amp", a tím snižujeme amplitudu. Poznamenejme, že vzorky signálu jsou vypočteny tak, že dostaneme signál sinu s maximální amplitudou, a proto ho jí můžeme už jen snižovat. Proměnná "amp" je nastavována v procesoru jako výběr amplitudy. Jelikož prostředí microC neobsahuje funkci "round", plníme registr pro ovládání DAC ne typem Int, ale typem double. V zásadě se ale nic moc neděje. DAC konverze probíhá normálně. Nejspíše to microC řeší za nás. V programu je místo DACxREFH registr VREFCON2. Detailnějším rozborem nastavování registru se zabývat nebudeme. Každý procesor je jiný, a uživatel musí pracovat s návodem od výrobce pro konkrétní typ procesoru. Ten obsluhuje DAC procesoru PIC18F45k22, který byl použit pro simulaci algoritmů. Připomínám, že já v diplomové práci používám PIC16F1777, ale ten v simulátoru není. Simulace prokázala, že cyklus "For" je relativně pomalý, a celý procesor zdržuje. Zkoušel jsem různé postupy jak vytvořit "nenáročnou" čekací smyčku, od "While" cyklu až po bitovou rotaci v registru. Nic ale nemělo patřičný efekt. Pokud bychom chtěli frekvenci výstupního signálu 100kHz, nesmíme tento cyklus použít. Pokud ho použijeme, tak i jedna iterace znamená, že se dostaneme na pouze zhruba 12kHz. Další iterace samozřejmě snižují kmitočet, což je dobře, ale regulace není tak plynulá. Nicméně uvidíme, jak se bude chovat systém při reálném měření. Co se regulace amplitudy týká, funguje celkem bez problému. Pro maximální amplitudu zvolíme hodnotu "amp" proměnné 32. Pro menší amplitudy bude hodnota vyšší. Čím je tedy vyšší, tím nižší amplituda výstupního signálu. To Nižší hodnota než 32 znamená přetečení kontrolního registru DACu a systém se chová nestabilně. Nejvyšší hodnota, kdy je signál ještě patrný je 100. Mám na mysli zesílený signál pomocí H-můstku, protože ten je, jak bude popsáno dále řízen přes rozhraní dvou invertujících zesilovačů třídy A. Signál přímo z DAC je patrný i pro vyšší hodnoty "amp". Prostředí microC obsahuje funkci "Delay_ms" a "Delay_us", nicméně problém je, že parametr určující dobu, ať už v milisekundách nebo mikrosekundách, lze nastavit pouze naplněním konstanty. Nikoliv však proměnou. To diskvalifikuje tuto funkci pro vytváření časového zpoždění na základě výběru frekvence výstupního signálu. Sama funkce je navíc také "pomalá". Za úvahu stojí použití vloženého asembleru pomocí direktivy „_ASM“. Díky instrukci „nop“ by jsme mohli vytvořit čekací smyčku teoreticky o délce pouze jednoho instrukčního celku. Podle zadané frekvence vypočteme hodnotu, kolikrát opakovat instrukci „nop“. Jednodušší způsob je, pokud použijeme podmínku „If“, pomocí které budeme kontrolovat onu hodnotu zda-li je rovna nule. Pokud není, dojde k dekrementaci hodnoty o jedničku a instrukce „nop“ se znovu provede. Podmínka „If“ by měla být méně časově náročná než předchozí cykly „For“ a „While“.

```
sin_fce                    (Globální rozsah)                    main()

#include "stdafx.h"
#include "string"
#include "iostream"
#include "fstream"
#include "math.h"

#define PI 3.14159265

int x;
double result;
int range = 90;


int main()
{

    std::ofstream out_file{ "../Test.txt" };


    for (x = 0; x < range; x++) {
        //result = (round((sin(x*PI / range) * 1000)/32));
        //result = (round((sin(x*PI / range) * 1000))/32);
        result = (round((sin(x*PI / range) * 1000)));
        int x = (int)result;

        //out_file << "VREFCON2 = " << x << ";" << std::endl;
        out_file << "VREFCON2 = " << x << "/amp;" << std::endl;
        //out_file << "VREFCON2 = round((sin(x*PI / range) * 1000) / 32" << std::endl;
        out_file <<"for(freq = 0;freq<refresh_period;freq++){}"<< std::endl;
        //out_file << "while(freq<refresh_period){};" << std::endl;
        //out_file << "variable<<x;" << std::endl;
        //out_file << "rotation();" << std::endl;
    }

    out_file.close();
    return 0;
}

100 %
```

Obr. 2.5: Vypsání hodnot funkce sinus a parametrizace signálu

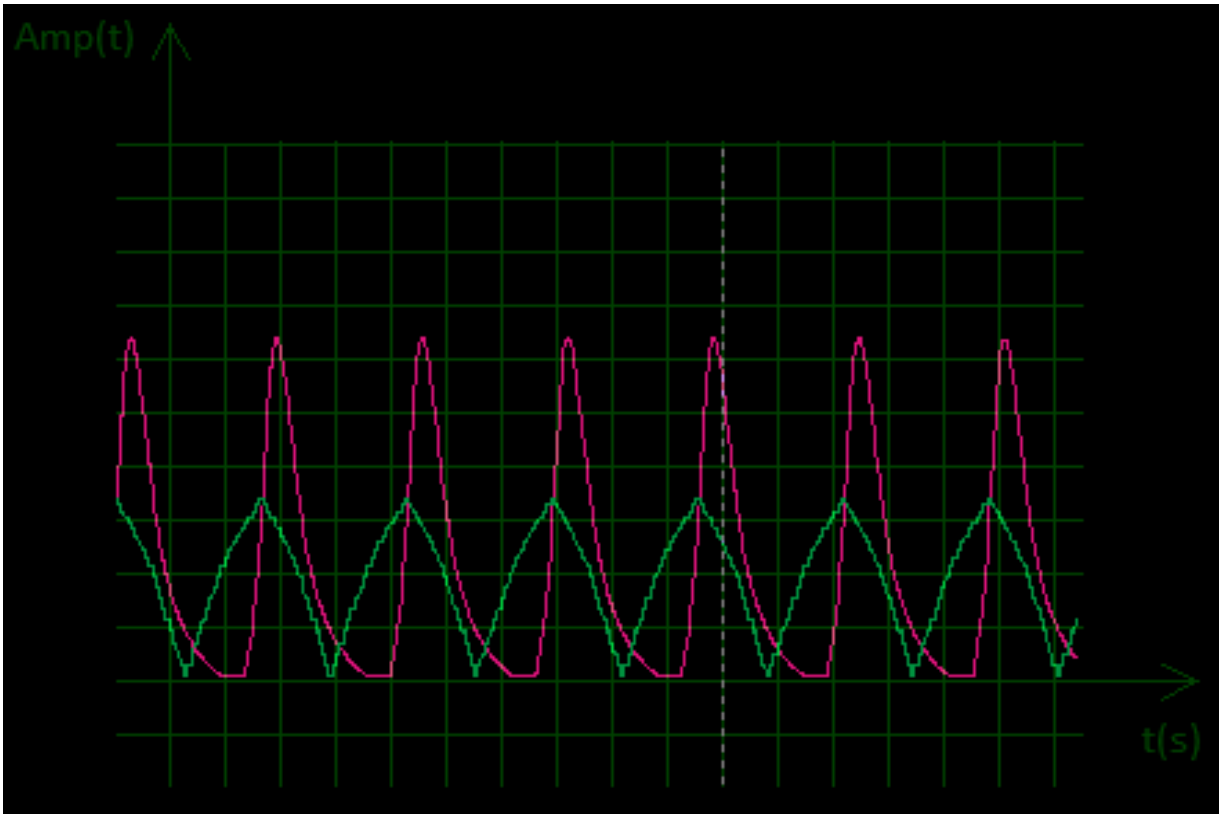


Obr. 2.6a: Nejnižší amplituda signálu(červená: výstup z H-můstku, zelená: výstup z DAC)



Obr. 2.6b: Nejvyšší amplituda při hodnotě 32

Na obr. 2.6b je také vidět vliv doby periody signálu při změně počtu iterací "For" cyklem pro iteraci jedna dostaneme hodnotu periody 1,45ms. Pro počet iterací dva je to 1,5ms atd. Poznamenejme, testovaný signál je sinus od 0 do 180 stupňů. Pro plný sinus v rozsahu 360 stupňů musíme přepínat směr proudu na H-můstku. Důvod, proč je tato horní půlvlna je necelá, je to, že demo verze microC je omezena co do počtu hodnot vzorků, které jsem mohl do programu uložit. Proto signál nepokračuje do 180 stupňů. Regulace amplitudy se paralelně provádí s regulací hodnoty odporu, který je součástí zesilovače třídy A řídící tranzistory H-můstku. Pokud bychom to nedělali, dojde vlivem útlumu signálu ke zvýšení offsetu signálu. Na druhou stranu i nastavení offset může být regulována nejen počáteční hodnotou v DAC (první vzorek má nenulovou hodnotu), ale právě i oním odporem. Odpor bude zastoupen tranzistorem NMOS, který se bude regulovat PWM signálem filtrovaným pomocí RC článku.



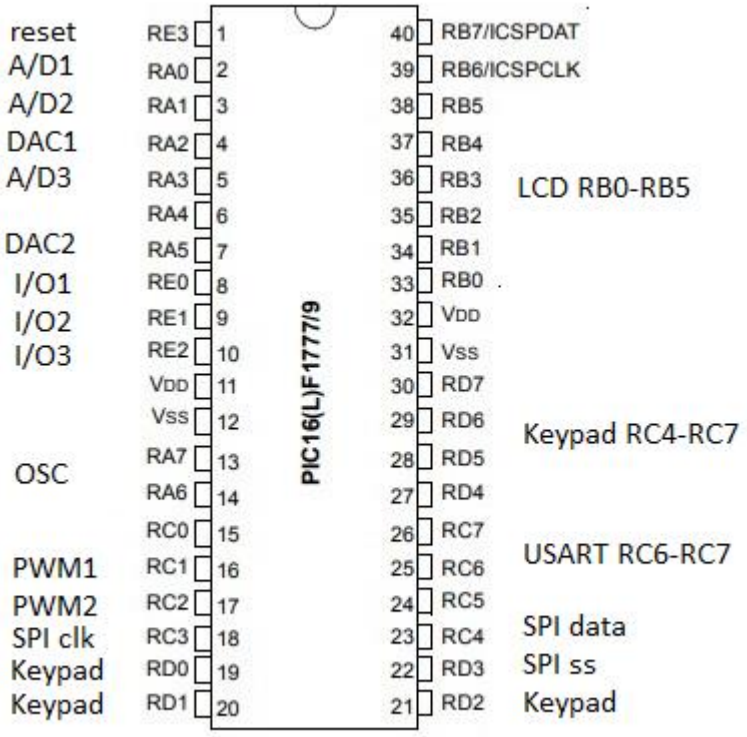
Obr. 2.6c: Průběh signálu trojúhelník

Signál trojúhelníku je symetrický podle osy y je generován pomocí "For" cyklu, kde lineární inkrementací hodnoty signálu roste do maxima. V tomto bodě opět lineárně klesá díky dekrementaci hodnoty. Signál pila se generuje stejně, až na to, že za maximem signálu následuje hodnota 0. Signál rampa se generuje tak, že pomocí změny doby mezi inkrementacemi hodnot registru ovládající DAC měníme sklon přímky. Obr. 2.7 ilustruje výstup programu v podobě .txt souboru. Vliv na deformaci signálu mohou mít nejen rušící vlivy z okolí, ale také optočleny použité pro galvanické oddělení obvodů procesoru a silového obvodu, a také tranzistory v H - můstku.

```
VREFCON2 = 0/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 35/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 70/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 105/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 139/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 174/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 208/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 242/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 276/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 309/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 342/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 375/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 407/amp;
for(freq = 0;freq<refresh_period;freq++){}
```

Obr. 2.7: Txt soubor

Pro úplnost ukažme mapování I/O pinů procesoru. Piny které neplní žádnou funkci jsou uzemněny.

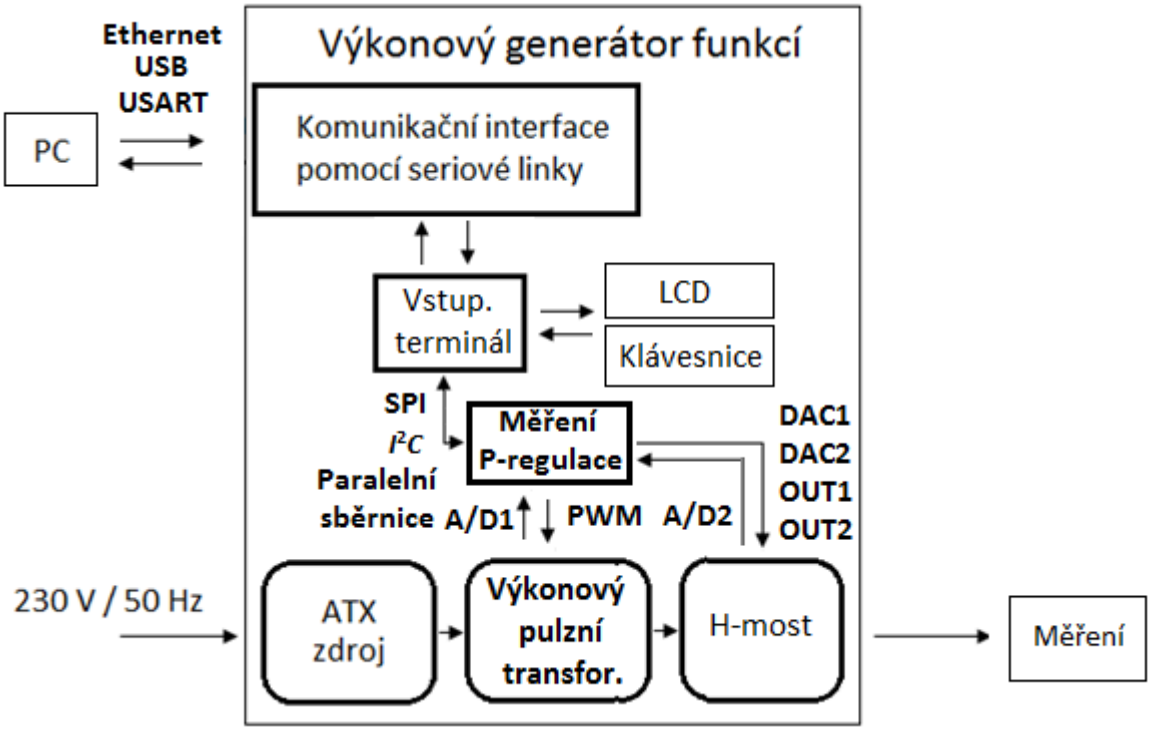


Obr. 2.8: Mapování pinů procesoru

Kapitola 3
Blokové návrhy VGF

V této části si shrneme možné přístupy založené na procesorových systémech. Systémy založené na bázi *FPGA*[5] nebudou kvůli svojí náročnosti a ceně v této práci diskutovány. Dále se podíváme na obsluhu systému.

3.1 Architektura VGF

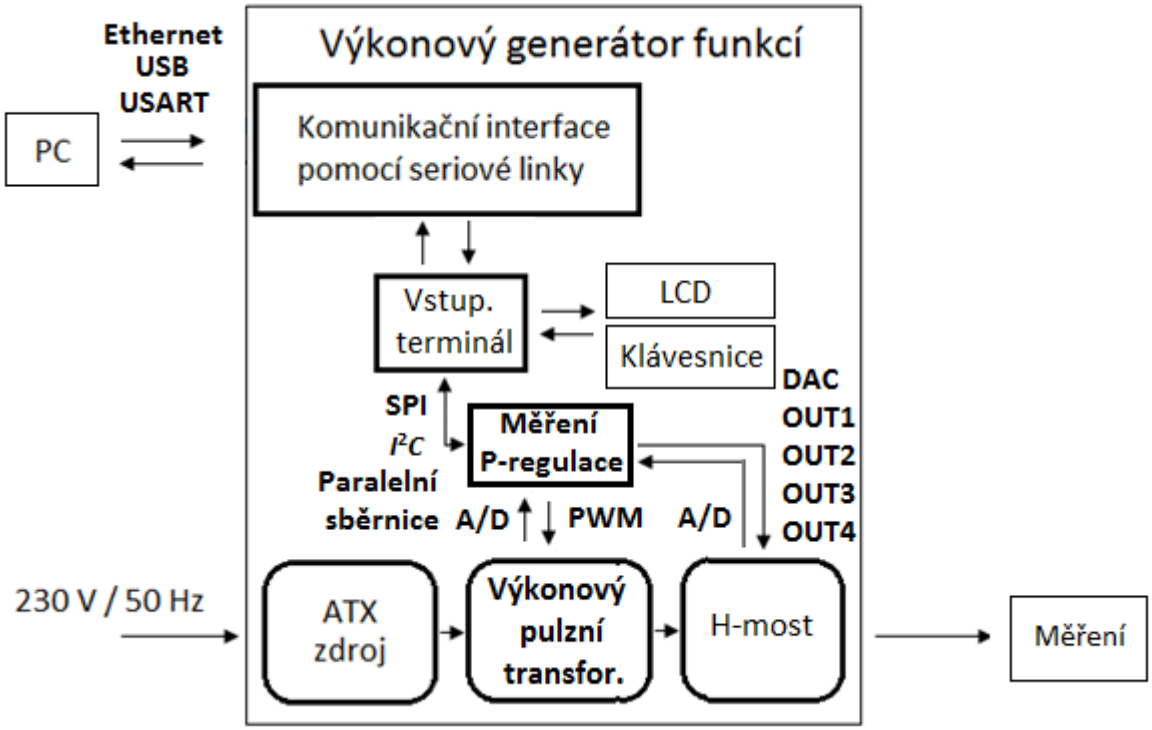


Obr. 3.1a: Multiprocesorová architektury

První architektura je multiprocesorová. Celý systém je řízen mikropočítačem, do kterého se zadávají vstupní parametry pomocí maticové klávesnice. Tyto parametry jsou zobrazeny na LCD[6] displej. Zároveň jsou tyto parametry odeslány do druhého mikropočítače přes datovou sběrnici (a to ať už se

[5] FPGA Field Programmable Gate Array, na rozdíl od sekvenčních procesorů umožňují provádět čistě paralelní zpracování více úkolů
[6] LCD Liquid crystal display

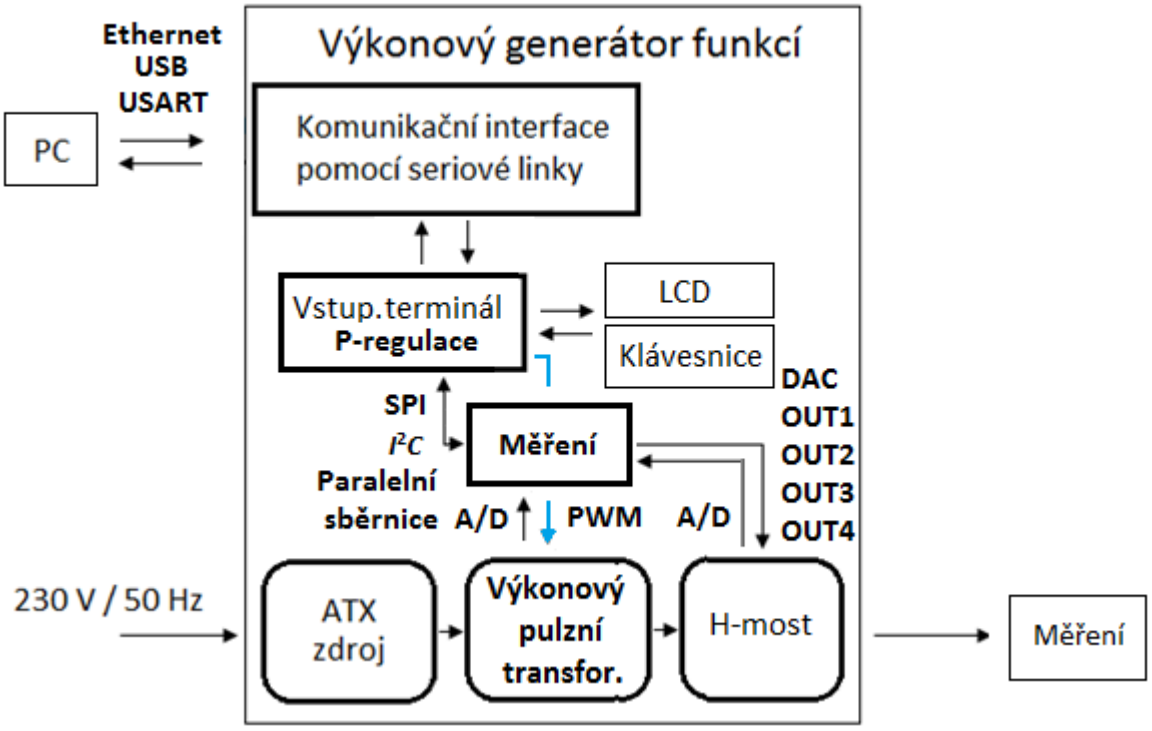
jedná o klasické standardizované přístrojové sériové sběrnice *I²C*, *SPI*, *RS-485*, *One-Wire* nebo o paralelní sběrnice, které si vývojář může definovat i na základě vlastního přenosového protokolu), který slouží jako signálový generátor a měřicí subsystém. Zároveň ovládá výkonový pulzní transformátor. Mezi výhody této architektury patří jistá míra paralelizace, kdy na vstupním terminálu obsluha zadává informace, zatímco druhý mikropočítač dál produkuje řídicí signály, jak PWM pro pulzní transformátor, tak i analogové a digitální signály pomocí dvou DAC (každý ovládá jeden ze dvou horních tranzistorů H-můstku) a dvou I/O(OUT1 a OUT2) pro vyžádané průběhy signálů. Nevýhody jsou ve spotřebě energie(dva mikropočítače), ve složitosti programových obsluh sběrnic, vyššímu počtu signálových drah, které musí být z hlediska EMC ošetřeny paralelním zemnícím vodičem, a tím ve zvýšení celkové velikosti plošného spoje. To se spolu s vyšším počtem součástek nepříznivě odráží na celkové ceně a složitosti systému.



Obr. 3.1b: Multiprocesorová architektura

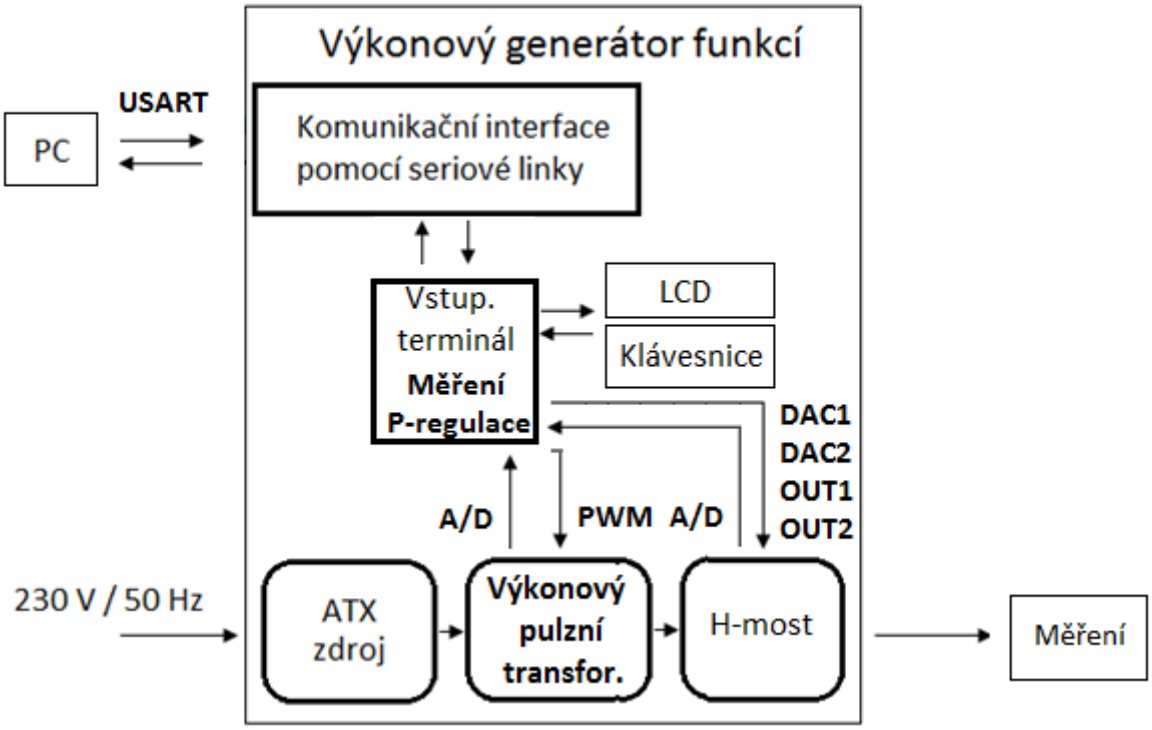
Na Obr.3.1b vidíme podobnou multiprocesorovou architekturu. Zde je však rozdíl v počtu signálových vodičů (DAC,OUT1-4), které přispívají k tvarování žádaného průběhu. Z hlediska návrhu se jedná o variantu, která potřebuje více součástek (dva tranzistory, které přivádí signál buď k levému nebo pravému hornímu tranzistoru H-můstku) a více řídicích signálů. Navíc oba předchozí modely obsahují zmíněné sběrnice, které vyžadují dva a více vodičů(v případě *I²C* vodiče SDA pro data, SCL pro hodinový signál a v případě například SPI jsou to vodiče MISO, MOSI,SCLK a výběrové vodiče SS1..SSn), ke kterým je potřeba připojit "pull up" rezistory kvůli držení definované napěťové úrovně. Tím nám roste velikost plošného spoje a cena. Není-li z podstaty projektu nutné, aby byl systém zajištěn multiprocesorovým systémem, například kvůli zmíněné paralelizaci úkolů, nebo kvůli

výpočetnímu výkonu, pak zcela jistě postačí architektura s jedním řídícím obvodem, tj. s jedním procesorem.



Obr. 3.1c: Multiprocesorová architektura se sběrnicovým měřením

Pro úplnost na obr.3.1c je také multiprocesorová architektura, kde řídicí úlohu, ve smyslu generování signálů a obsluhy H-můstku má "Vstupní terminál", ale údaje o naměřených veličinách jsou poskytovány druhým procesorem. Toto je vůbec nejhorší řešení, protože dochází k velkému zpoždění při sběrnicové komunikaci.



Obr. 3.1d: Architektura s jedním řídicím procesorem

Závěrem poznamenejme, že blok "Měření" v pravém dolním rohu je externí měřící zařízení pro kontrolu činnosti, zatím co Měření v bloku "Vstup. terminál, Měření, P-regulace" je měření pomocí A/D převodníku v procesoru pro kontrolu výstupních parametrů.

Jako procesor byl vybrán PIC16F1777, díky svým implementovaným perifériím, jako jsou digitálně analogové převodníky, PWM generátory a analogově digitální převodníky. Procesor je taktován frekvencí 64Mhz.

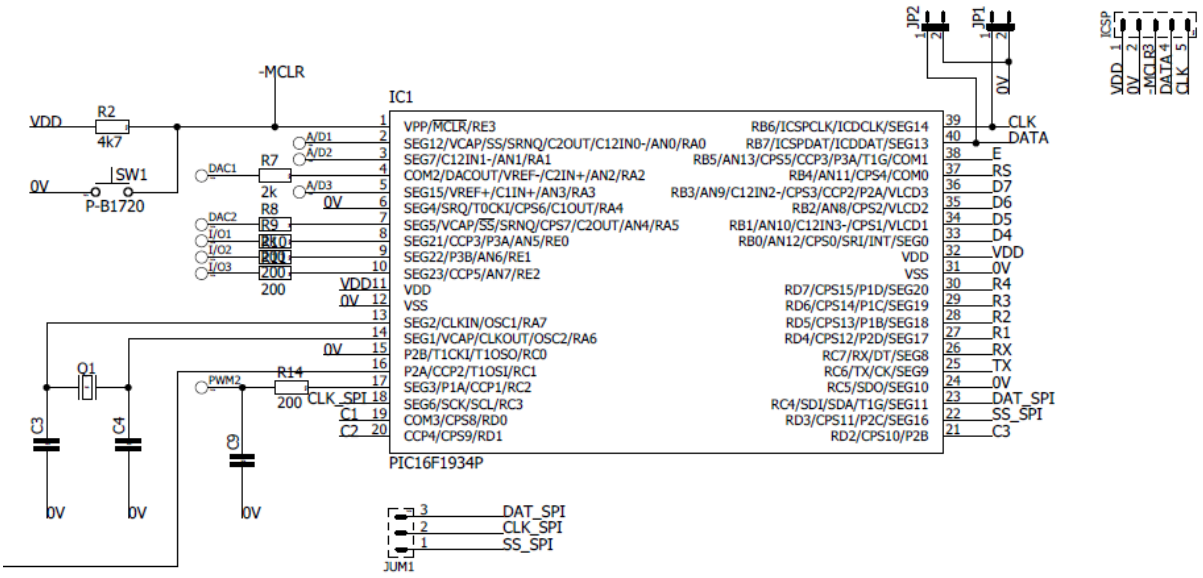
3.2 Obsluha

Jak bylo popsáno výše, systém může uživatel kontrolovat buď zadáním hodnot do maticové klávesnice, ne poslat data přes univerzální sériovou linku USART. Data jsou následující. Typ signálu, kmitočet, amplituda. Také ofset signálu je možné zadat, je možné zadat, ale nesmí být překročena hodnota maximálního napětí, které generátor může poskytnout. Maximální ofset by byl v podstatě stejnosměrný signál teoreticky hodnoty 100V. Chceme-li generování přerušit nebo nastavit znova, jednoduše zmáčkneme reset. Parametry vybraného signálu se zobrazí na LCD displeji. Systém se zapne spínačem na zadní straně zdroje ATX. Jsou-li parametry mimo rozsah, systém to napíše na LCD displej. Protože se však jedná o parametry, které by byli složité na výpočty, jsou mezní hodnoty stanoveny empiricky. Například zadám-li napětí 100V, ale zdroj poskytne pro jakoukoliv zátěž pouze 90V, pak bude těchto 90V hranice, za kterou systém už nemůže jít. Pomocí auto kalibrační metody,

kdy by systém sám zadával parametry, a na základě měření by je porovnával zda jich dosahuje, nebo ne. Toto je nápad pro budoucí vylepšení, avšak teď není implementován.

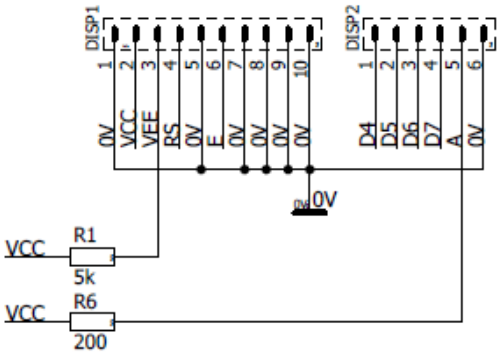
3.3 Řídicí deska

Jak bylo řečeno ve druhé kapitole, VGF je řízen procesorem PIC16F1777. Ten lze pomocí ICSP[7] Zadávání dat je řešeno pomocí maticové klávesnice 4x3(čtyři řádky, tři sloupce). Zároveň je k procesoru připojen resetovací spínač, kterým uživatel resetuje program, a vyvolá menu pro zadání parametrů signálu. Jak bylo zmíněno výše, nepoužité piny procesoru jsou ošetřeny připojením na zem. Procesor má možnost komunikace s PC pomocí RS-232. Interface je realizován obvodem MAX3232, který je možné napájet napětím 3,3V. Na desku jsou ze zdroje ATX přivedená dvě napětí, a to 3,3V a 5V. Dále je zde invertor, jehož funkci popíšeme v kapitole 4. Mikroprocesor, invertor a LCD displej bude napájen napětím 5V. Procesor a MAX3232 napětím 3,3V. Deska plošného spoje byla vytvořena za pomocí návrhového softwaru Eagle 8.4.2.

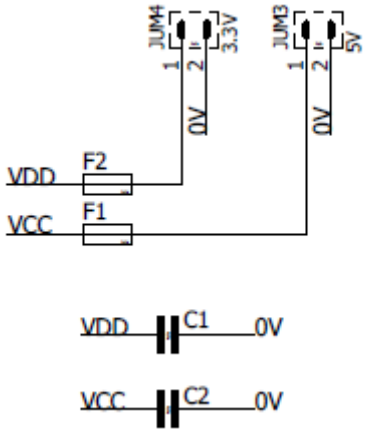


Obr. 3.2: Procesor a jeho porty

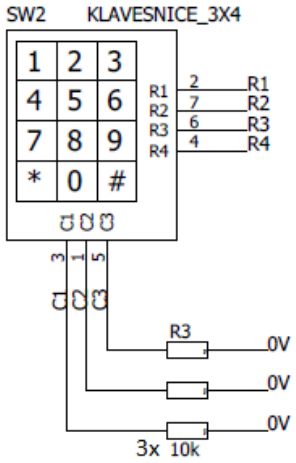
[7] ICSP- In Circuit Serial Programing, druh programování procesoru, který je zapojen v aplikaci



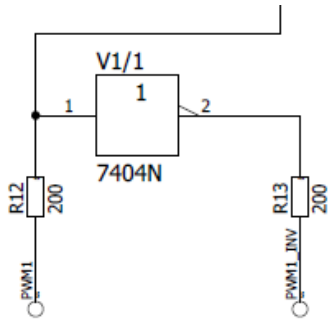
Obr. 3.3: Připojení LCD



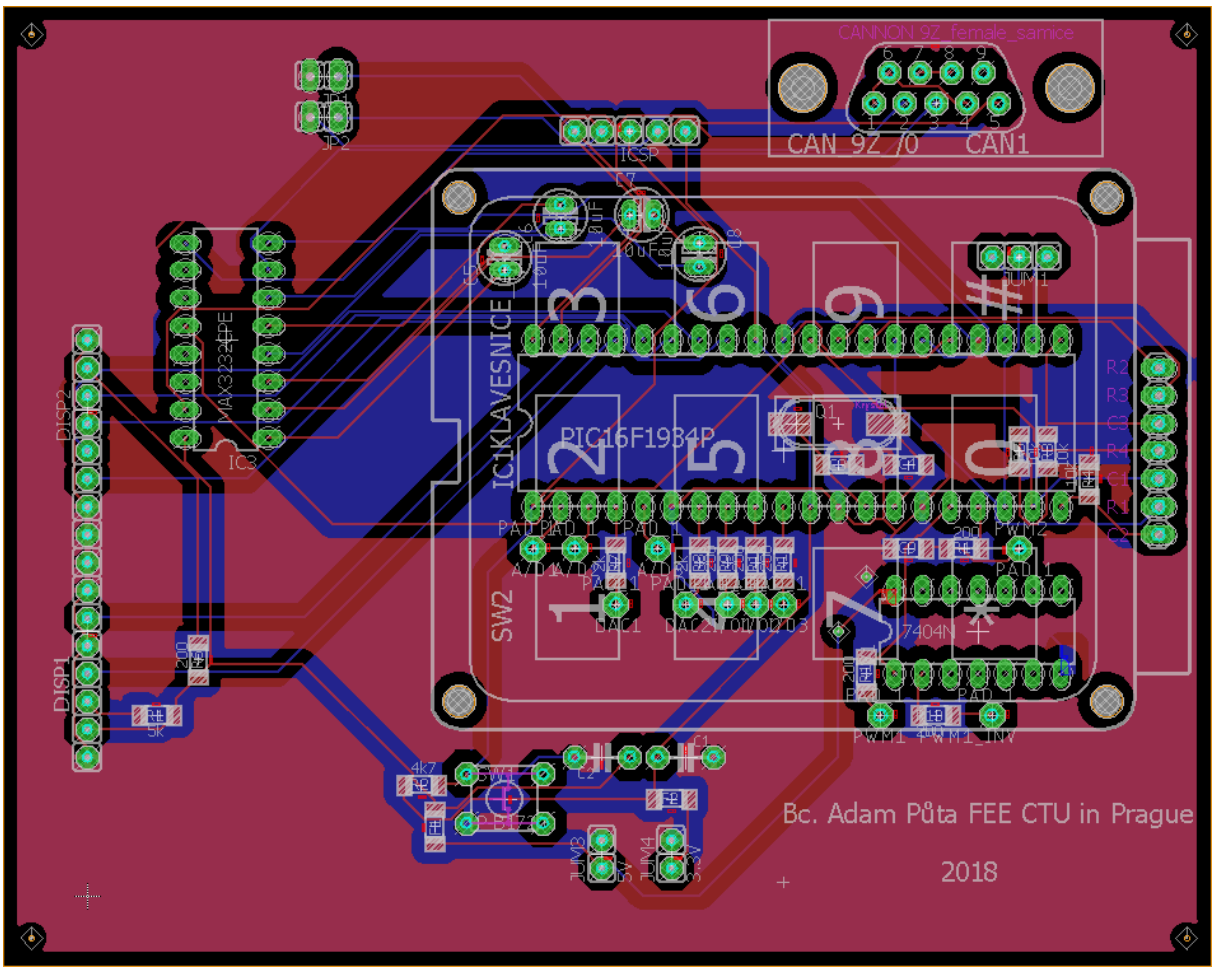
Obr. 3.4: Napájecí vstupy



Obr. 3.5: Připojení klávesnice



Obr. 3.6: Invertor PWM signálu



Obr. 3.7: Plošný spoj řídicí desky

Problém při návrhu desky je omezená velikost motivu, který je v prostředí Eagle demo relativně malý. Dále u invertoru PWM signálu, tj. u obvodu 74HCT04 jsou v knihovně napájecí piny jiným

názvem, než je název pro napájení. To má za následek, že napájecí dráha nejde napojit. Toto jsem vyřešil pokrytím měděné plochy přes napájecí dráhu a pin integrovaného obvodu, a tím jsem je spojil.

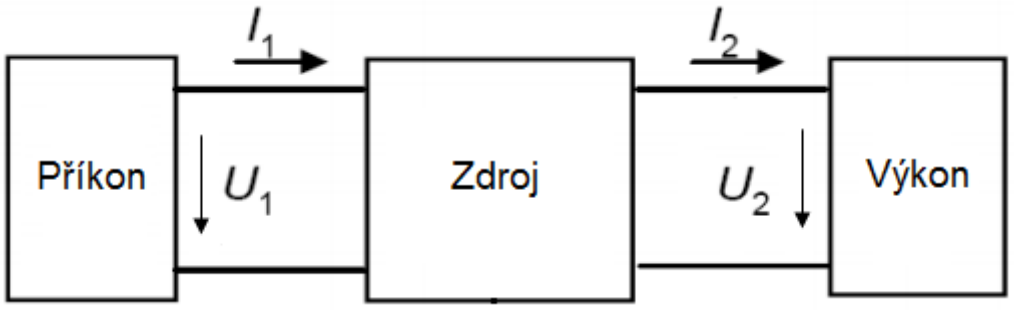
Velký problém byl s řadičem LCD displeje, který nekomunikoval s knihovnou v prostředí microC. Pro tuto knihovnu je potřeba řadič HD44780. Rozlitá vrstva mědi je spojená s GND signálem (reprezentovaným 0V). To je s ohledem na zlepšení EMC. Deska tak nebude tolik vyzařovat do okolí, a to hlavně co se PWM a dalších rychle se měnících signálů týká. Pro RS-232 je implementován klasický CAN9 konektor. Nakonec poznamenejme, že je vyvedena SPI sběrnice pro komunikaci s případnými okolními periferiemi.

Kapitola 4
Zdroj pro VGF

Na trhu se můžeme setkat se širokou škálou zařízení, která slouží jako zdroj výkonu. Rozlišujeme je podle několika hledisek:

- Principu činnosti jednotlivých složek zdroje
- Druh zdroje
- Architektura zdroje
- Dodávaného výkonu
- EMC[8]
- Účiník
- Tepelné vyzařování, účinnost

Princip činnosti popisuje algoritmus, pomocí kterého dochází k transformaci elektrických veličin. Tím tedy definuje i jednotlivé hardwarové prvky systému. Například sem spadá to, jestli jepro převod napětí a proudu použit transformátor, zda se konkrétní zdroj chová jako snižující, nebo zvyšující spínaný měnič, který využívá spínání indukčnosti pomocí tranzistoru, nebo to, jestli je použit například princip spínaných kapacit, zda se jedná o lineární zdroje na bázi stabilizátorů atd..



Obr. 4.1: Elementární princip zdroje

Poznamenejme, že u ideálního zdroje dochází k transformaci proudů a napětí, přičemž vstupní a výstupní výkony se rovnají. To však ve fyzikální realitě není možné, uplatňují se tepelné ztráty jednotlivých prvků.

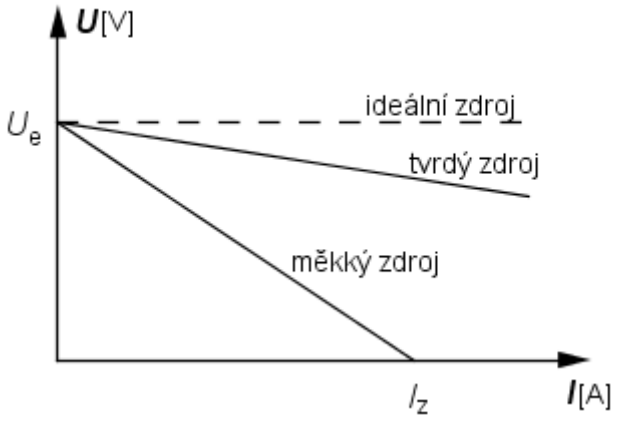
Z hlediska dodávaného výkonu nás zajímá, jakou spotřebu má koncové zařízení, které chceme napájet. Zajímají nás tedy nominální hodnoty výstupního napětí a proudu našeho zdroje, a tím jeho nominální výkon.

[8] EMC- Electromagnetic compatibility – Elektromagnetická kompatibilita

Ten je dán následujícím vztahem:

$$P_2 = U_2 . I_{2max} \quad [W, V, A] \tag{4.1}$$

kde P_2 je výstupní výkon, U_2 je výstupní napětí a I_{2max} je maximální výstupní proud zdroje.



Obr. 4.2: Zatěžovací charakteristika zdroje

Vlivem vnitřního odporu zdroje je nemožné dostat nekonečně veliký výstupní proud při konstantním výstupním napětí. Sklon přímky zatěžovací charakteristiky určuje tzv. tvrdost zdroje. Je-li zapojená zpětná vazba kvůli regulaci napětí, může být charakteristika zpočátku konstantní, a poté při mezním proudovém odběru prudce klesnout.

EMC je velice důležitá hlavně v průmyslu a dopravě, ale i například v armádní sféře. Je velmi důležité, aby napájecí zdroj nerušil svým vyzařováním okolní zařízení, a tím nepřispíval k narušení celkové stability a robustnosti systému, v němž je implementován. Poznamenejme však, že problematika spojená s EMC je velmi složitá, a vyžaduje si nemalé znalosti z teorie elektromagnetického pole. EMC je rovněž řešena v rámci norem(na státní úrovni, dříve ČSN, nebo normy EU).

Efektivní hodnota výkonu je hodnota, která se rovná odvedené práci, jako vykoná příslušný stejnosměrný výkon. Není-li řečeno jinak, označuje se efektivní hodnota například napětí velkým písmenem bez indexu. Pro korektnost ještě dodejme, že jakákoliv efektivní hodnota se dá spočítat z amplitudy průběhu. Následuje příklad přepočtu efektivního výkonu, a jeho amplitudy.

$$P_{ef} = \frac{P_{Max}}{\sqrt{2}} \ [W, W] \tag{4.2}$$

kde P_{ef} je efektivní činný výkon a P_{Max} je amplituda výkonu.

4.1 Vstupní část

Do vstupní části zahrnujeme usměrňovač (dále USM) a filtr (zpravidla RC, RLC), který má za úkol odfiltrovat ze vstupního napětí a proudu nedokonalosti vzniklé usměrněním. Někdy se můžeme setkat také se snižovacím, nebo oddělovacím transformátorem[9], který celý systém galvanicky oddělí od zbytku sítě 230V[10]. Podle potřeby pak můžeme napětí na výstupu vstupní části měnit jak je potřeba. To se zpravidla dělá pomocí pulzního transformátoru.

Pro naše potřeby bylo potřeba vyrobit takový zdroj, který by pokryl požadovaný maximální výkon. Hlavně tedy zmíněný transformátor, který by naše vstupní napětí převáděl na požadované hodnoty. Zároveň bylo potřeba sekundárního zdroje pro napájení řídicí elektroniky. To se však po neshodách s dodavatelem nepodařilo, a proto jsem zvolil jako zdroj do výkonové generátoru PC zdroj typu ATX[1]. Konkrétně se jedná o zdroj **Fortron Hydra 850W**[2]. Na trhu jsou sice výkonnější zdroje, ale jsou podstatně dražší. PC zdroje obsahují výše uvedené části a kromě toho také ochrany proti zkratu, přetížení nebo přepětí. ATX zdroj dodává tyto napětí:

PSU napěťové hladiny						
+12V	-12V	+5V	-5V	+3,3V	-3,3V	+5VSB

Tab. 4.1: Výstupní napětí zdroje ATX

Testováním bylo dokázáno, že malé zvlnění výstupních napětí nemá vliv na funkci procesoru, a proto není nutné stabilizovat výstup například obvody 7805.

4.1.1 Výkonová bilance

Jak jsme již zmínili, maximální proud je 10 A a napětí 100 V, měli-by jsme počítat s výkonem 1kW. K tomu se také musí započítat spotřeba řídicí systému(mikrokontrolér, LCD atd.). U mikrokontroléru to bývá maximálně kolem 800mW při plném zatížení. Protože ale máme k dispozici ATX zdroj o výkonu 850 W, měli bychom být schopni dostat na základě vztahu (3.1)získat buď 100 V při 8,5 A, nebo 10 A při 85 V. Tyto hodnoty by měli být k dispozici po transformaci pomocí pulzního transformátoru v režimu "step-up" měniče, jehož vstupem je právě zdroj ATX. Musíme však počítat s tím, že pulzní transformátor vykazuje určité ztráty. Je také spotřeba řídicí desky a to znamená, že by hodnoty měly být o něco nižší. Jako vstup do pulzního transformátoru je ze zdroje ATX přivedeno napětí 12 V, z čehož vyplívá, že při deklarovaných 850 W by mělo téct zhruba 70 A přes primární cívku. Protože jako pulzní transformátor používám transformátor z jiného vyřazeného zdroje, nejsou k dispozici parametry o výkonu, počtech závitů atd.. Je proto potřeba zařízení otestovat a zjistit jak moc se od

[9] V anglosaské literatuře bývá tento subsystém (transformátor, USM, a filtr) označován zkratkou TR - Transformer Rectifier

[10] Efektivní hodnota napětí, síťová frekvence 50 Hz

zadaných parametrů výkonu signálu odchylujeme. Po té je například možné vyměnit transformátor za výkonnější, nebo přidat další, tak aby pracoval paralelně k prvnímu. Testování výkonu pulzního transformátoru probíhá pomocí H-můstku se změnou plnění PWM, který budí primární vinutí transformátoru. Jako zátěž je proměnný odpor v podobě tranzistoru NMOS. Ten je ovládán potenciometrem, který reguluje napětí 12V. Tak tranzistor otevíráme, a jsme schopni měřit napětí nebo proud.

4.1.2 Optimalizace výkonu

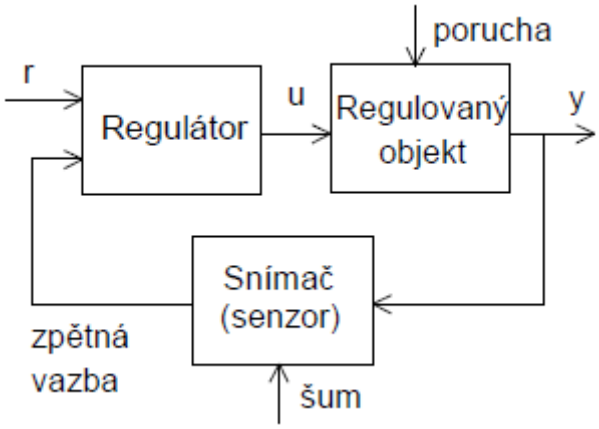
Samozřejmě lze odběr řízení do jisté míry regulovat vhodným návrhem softwaru. Experimentoval jsem také s možností odpojení napájení mikrokontroléru v době, kdy je na jeho výstupu logická nula. Konkrétně při jednoduchém ON/OFF signálu. Nicméně výsledky simulace a měření nedopadly přesvědčivě. Spíše naopak to přineslo nepatrně větší odběr, který si vysvětluji jako impulzní odběr. Další problém byl v tom, že bude-li mikrokontrolér vykonávat složitější operace, nebo jen například inkrementovat nějakou hodnotu čítače, pak po odpojení a opětovném připojení napájení dojde vlivem resetu nastavení programového čítače na hodnotu nula. Program tak jede od začátku. Jistým řešením by mohlo být uložení současné adresy programového čítače do nevolatilní paměti (energeticky nezávislé), jako je například FLASH paměť. Celkově by to ale spíše celý chod programu zpomalovalo.

Další možností je napájení řídicích obvodů co možná nejmenším napětím, které je pro daný čip povoleno.

4.1.3 Řízení pulzního transformátoru

Řízení transformátoru může být provedeno několika způsoby. Plný H-můstek, poloviční můstek, push-pull, nebo s jedním spínacím prvkem. Algoritmus nastavení hodnoty napětí je následující:

- plné otevření dvou diagonálních tranzistorů H-můstku, kvůli co nejmenšímu úbytku napětí
- zvyšování PWM1 aktivní doby a měření napětí -> P-regulace
- po dosažení požadovaného napětí začne procesor generovat pomocí DAC signál



Obr. 4.3: Zpětnovazební regulace[3]

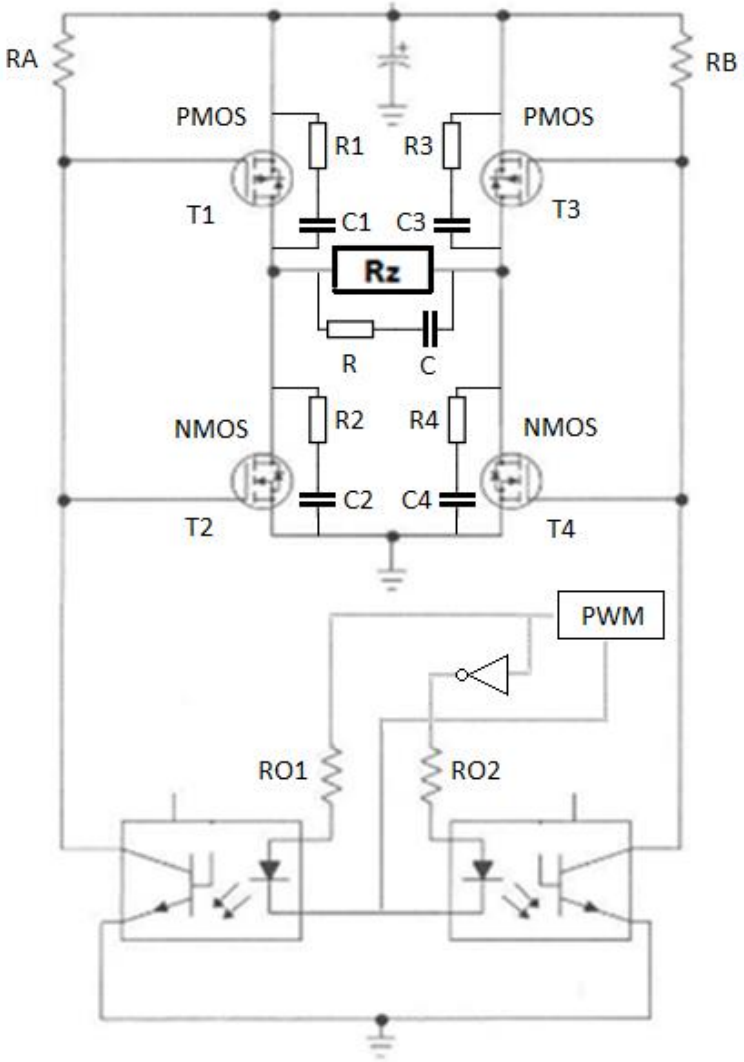
Z hlediska zvýšení EMC kvality bude řízení provedeno pomocí H-můstku ovládaným PWM signálem s unipolárními tranzistory NMOS(nejedná se o H-můstek pro tvarování signálu). Vstup do můstku bude 12 V, které přivedeme ze zdroje ATX. Jak bylo řečeno výše, zdroj ATX dodá při napětí 12 V a výkonu 850 W zhruba 70 A. Musíme tedy dimenzovat můstek na tento proud. Ve hře je několik variant. První varianta je použít klasický H-můstek v podobě VNH3SP30-E, který je dimenzován na 30 A. Z toho ale vyplívá že bychom museli mít alespoň tři takové obvody, aby jsme mezi ně proud rozdělili na přijatelnou mez. To je z hlediska ceny nepřijatelné, protože jeden obvod stojí kolem 350 Kč. Půjdeme na to pomocí diskrétních součástek. Princip na Obr. 3.3 ilustruje zapojení s tranzistory NMOS a PMOS. Na každé straně je optočlen, pomocí kterého řídíme najednou PMOS tranzistor nahoře a NMOS tranzistor pod ním. Řídicím signálem bude PWM. Pokud je signál PWM ve stavu logická jedna, horní tranzistor je rozepnut a dolní sepnut. Podobně na druhém optočlenu. Výhoda je že každý sloupec tranzistorů (NMOS, a pod ním PMOS) je možné řídit pouze jedním optočlenem, je-li na optočlen přivedená logická jednička, je dolní NMOS rozepnut a horní PMOS sepnut. Při použití jednoho PWM signálu potřebujeme invertor, který by ze signál PWM dělal dva komplementární signály, čili k sobě opačné, a tím pádem otevíral pouze dva protilehlé tranzistory. Problém je že při prohazování stavů může nastat situace, kdy budou na malou chvíli sepnuty oba dva tranzistory pod sebou, a tím pádem dojde ke zkratu. To vychází z podstaty, že jeden je NMOS a druhý PMOS. Je sice pravdou, že PWM signál je velmi rychlý, přeběhy mezi úrovněmi jsou v řádech ns, tím pádem tento stav trvá velmi krátkou dobu, a tak bychom mohli s výhodou použít pouze jeden PWM signál. Problém u tohoto řešení je, že rapidně roste spotřeba proudu, a právě použitím dvou PWM signálů pro řízení by se tento problém nevyřešil. Dva PWM signály bychom použili například pro "dead time kontrolu" tzn. pokud by jsme měli nad sebou dva NMOS nebo dva PMOS tranzistory ve sloupci, pak bychom fázovým posuvem PWM signálů zamezili tomu, aby se tranzistory sepnuli najednou. Určitou změnou odporů RA a RB můžeme spotřebu zmenšit, ale tím nám rapidně klesne proud zátěží. Toto řešení tedy zavrhuji a použiji místo toho klasický H-můstek pouze s NMOS tranzistory s jedním řídicími signálem PWM s invertorem. Co se invertoru týká, lze ho nahradit hradlem NAND, jehož vstupy jsou spojeny do jednoho. Co se týče doby přeběhu mezi logickými úrovněmi, s tím není problém. Jde o desítky nanosekund. Důležitá je kompatibilita z hlediska napěťový úrovní. Výstupy procesoru jsou typu CMOS. Obecně obvody TTL nejsou kvůli rozdílným napěťovým úrovním kompatibilní s CMOS obvody. 0V až 0,8V je u TTL logická nula na vstupu. Od napětí 2V je to logická

jednička. Pásmo mezi reprezentuje zakázaný pás neurčitosti. U CMOS obvodů je na vstupu log. jednička při napětí od 0,7V a log. nula do napětí 0,3V[4]. Použijeme proto CMOS invertor 74HCT04.

74HCT04	
Log. 0 [V]	Log.1 [V]
0-0,8	>2

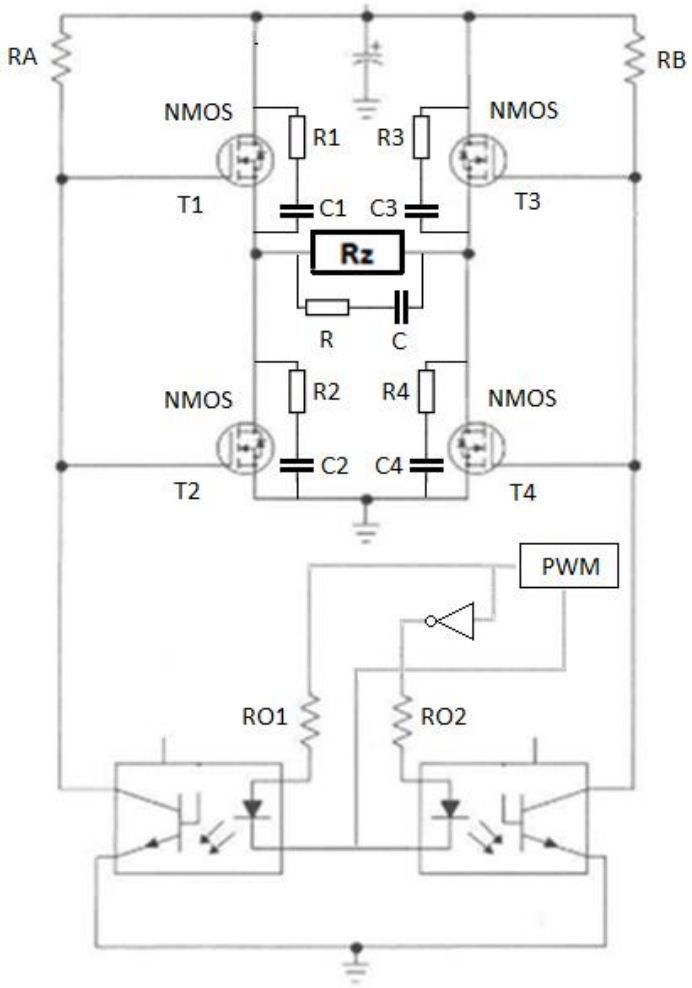
Tab. 4.2: Napěťové úrovně invertoru 74HTC04

Z hlediska EMC je lepší mít vf[11]. signálů co nejméně. Obě varianty byli simulovány programem Proteus viz. přílohy a zkušebně změřeny.



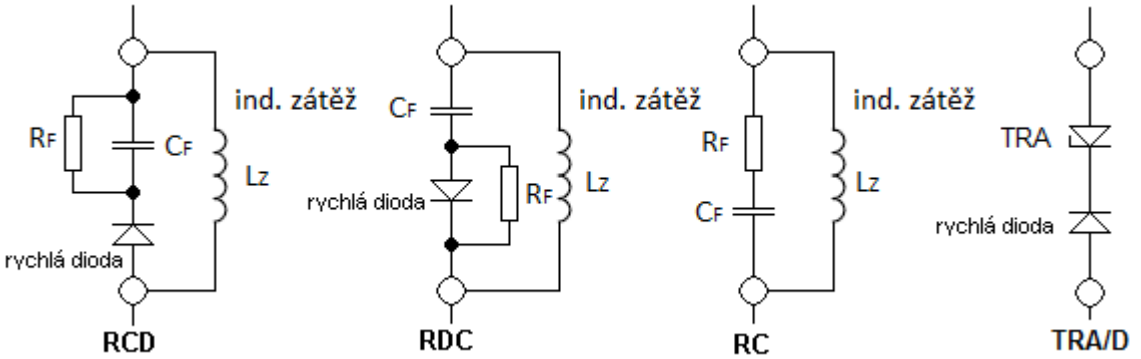
Obr. 4.4: Řízení primárního vinutí transformátoru pomocí H-můstku s komplementárními tranzistory

[11] vf - vysoko frekvenční

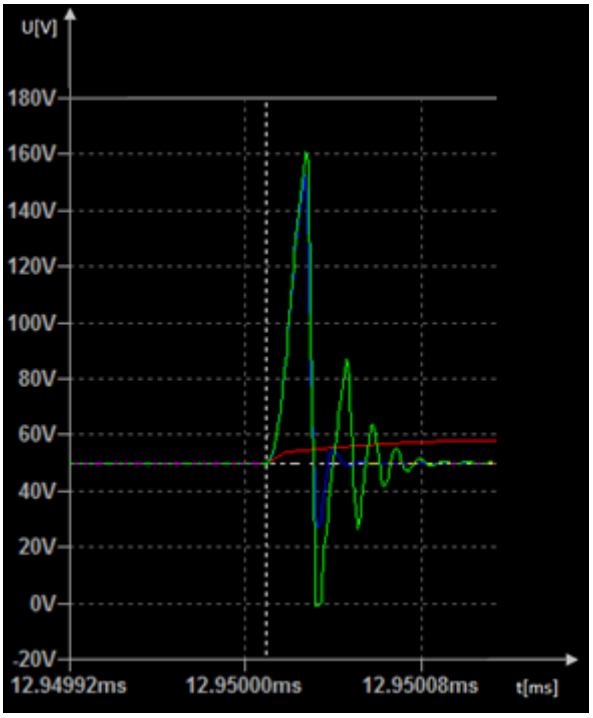


Obr 4.5: H-můstek s NMOS tranzistory

Rezistory R, R1-R4 a kapacity C, C1-C4 tvoří tzv. snubber, nebo-li tlumič napěťových špiček, které vznikají spínáním tranzistorů T1-T4. Na obrázku 4.6 jsou další možné způsoby zapojení ochran proti přepěťovým špičkám.



Obr. 4.6: Zapojení RCD a TRA/D ochrany, a další možnosti odstranění přepěťových špiček



Obr. 4.7: Průběh utlumení napěťové špičky

Vstupní rezistory RO1 a RO2 byly voleny podle maximálního proudu pro výstupní piny procesoru, které jsou 25mA. LED vysílač v optočlenu můžeme zatížit až 50mA. To znamená 200 Ohmů při signálové úrovni 3,3 V. Horní dva rezistory RA a RB, které tvoří zesilovač spolu s tranzistorem v optočlenu, byly nastaveny na hodnotu 1kΩ. Těmito odpory se dá také snížit zmíněný nadproud vzniklý při konverzi PWM z úrovně logická jedna na logickou nulu, ale zároveň mají vliv na to, jak moc budou tranzistory T1 až T4 otvírány. Optočlen je řady TLP281-4 SO16 TOSHIBA. NMOS tranzistory jsou typu IRL2203N. To s ohledem na předpokládané proudové zatížení kolem 70 A. Co se měření a ochran na primární straně impulzního transformátoru týká, není nutné je řešit. Zdroj ATX sám o sobě obsahuje ochrany proti zkratu a přepětí, což je jeho výhoda. Na závěr poznamenejme, že na Obr.3.3 je primární vinutí pulzního transformátoru zobrazeno jako blok Rz.

Za transformátorem následuje rychlá usměrňovací dioda BYV42E-200, která je dimenzovaná na 30 A. To proto, že při návrhu se dimenzuje dioda na maximální výstupní proud $I_{out} = I_F$. To je sice více než požadovaných 10A, ale je to pojistka do budoucna pro navýšení proudového odběru silnějším vstupním zdrojem. Dále se dioda dimenzuje na dvojnásobek špičkového napětí v případě jednocestného usměrňovače[5][6]. Díky vysoké frekvenci spínání tedy nebude potřeba Graetzova můstku, ale postačí obyčejný jednocestný usměrňovač. Vyhlazovací kapacita byla volena tak, aby činitel zvlnění byl 0,1%. Pro frekvenci 50Hz by byla kapacita podle vztahu (3.3) obrovská.

$$C = \frac{kI_{výst}}{pU_{ss}} \quad [\mu F, mA, \%, V] \tag{4.3}$$

kde *C* je kapacita vyhlazovacího kondenzátoru, *k* je konstanta[12], $I_{výst}$ je výstupní proud záťěže, *p* je činitel zvlnění[13] a U_{ss} je výstupní stejnosměrné napětí.

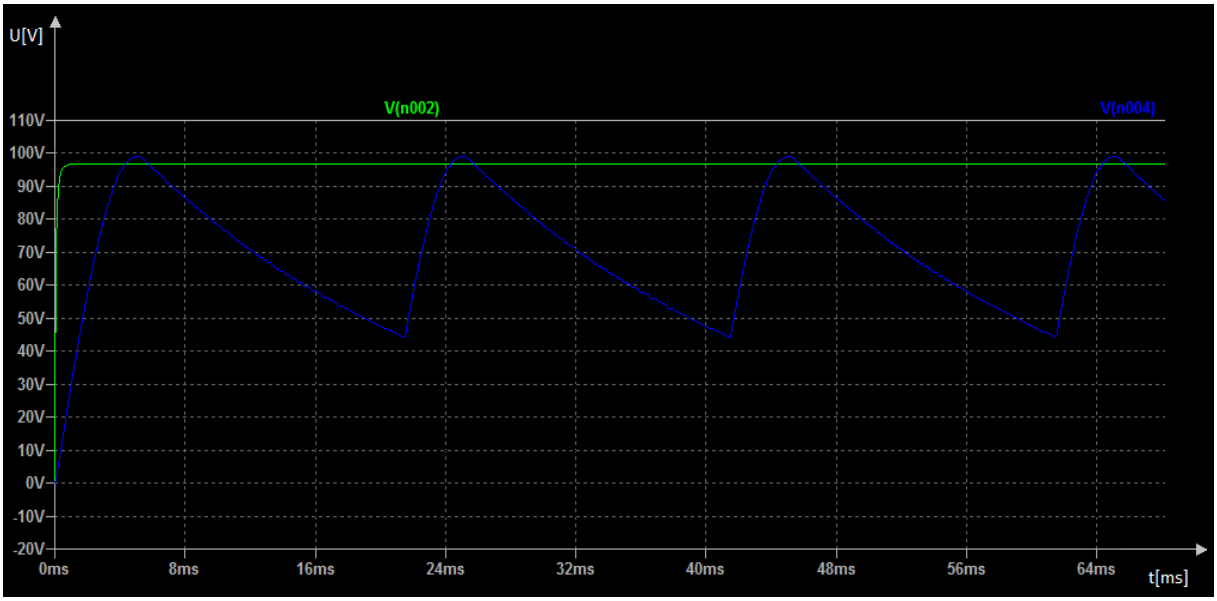
Museli bychom tedy buď zvětšit činitele zvlnění, což je v našem případě nežádoucí, nebo použít další typy filtrů, jako například LC filtry[7]. Protože ale usměrňujeme napětí o frekvenci zhruba 85kHz, což je frekvence PWM signálu, který řídí pulzní transformátor, bude hodnota kapacity C pro daný činitel zvlnění 2000uF. Tento předpoklad vychází ze simulace programem Lt Spice. Zelený průběh je pro frekvenci 85kHz, modrý pak pro 50Hz.

Pro RC filtr: $$\varphi = \omega RC = 2\pi f RC \ \ [\text{-}, rad^{-1}, \Omega, F] \qquad (4.4)$$

Pro LC filtr: $$\varphi = \omega^2 LC = (2\pi f)^2 LC \ \ [\text{-}, rad^{-1}, H, F] \qquad (4.5)$$

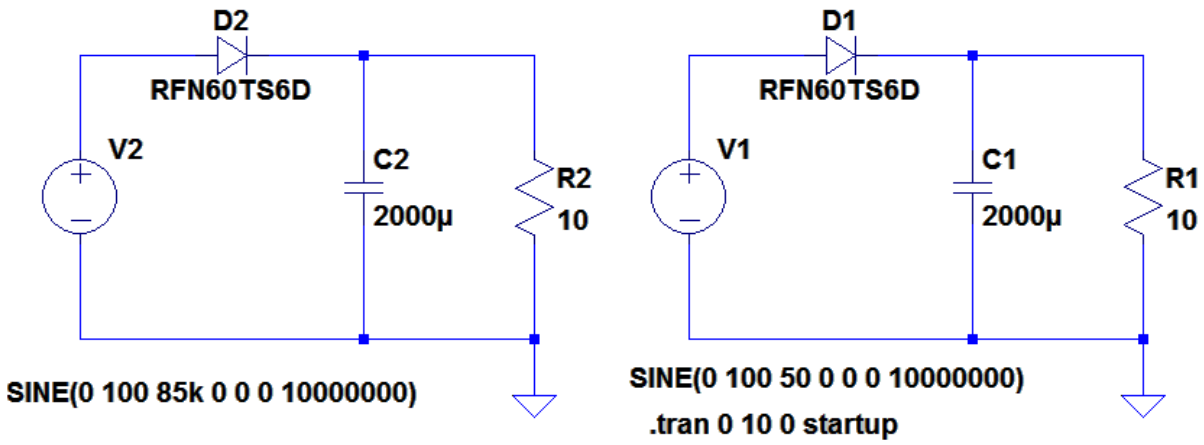
kde φ je činitel filtrace, ω je úhlová rychlost, R, L a C je odpor, kapacita a indukčnost.

Vztahy (4.4) a (4.5) dokládají kmitočtovou závislost filtru.



Obr. 4.8a: Simulace vlivu frekvence spínání na průběh usměrňovaného napětí

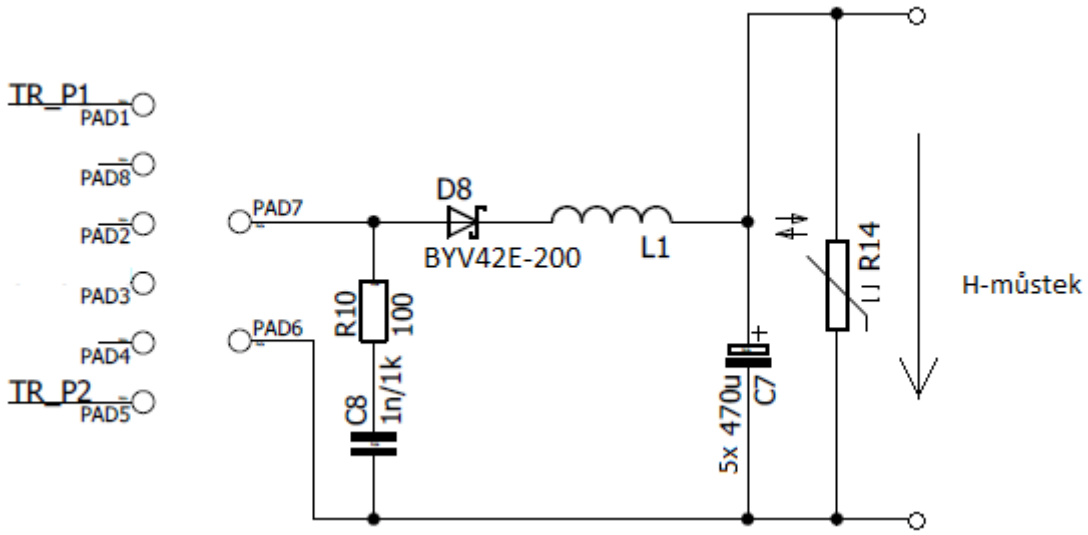
[12] Pro jednocestný USM 600, pro dvoucestný 300.
[13] Dokonalé zvlnění = 0%, fyzikálně nerealizovatelné.



Obr. 4.8b: Simulovaný obvod

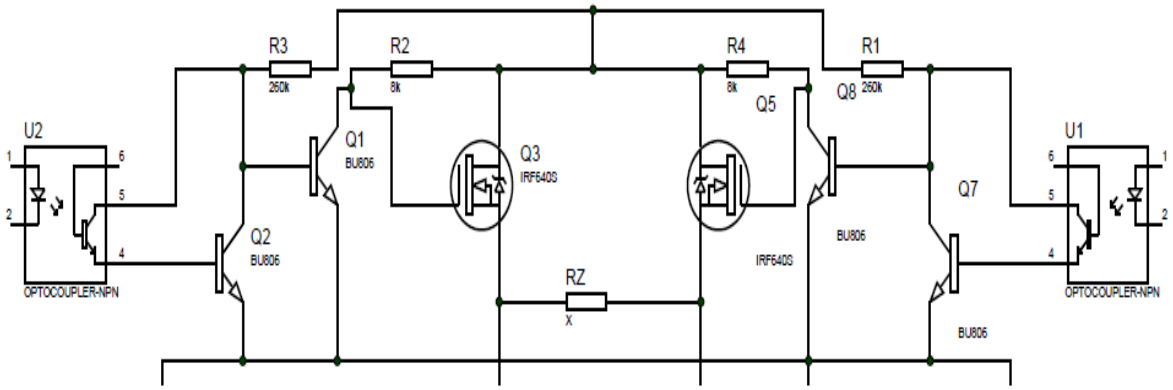
Za filtrovacím kondenzátorem bude tlumivka, která tak vytvoří spolu s kapacitou LC filtr. Tlumivka je stejně jako pulzní transformátor odejmuta z jiného ATX zdroje. Její hodnota není známa a proto se přesným výpočtem LC filtru nebudeme zabývat.

Na sekundární straně bude implementována trubičková pojistka F 16A 5x20 na 16 A jmenovitého proudu. Jako přepěťová ochrana je použit varistor VCR10D151KAR. Jeho jmenovité napětí je kolem 165V. Pokud vlivem přepětí sníží varistor svůj odpor, následný proud vyvolá na odporu R16 úbytek napětí, který změří mikropočítač generátorové části, a dojde k vypnutí systému. Přepětí se může objevit například nesprávným nastavením softwaru ovládající vf. transformátor. To se může stát při prvním spuštění, kdy se systém testuje. Detekce přepětí by měla fungovat tak ,že jakmile varistorem začne protékat proud, napětí mezi jeho svorkami by mělo rapidně klesnout. Pokud nebude v souladu pokles napětí s PWM signálem(například bude PWM odpojeno vlivem resetu uživatele pro nastavení nových hodnot), který transformátor ovládá, bude toto detekováno subsystémem pro měření napětí jako porucha. Poté dojde k odpojení PWM a systém se zablokuje do doby, než dojde k resetu uživatele. Obvod řízení transformátoru bude spolu s obvodem H-můstku realizován na univerzálním plošném spoji z důvodu jednak ušetření místa, a jednak proto, že verze návrhového programu Eagle je ve volné licenci omezená co do velikosti obrazce plošného spoje.



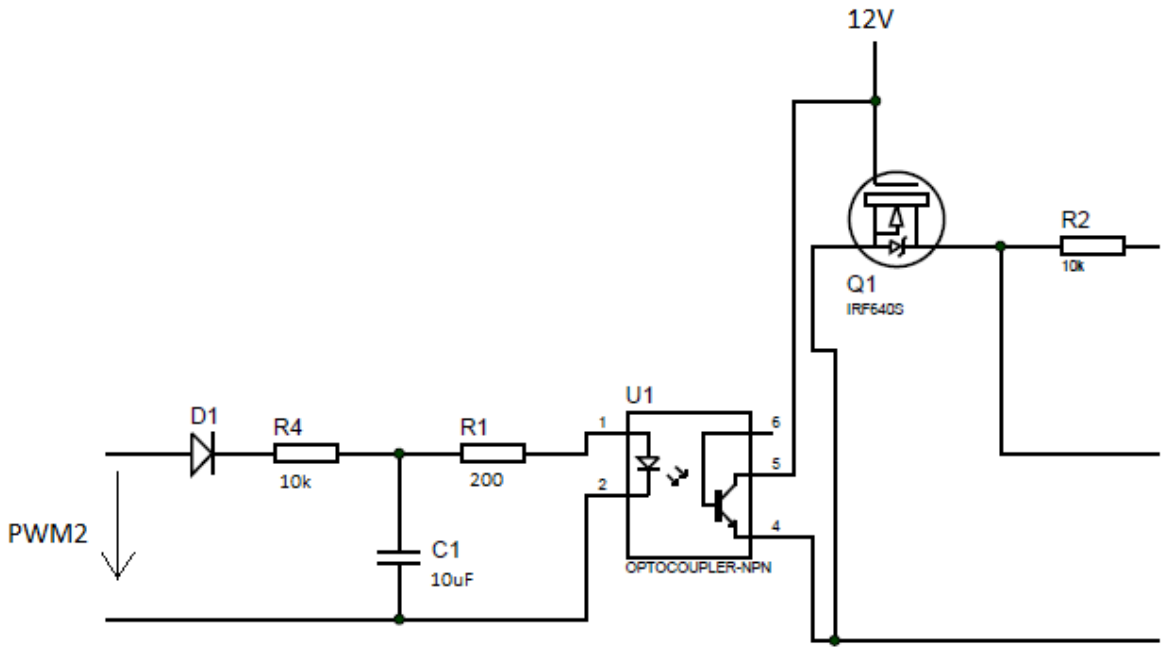
Obr. 4.9: Sekundární strana pulzního transformátoru

Kapitola 5
H-můstek pro modulaci výkonu

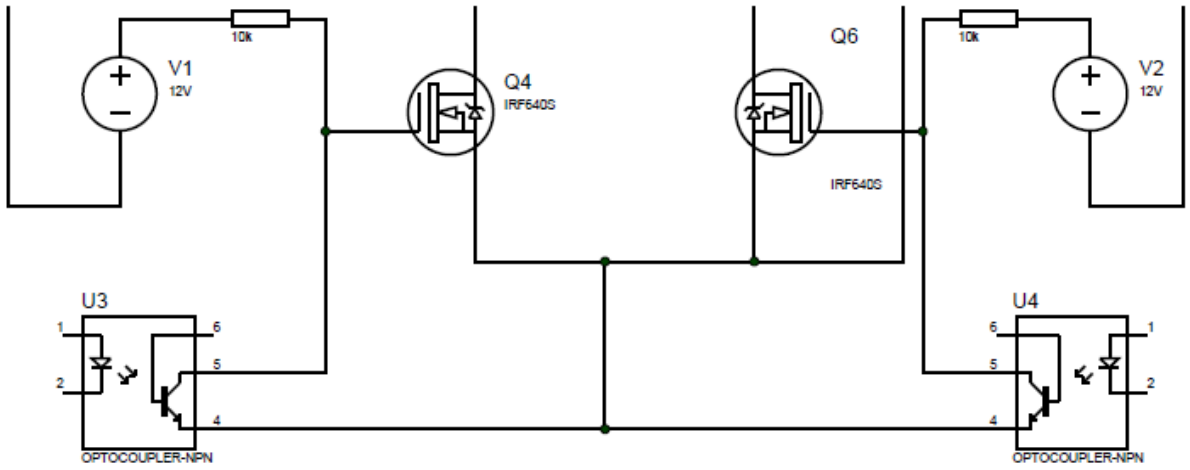


Obr. 5.1: Horní polovina můstku

Horní polovina můstku je ovládána pomocí dvou zesilovačů třídy A, které tvoří tranzistory Q1,Q2,Q7 a Q8. Jedná se o typ BU406. Typ byl vybrán s ohledem na maximální napětí mezi drainem a sourcem, a také díky vysoké frekvenci spínání. Problém je že, že při použití jednoho zesilovače dochází k invertování výstupního signálu. Použitím druhého zesilovače signál opět překlopíme zpátky. To je příjemnější z hlediska psaní programu, ale z hlediska návrhu obvodu složitější. Při použití pouze jednoho zesilovače pro ovládání NMOSu by jsme si museli dávat pozor na to, že ta strana, která má být vypnutá, musí být ve stavu log. 1. Šlo by tedy o řízení negativní logikou. V případě, že bychom na optočlen, který je připojen na stranu, která má být vypnutá přivedli log. 0, došlo by ke zkratu. To, jak velkou amplitudu signálu jsme schopni dostat záleží jednak na hodnotě v DAC, jednak na rezistorech R1-R4, a také na hodnotě odporu, který vede z procesoru na vysílací LED optočlenu. Pokud je tento odpor velký, snížíme amplitudu signálu, pokud je moc malý, optočlen se dostane do saturovaného stavu, zmenšíme tím rozlišení signálu. Navíc bychom kvůli malé hodnotě tohoto odporu mohli překročit hranici dovoleného proudu pinem procesoru. Hodnota byla stanovena na 910Ω. Jak bylo zmíněno v kapitole 2, podkapitola 2.2 Digitální generování signálu, pokud měníme, respektive zvyšujeme frekvenci výstupního signálu, dochází k utlumování jeho amplitudy. Musíme tedy měnit jeden z odporů R1-R4. Vyberme tedy odpory R1 a R3. Tyto odpory nahradíme NMOS tranzistorem IRF640, stejným jako tranzistory Q3-Q6, které plní funkci H-můstku. Pro jejich řízení použijeme PWM2 signál, který budeme filtrovat RC členem. Tento signál pomocí optočlenu přivedeme na gate tranzistoru. RC člen nám vytvoří analogové napětí. Tímto způsobem by se měl regulovat odpor kanálu tranzistoru NMOS. Odpor se mění jen málo, a proto jsem zkusil zapojit tranzistor jako dělič napětí.



Obr. 5.2: Regulace odporu A zesilovače pomocí NMOSu



Obr. 5.3: Dolní strana H-můstku

Napětí na dolních tranzistorech spínáme pomocí 12V. Není potřeba napětí Ugs příliš zesílit, protože source NMOSu je připojen k zemi. Všechny můstkové NMOS tranzistory jsou připojeny k chladiči ze starého zdroje ATX kvůli výkonovému zatížení.

Kapitola 6
Měření proudu a napětí

Obecně můžeme měřit proud pomocí převodníků proud-napětí s OZ[14], nebo pomocí Hallovy sondy v modulu ACS712[8]. Modul ACS712 má výhodu v tom, že nepotřebuje podpůrné obvody pro galvanické oddělení. Je pouze potřeba počítat s proudovým rozsahem který je pouze od -5A do 5 A. Samozřejmě to není problém, protože můžeme spočítáme bočník ampérmetru, který lze vypočítat jako:

$$R_B = \frac{I_m \cdot R_m}{I - I_m} \; [\Omega,\Omega,A,A] \tag{6.1}$$

kde R_B je odpor bočníku, I je celkový proud, I_m je proud procházející ampérmetrem a R_m je vnitřní odpor ampérmetru (v našem případě Hallovy sondy ACS 712).

Vnitřní odpor byl změřen multimetrem a má hodnotu 0,3Ω. Z toho vyplívá hodnota bočníku jako 300mΩ. Výstup ze sondy je analogový signál s rozlišením 180mV/A. Při nulovém odběru je signál 2,5V. K této hodnotě se podle směru proudu buďto přičítá, nebo odečítá zmíněné rozlišení vynásobené velikostí proudu. Pro 10 bitový A/D převodník je vztah mezi měřeným napětím a hodnotou uloženou v paměti hodnota daná vztahem (6.2)[9].

$$X_n = \frac{V_{meas} \cdot nbit}{V_{ref}} \; [-,V,bit,V,A] \tag{6.2}$$

kde X_n hodnota v registru po převodu A/D převodníkem, V_{meas} je změřené napětí, nbit je rozlišení A/D převodníku a V_{ref} je referenční napětí.

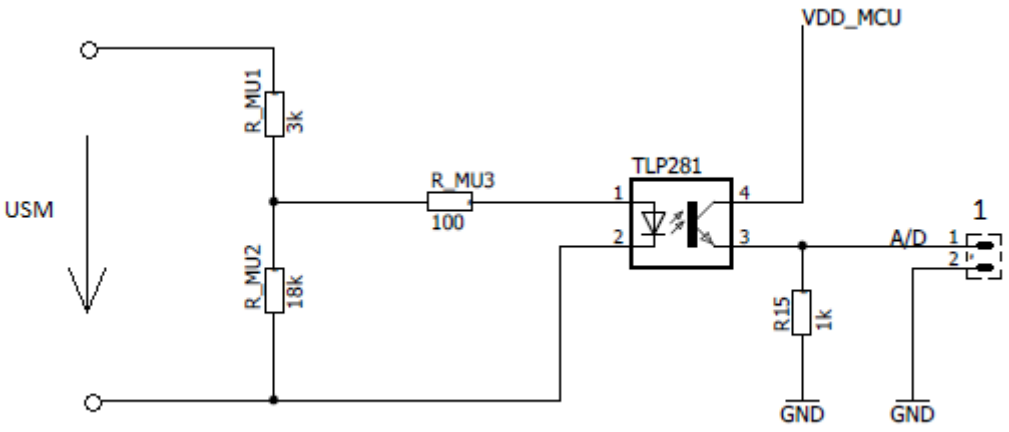
Obr. 6.1: ACS712

Další možností, pokud se nám nechce počítat bočník, ale velice nákladnou možností je zapojit dva ACS 712 paralelně. Proudy se jednoduše rozdělí. Poslední možností je použít trochu profesionálnější přístup a zakoupit jeden modul ACS 713, který je na cca. 10 A. Proud můžeme měřit kdekoli mezi usměrňovačem a můstkem nebo přímo v můstku u zátěže. To je dané vlastností H-můstku.

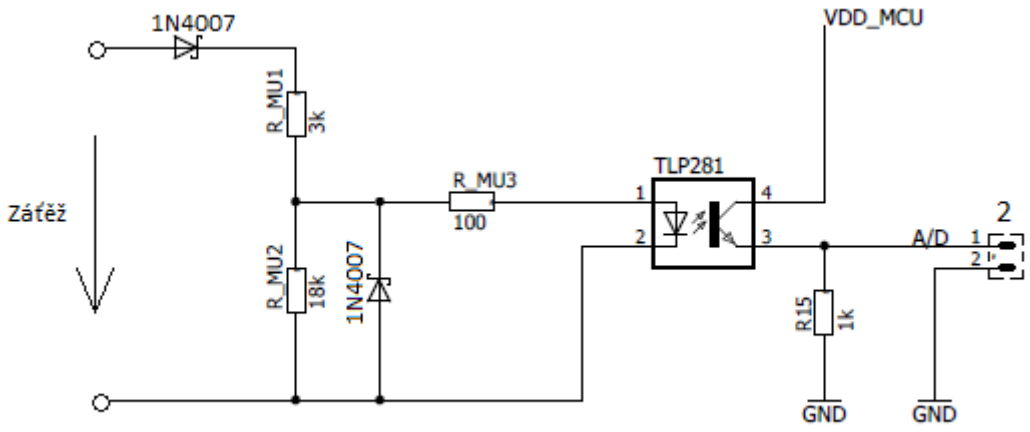
[14] OZ- operační zesilovač

S měřením proudu je však problém. Za předpokladu, že máme stejnosměrný průběh nebo například obdélníkové impulzy, není problém napětí na výstupu ACS712 jednoduše změřit pomocí A/D převodníku. Horší situace nastane, máme-li periodické, neharmonické signály. Mezi takové signály patří například pila, trojúhelník, ale i harmonický sinus. Problém je, že na rozdíl od stejnosměrného konstantního signálu potřebujeme navzorkovat celou periodu, hodnoty uložili do pole hodnot a nakonec vybrali tu největší, tedy amplitudu. Tu můžeme použít pro výpočet efektivní hodnoty, ale navíc by jsme museli odfiltrovat fluktuace signálu. Další problém by byl se synchronizací měření a dobou kdy je signál skutečně v maximu. Při vzorkování celé periody také narušíme generovaný kmitočet signálu, protože ho generujeme cyklickým nastavováním DACu, a to znamenalo, že po každé nové hodnotě poslané na DAC musíme provést čtení na A/D převodníku. To sebou přináší také problémy v podobě požadavků na výpočetní výkon a rychlost procesoru, protože kromě výpočetního zatížení procesoru, musíme také kompenzovat vliv útlumu signálu při různých frekvencích. Poněkud jednodušší metoda by byla, pokud známe periodu signálu 1/f, kde f je zadaná frekvence signálu. Za určitý čas, tedy zhruba na úrovni jedné periody, by jsme provedli měření, a tím získali amplitudu signálu. Museli by jsme se ale opravdu trefit do času, kdy je signál v maximu. To ovšem platí pro periodické signály. V případě uživatelem zadaný náhodný průběh by takto měřen být nemohl, protože neznáme periodu signálu.

Dále je zde měření napětí. Na rozdíl od měření proudu nás napětí zajímá pro zpětnou vazbu. Měřicí obvod je velice jednoduchý. V podstatě pouze o zatížený dělič napětí a galvanické oddělení v podobě optočlenu. Jeho hodnoty byly zjištěny ze simulačního programu proteus pří požadavku, aby na výstupu z optočlenu bylo napětí od 0V do 3,2V. To kvůli referenci 3,3V, která je zároveň napětím V_{DD} pro procesor. Při měření na zátěži, kde dochází k časové změně napětí, a je proto potřeba ho



Obr. 6.2: Měřeni napětí za usměrňovačem



Obr. 6.3: Měření napětí na zátěž

Obě měření napětí provádíme opět za pomoci optočlenu kvůli ochraně procesoru.

Pokud je nižší, zvýšíme PWM plnění, pokud nižší, snížíme PWM plnění. U měření napětí stejně jako u proudu by jsme ho mohli vykreslovat na grafický LCD displej a vytvořit si tak osciloskop. Díky znalosti napětí a proudu můžeme vypočítat odpor zátěže. Potřebujeme k tomu ale stejnosměrné napětí a proud, nebo efektivní hodnoty. Pak jednoduše podle Ohmova zákona spočteme odpor zátěže.

$$R = \frac{U}{I} [\Omega,V,A] \qquad (6.3)$$

Měření se provádí po několika periodách signálu. Ne během každé periody z důvodu přeťěžování procesoru.

V případě například induktivní zátěže by jsme při harmonickém průběhu signálu mohli zjistit, fázový posuv mezi napětím a proudem. Stačilo by porovnat hodnotu napětí v nule s hodnotou proudu. Pak, pokud je napětí rovno 0V, a zároveň proud není 0A, pak z toho lze usuzovat kapacitní, nebo Tím by jsme teoreticky vypočítali impedanci zátěže, a v určitém omezeném rozsahu automaticky provádět impedanční přizpůsobení připojováním vhodných součástek. To je pouze teoretický nápad na budoucí vylepšení.

7 Závěr

Bohužel před převedením práce jsem nedopatřením spálil řídicí desku. A proto jí musím do konání státní závěrečné zkoušky předělat. Výsledky simulací obvodů však napovídají, že parametrů bylo zhruba dosaženo. Maximální napětí na zátěži v H-můstku vlivem ztrát na tranzistorech a útlumu při spínání je okolo 70V. Tvarování signálu jak je vidět v kapitole 2 relativně funguje. Samozřejmě dochází vlivem útlumu k určité deformaci na H-můstku, nicméně softwarově jsme schopni generovat pomocí DAC čisté průběhy. Do budoucna bych zvolil vhodnější typ procesoru. Dále bych se zamyslel na stylem programování, respektive tím jak zlepšit rychlost mezi vybavením různých po sobě jdoucích vzorků signálu. Jak víme, požívání standardních knihoven je trochu problematické, protože vývojář dopředu neví, jak moc zpomalí procesor. Z pulzního transformátoru se podařilo dostat napětí lehce pod 100V.

Jeden z kroků bylo vytvořit grafickou aplikaci pro ovládání výstupního napětí z PC. K tomuto účelu byl použit software Microsoft Visual Studio 2015 (dále MVS). Aplikace je psaná v jazyce VC#. Do aplikace uživatel jednoduše zadá požadované vstupní napětí, nebo pro přímé ladění může zadat přímo PWM střídu a frekvenci spínání. Aplikace se ovládá jednoduchým zapsáním hodnot do textového pole. Tlačítkem Start jsou přes komunikační rozhraní USART(RS232) odeslány parametry do procesoru. Aplikace sama najde komunikační porty PC, nabídne výběr. Při práci na předchozích procesorech komunikace fungovala, ale s procesorem PIC16F1777 jsou problémy v komunikaci. Grafická podoba viz. přílohy.

Protože zařízení není certifikované pro průmyslové nebo domácí aplikace, slouží výhradně k účelům této diplomové práce. Autor práce nenese zodpovědnost za případné zranění nebo škody na majetku při manipulaci osobami, kterým to ze strany autora nebylo povoleno.

Literatura

[1] Ronešová ZČU zdroje ATX
http://home.zcu.cz/~ronesova/index.php?menuitem=ec200x

[2] Dokumentace zdroje ATX
https://www02.cp-static.com/objects/pdf/1/10a/1349906573_1_power-supply-units-fsp-fortron-hydro-g-650w-ppa6502804.pdf

[3] Teorie řízení Jiří Skalický VUT Brno

[4] Tabulka napěťových hodnot TTL CMOS
https://cs.wikibooks.org/wiki/Praktická_elektronika/Logické_obvody

[5] Návrh napájecích zdrojů: Husák, Jirásek FEL ČVUT mikroelektronika

[6] Koncepce napájecích zdrojů Jan Novák FEL ČVUT mikroelektronika

[7] Filtrace zvlněného napětí
https://coptkm.cz/portal/reposit.php?action=0&id=21819&instance=2

[8] ACS712
https://dspace5.zcu.cz/bitstream/11025/10062/1/DP_Svejda.pdf

[9] Přepočet DAC hodnoty na napětí
https://learn.sparkfun.com/tutorials/analog-to-digital-conversion

Přílohy

Příloha A.1

Zkratky

VGF	výkonový generátor funkcí
PWM	pulzně šířková modulace
ASK	amplitudová modulace
PSK	fázová modulace
OZ	operační zesilovač
USART	univerzální sériová sběrnice
TTL	tranzistorově tranzistorová logika
CMOS	komplementární MOS
Sin	funkce sinus
LED	Světelná dioda

Příloha A.2

Seznam symbolů

P_2	výstupní výkon (W)
P_1	vstupní výkon (W)
I_1	vstupní proud (A)
I_2	výstupní proud (A)
η	účinnost (-)
R	elektrický odpor (Ω)
t	čas (t)

φ	účiník
ϕ	činitel filtrace
R_B	odpor bočníku
R_m	odpor ampérmetru (Ω)
ϕ	magnetický tok (Wb)
N	počet závitů
M	vzájemná indukčnost (H)
C	kapacita (F)
s(t)	signál v časové oblasti
L	indukčnost (H)
V_{ref}	referenční napětí (V)
V_{meas}	měřené napětí (V)

Příloha A.3

Kódy generování signálů v C

-----Inicializace periférii------

```
unsigned short kp, cnt, oldstate = 0;
char txt[6];

// Keypad module connections
char  keypadPort at PORTD;
// End Keypad module connections

// LCD module connections
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections

void main() {
  cnt = 0;                                // counter 0
  Keypad_Init();                          // Keypad
  ANSELE = 0;                             // I/O pins settings
  ANSELA = 1;                             // analog pins settings
  Lcd_Init();                             // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);                    // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);               // Cursor off
  Lcd_Out(1, 1, "Menu  :");
  PWM1_Init(10000);                       //10kHz
unsigned short current_duty, old_duty, current_duty1, old_duty1;

   double array[180];

   int i;

    int x=0;

    int freq;

    int refresh_period=2;

    int amp=32;

 Settings();
```

-----Sinus funkce, hodnoty vypočtené programem v C++-----

```
VREFCON2 = 0/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 35/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 70/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 105/amp;
for(freq = 0;freq<refresh_period;freq++){}
VREFCON2 = 139/amp;
...
```

-----Signál trojúhelník------

```
int y = 2;
for(int i=0;i<100;i++){                    //for loop for number of repeat of period
while(y>0){
if(DAC1CON0 <=30&&x==0){
  DAC1CON0++;
  if(DAC1CON0==30){
  x = 1;
  y --;                                    //y periode detection variable
  }
  }
  else{
   DAC1CON0--;
   if(DAC1CON0==0){
   x=0;
   y --;
```

---obdélníkový signál---

```
for(int i =0;i>delay;i++){
  DAC1CON0 = amp;                    //amp -> signal ampllitude
}
```

---obdélník lepší----

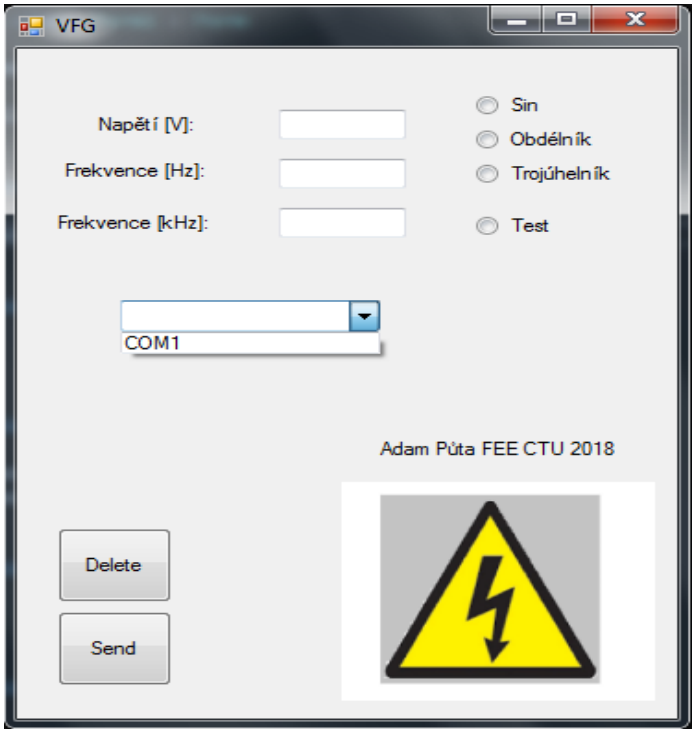
```
DAC1CON0 = amp;                    //amp -> signal ampllitude
for(int i =0;i>delay;i++){          //just hold DAC value and do nothing
}
DAC1CON0 = 0;                    //amp  = 0
```

----Visual C# Nalezení COM portu-----

```
private void Form1_Load(object sender, EventArgs e)
        {
            getAvaiblePorts();
        }


void getAvaiblePorts()
        {
            String[] ports = SerialPort.GetPortNames();
            comboBox1.Items.AddRange(ports);

            }
```

Příloha A.4

Ukázka programu pro PC



Grafická podoba aplikace pro řízení pomocí PC

Příloha A.5

Fotodokumentace z výroby

Řízení pulsního transformátoru

Řídicí deska

Řídicí deska a řízení můstku

Příloha CD

1) Testovací program generátoru

2) Výpočtový program pro funkci sinus

3) Simulační soubory proteus

4) Kontrolní deska Eagle

5) Datasheety