Středoškolská technika 2013

Setkání a prezentace prací středoškolských studentů na ČVUT

PWM regulátor s ATmega8

Pavel Anděl

SPŠ A VOŠ PÍSEK

Karla Čapka 402, 397 11 Písek

Konzultant: Mgr. Milan Janoušek

ANOTACE

Tento dokument popisuje vývoj mého projektu, PWM regulátoru s ATmega8-16PU. Dokument obsahuje informace o vývoji a tvorbě schémat, plošných spojů, programu pro MCU a mnoho dalších informací. Mimo jiné je přímo doplněn zmiňovanými schématy a návrhy plošných spojů. V neposlední řadě také dokument pojednává o konstrukčních řešeních tohoto výrobku.

PODĚKOVÁNÍ

Tímto bych rád velice poděkoval všem lidem, kteří mi s projektem pomáhali. Konkrétně to jsou tito lidé:

Mgr. Milan Janoušek
Poděkování za vedení mé práce a konzultace ohledně problematiky.

Bc. Josef Pajer
Poděkování za umožnění výroby plošných spojů ve školní laboratoři.

Ing. Josef Kubeš
Poděkování za poskytnuté informace o PWM v hodině ELT.

Ondřej Krejčí
Poděkování za zapůjčení stanicové mikropájky.

PROHLÁŠENÍ

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Písku dne …… podpis: ……

OBSAH

1) ÚVOD

1.1 Proč jsem si vybral právě tento projekt

Jakožto letecký modelář již řadu let používám ve své dílně odporovou pilu. Toto zařízení je mi velkým pomocníkem při opracovávání materiálů, jako je např.: polystyrén, expandovaný polypropylen (EPP) a materiál zvaný „depron“, ze kterých stavím své modely. Princip odporové pily je založen na napnuté odporové struně, která je zahřívána na vysokou teplotu procházejícím elektrickým proudem. Tento proud je však nutno regulovat, neboť každý opracovávaný materiál má pro dokonalý řez svou ideální teplotu. Pokud je procházející proud malý, struna je příliš studená a materiál lze opracovávat velice těžce a pomalu. Pokud je však procházející proud velký, struna je příliš horká a řez materiálem je sice rychlý, ale nepřesný. Proto jsem se rozhodl vyrobit regulační člen, který by dokázal plynule nastavit velikost proudu, protékajícího odporovou strunou.

1.2 Možnosti regulace

Ještě před tím, než jsem se stal studentem SPŠ a VOŠ Písek, neměl jsem dostatečné znalosti pro řešení tohoto úkolu. Neuměl jsem tedy sestavit žádný složitější obvod regulátoru. Proto jsem se rozhodl pro nejjednodušší cestu, kterou bylo připojení proměnného, výkonového odporu do série s odporovou strunou. K tomuto účelu jsem vytvořil jednoduchý, výkonový potenciometr, složený z dřevěné latě, dvou vrutů do dřeva a zhruba metrového kusu odporové struny. Vruty jsem zašrouboval, zhruba do ½ jejich délky, z jedné strany latě na její konce a napnul mezi ně odporovou strunu. Jako jezdec potenciometru jsem použil „krokosvorku“. Tento regulátor fungoval dobře a používal jsem ho několik let. Používání tohoto regulátoru však bylo značně neekonomické, neboť na odporové struně vznikal velký ztrátový výkon, přeměňovaný na teplo. Dalším záporem toho regulátoru byly jeho velké rozměry.

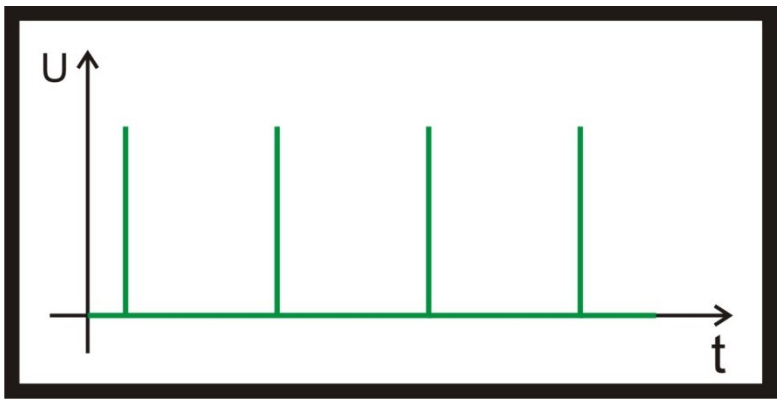
Po několika letech, během kterých jsem ve škole čerpal informace a zkušenosti, jsem se rozhodl, že regulátor zmodernizuji. Jako první regulátor, který jsem se rozhodl vyrobit, byl obyčejný tyristorový regulátor. Schéma tohoto regulátoru jsem nalezl v jednom starém modelářském časopise a velice se mi líbilo. Bylo totiž velice jednoduché a složené jen z několika málo součástek. Z popisu činnosti tohoto regulátoru jsem se však dozvěděl, že tento regulátor dokáže pracovat pouze se střídavým nebo pulsujícím napětím. Mým požadavkem ale bylo, abych mohl regulátor napájet stejnosměrným napětím z autobaterie, nebo upraveného PC zdroje. Tímto zjištěním jsem výrobu tyristorového regulátoru zavrhnul a nadále používal regulaci za pomoci výkonového potenciometru.

Když jsme později ve škole probírali funkci PWM regulátorů, vnuklo mi to nápad na PWM regulátor pro mou odporovou pilu. Inspiroval jsem se na internetu a postavil na

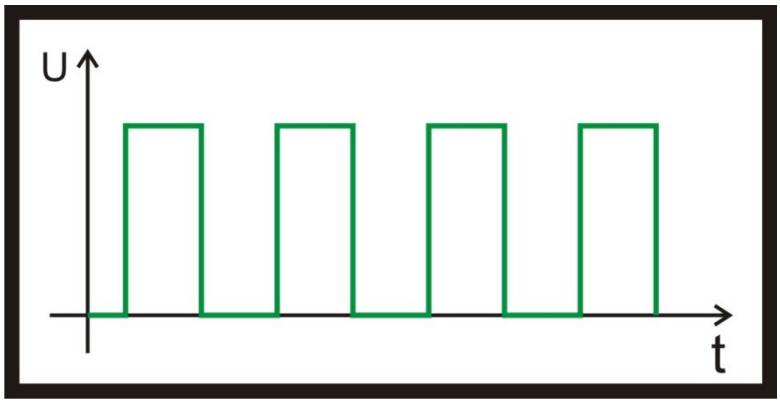
kontaktním, nepájivém poli zkušební obvod. Ten byl opět velice jednoduchý, ovšem plně funkční. Základem tohoto obvodu byl časovač NE555, který byl zapojen jako generátor obdélníkových impulsů, s možností změny střídy potenciometrem. Tyto impulsy byly následně zesilovány výkonovým, bipolárním tranzistorem. Tento regulátor fungoval velice dobře, byl ekonomičtější a splňoval můj požadavek na SS napájení. Od tohoto regulátoru se odvíjí i tento projekt, který je ovšem mnohem propracovanější.

1.3 Popis mého výrobku

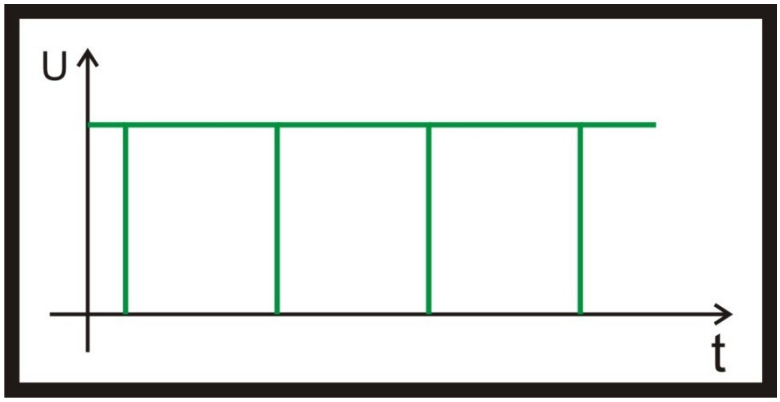
Jak jsem již prozradil, mnou vybraný regulátor využívá k regulaci výkonu pulsně šířkovou modulaci. To znamená, že regulátor generuje na výstupu impulsy, s frekvencí v řádech KHz, velikostí napájecího napětí a s možností změny střídy. Na obrázku **0.1.1** můžeme vidět, jak vypadá průběh výstupních impulsů, pokud nastavíme výkon 0%. Obrázek **0.1.2** zobrazuje průběh při 50% výkonu a obrázek **0.1.3** při 100% výkonu. Při návrhu byl celý regulátor rozdělen na tři plošné spoje. První spoj představuje zdroj, který napájí MCU a koncový spínací obvod. Díky tomu, že je v tomto zdroji zařazen do zapojení i výkonový usměrňovací můstek, můžeme regulátor napájet jak stejnosměrným, tak i střídavým napětím. To také zajišťuje, že obvod nelze připojit ke zdroji opačně (s opačnou polaritou). Druhý plošný spoj představuje řídící centrum celého regulátoru. Na tomto plošném spoji se nachází MCU, programovací konektor, ovládací tlačítka, sedmisegmentový displej, signalizační LED a mnoho dalších potřebných součástek. Třetím, a také posledním, plošným spojem je spínací obvod. Zde dochází k zesílení impulsů, generovaných mikroprocesorem. K tomuto účelu je zde zapojen unipolární tranzistor IRF 530, který dokáže spínat proudy o velikosti až 17A. Byť byl původně obvod regulátoru rozdělen na 3 plošné spoje, během výroby byl plošný spoj s MCU a spoj spínače z konstrukčních důvodů spojen do jednoho. Celý obvod je uložen v krabičce, kterou jsem kvůli atypickým rozměrům výrobku vytvořil svépomocí.



0.1.1



0.1.2



0.1.3

2) POUŽITÝ MCU

2.1 Můj výběr, odůvodnění

Mnou vybraný mikroprocesor pro tuto aplikaci je jednočipový, osmibitový počítač značky Atmel, konkrétně ATmega8-16PU z řady AVR. Tento mikroprocesor jsem si vybral kvůli tomu, že byl pro mě dostupný ve školním skladu součástek. Dalším a velice podstatným důvodem bylo to, že jsem potřeboval mikroprocesor s výstupem PWM, výstupem pro signalizaci (sedmi segmentové zobrazovače) a vstupy pro ovládací prvky (tlačítka).

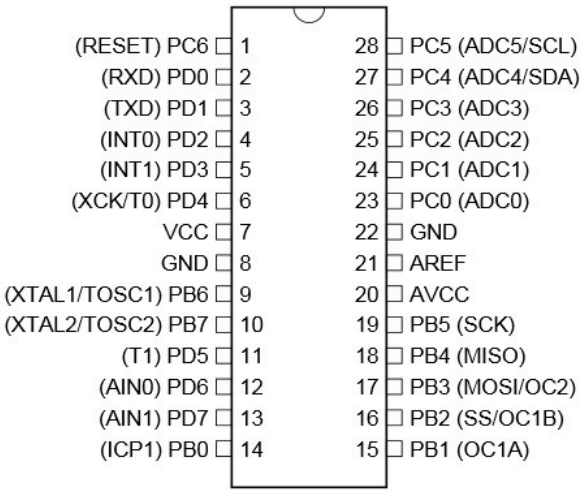
2.2 Základní vlastnosti použitého MCU

Základní vlastnosti mikroprocesoru ATmega8-16PU jsou:

8kB ISP FLASH paměť programu (zhruba 10 000 cyklů)
1kB vnitřní paměť dat RAM
512B paměť EEPROM (zhruba 100 000 cyklů)

23 programovatelných I/O pinů
2x8bit + 1x16bit čítač/časovač
3xPWM výstup
4x10bit A/D převodník
2x8bit A/D převodník
Interní oscilátor (1, 2, 4, 8MHz)
Úzké pouzdro DIL28
Napájecí napětí 4,5-5,5V (nejlépe 5V)

Vývody mikročipu obr. **0.2.1**



0.2.1

3) VÝBĚR SW PRO VÝVOJ APLIKACE

Při vývoji této aplikace jsem použil několik různých softwarových programů. Většina z nich mi velice pomohla, zjednodušila práci a ušetřila čas.

Prvním programem, který jsem při vývoji použil, byl program EAGLE. V tomto programu jsem nakreslil všechna schémata a navrhl plošné spoje. Program EAGLE jsem si vybral z několika zásadních důvodů. Prvním důvodem je, že s tímto programem, díky výuce ve škole, dokážu bez problémů pracovat. Druhým důvodem je fakt, že jiný podobný program na návrh obvodů a tvorbu DPS neznám.

Druhým použitým programem je ATMELSTUDIO. V tomto programu jsem navrhl a napsal program pro MCU, a to v jazyce C.

Dalším programem je ASIX UP, který je nutný pro použití programátoru PRESTO od firmy ASIX. Program je volně šiřitelný a lze jej nalézt buď na CD přiloženém k programátoru, nebo na webu výrobce. K tomuto programu bylo ještě nutné doinstalovat potřebné ovladače z instalačního CD, které zajistí správnou komunikaci PC s programátorem.

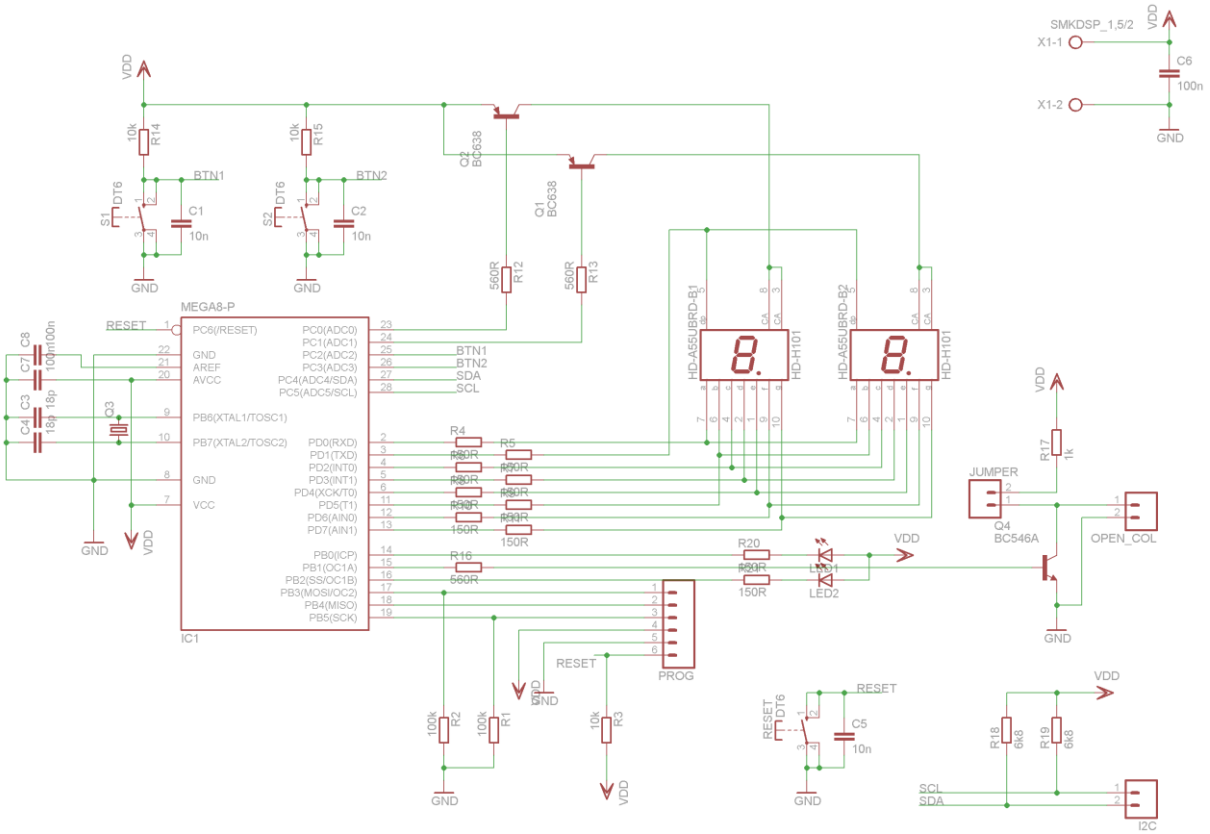
4) POSTUP PRÁCE PŘI VÝVOJI APLIKACE

4.1 Návrh obvodů, kreslení schématu, návrh DPS

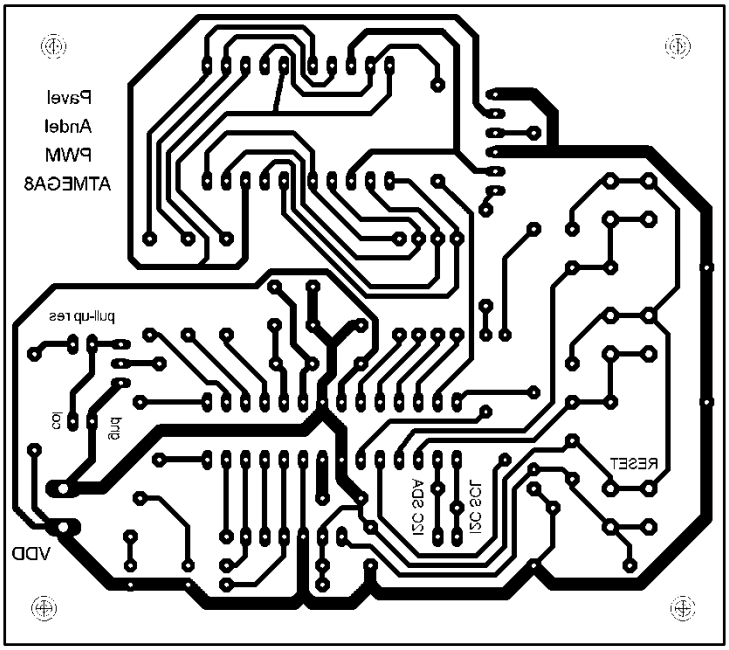
Jako první jsem začal s návrhem řídícího obvodu s MCU (**obr. 0.4.1)**. Hlavní součástkou tohoto obvodu je mikroprocesor ATmega8-16PU, který jsem si vybral z důvodů, viz Kapitola **2.1**. Na vstupy PC2 a PC3 jsou připojena tlačítka S1 a S2. Tato tlačítka jsou zapojena tak, že při rozepnutém stavu je na vstup přiváděna logická jednička v podobě napájecího napětí, zmenšeného o úbytek napětí na 10K rezistorech R14 a R15. Samotná tlačítka jsou navíc opatřena kondenzátory C1 a C2 (10nF), které filtrují kmitání signálu při sepnutí tlačítka. Na vývody PB6 a PB7 je připojen 8.0MHz krystal, který ovšem u tohoto zapojení není nutný. Bohatě by postačil i interní oscilátor mikroprocesoru, ale já chtěl přes to použít tento stabilnější a přesnější oscilátor. Na výstupy PD0-PD7 jsou připojeny dva, sedmi segmentové zobrazovače, se společnou anodou. Ty jsou zapojeny dynamicky, což znamená, že jsou jeden a druhý střídavě rozsvěcovány s takovou frekvencí, aby pro lidské oko (díky jeho světelné setrvačnosti) neblikaly. Díky tomuto zapojení se i omezil počet potřebných vývodů čipu. Obě sedmi segmentovky jsou spínány přes PNP tranzistory BC638, neboť samotné výstupy PC0 a PC1 by jinak byly přetíženy. Kromě signalizace v podobě sedmi segmentových zobrazovačů, jsou na desce umístěny i dvě LED diody. Tyto LED jsou připojeny na výstupy PB0 a PB2 a slouží především jako pomůcka při odlaďování programu. Já je však použil i nadále, jako kontrolky napájení a správného chodu programu. Na vývody PB3-PB5 je připojena část programovacího konektoru, jehož zapojení vychází ze zapojení doporučeného výrobcem programátoru, firmou ASIX. Z vývodu PB1 je odebírán PWM signál, který je ještě zesilovaný NPN tranzistorem BC546A. Nakonec je na desce ještě umístěn konektor, připojený k vývodům PC4 a PC5. Tento konektor, pokud by chtěl někdo v budoucnu tento regulátor nějak rozšířit, slouží ke komunikaci s jiným procesorem, nebo EEPROM pamětí. Při návrhu DPS (**obr. 0.4.2**) tohoto obvodu jsem se snažil, aby spolu související součástky nebyly roztroušeny po plošném spoji. Všechna tlačítka jsem tedy umístil na pravou stranu, hned vedle sedmi segmentových displejů a LED diody usadil, hned vedle programovacího konektoru do spodní časti DPS. Díky tomuto rozmístění součástek jsem nemohl vodivé cesty vytvořit dokonale, a proto jsem byl nucen použít 3 drátové propojky.

Druhým navrhovaným obvodem je zdroj (**obr. 0.4.3**). Vstupní napětí, přiváděné z napájecího zdroje, je zde usměrněno výkonovým usměrňovacím můstkem, který jsem dostatečně dimenzoval. Dále je napětí vyfiltrováno dvěma kondenzátory a rozděleno do dvou větví. První větev vede přes napěťový stabilizátor na svorky, určené pro napájení MCU. Druhá větev vede přímo na svorky pro napájení spínacího obvodu. Při návrhu DPS (**obr. 0.4.4**) pro tento zdroj, jsem se snažil vytvořit co nejširší vodivé cesty, neboť je počítáno s protékajícími proudy do cca 10A. Takto široké cesty jsem ovšem vytvořit nemohl, proto jsem se rozhodl, že při osazování dle potřeby cesty posílím plným, měděným vodičem. Dále jsem dbal také na to, aby byly dané kondenzátory co nejblíže ke stabilizátoru.

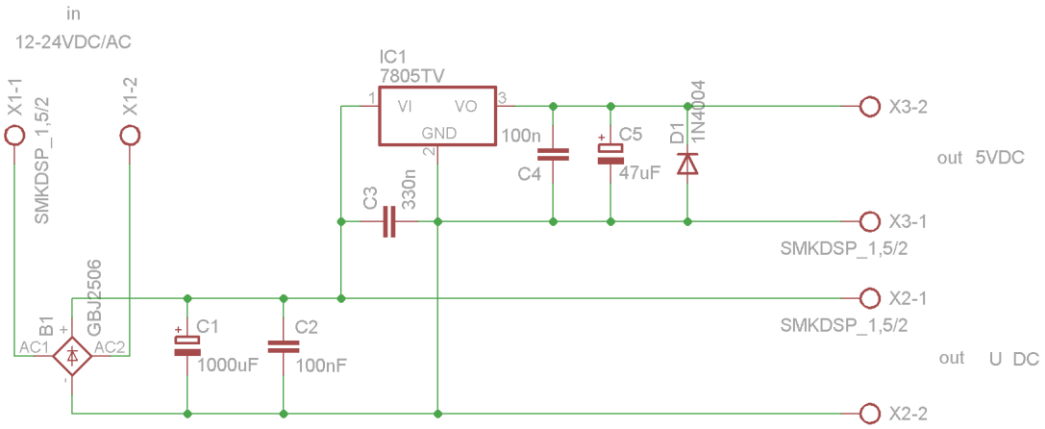
Posledním obvodem projektu je obvod spínače (**obr. 0.4.5**). U tohoto obvodu jsem vycházel z toho, že bude potřeba zkonstruovat budič výkonové tranzistoru IRF530. Začal jsem tedy tím, že jsem zkonstruoval základní budič s jedním tranzistorem BC546A a rezistorem R1. Ten však nebyl příliš dokonalý. Proto jsem do obvodu zařadil další dva tranzistory (BD139 a BD140), které jsou zapojeny jako AB zesilovač. Tím jsem dosáhl toho, že se na hradle tranzistoru IRF 530 dokáže velice rychle měnit logická jednička a nula. Toto zapojení již bylo celkem dobré, ale zbývalo vyřešit poslední nedostatek. Tímto nedostatkem bylo to, že maximální napětí, které je možné přivést na hradlo tohoto tranzistoru, je 15V. Napájecí napětí mého regulátoru se však může pohybovat v rozmezí 12-24V. Proto jsem do zapojení doplnil ještě rezistory R2, R3 a tranzistor Q5. Tato trojice součástek slouží jako stabilizátor, který zajistí, že se na hradlo nedostane více, než povolených 15V. Na konec jsem mezi vstupní napájecí svorky umístil dva kondenzátory, které by měly při velkých proudových špičkách dodávat výkon do zátěže, a tím filtrovat napětí. Při návrhu DPS (**obr. 0.4.6**) jsem stejně jako u obvodu zdroje dbal na co největší šířku proudově namáhaných cest. Proto jsem také počítal s tím, že je posílím měděným vodičem.



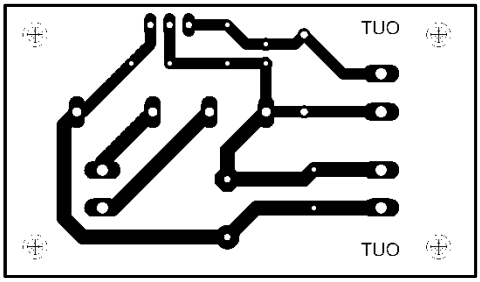
0.4.1


Pavel
Andel
PWM
ATMEGA8
pull-up res
ISP
plug
ISC SDA
ISC SCL
RESET
VDD

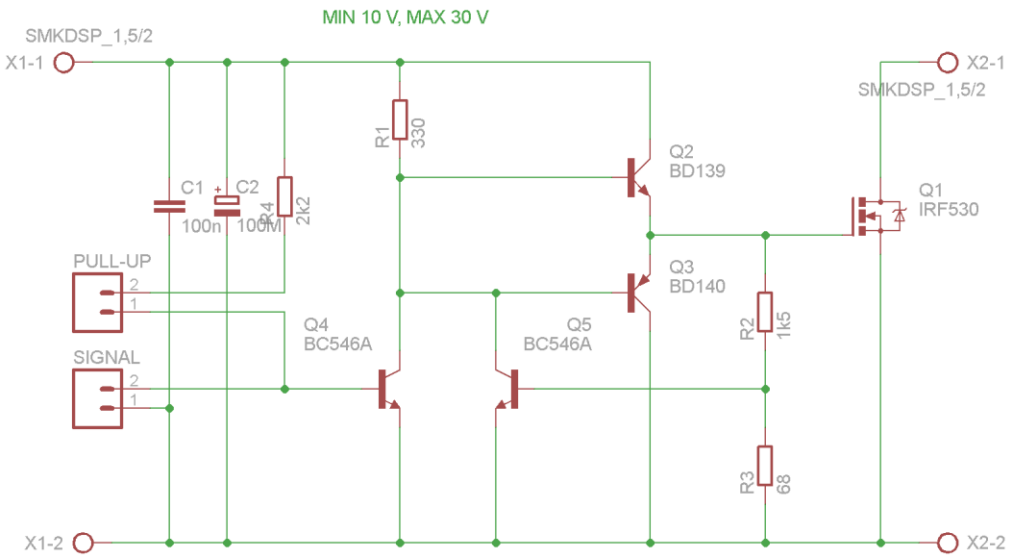
0.4.2


in
12-24VDC/AC
X1-1
X1-2
SMKDSP_1,5/2
IC1
7805TV
VI
VO
GND
100n
C4
C5
47uF
D1
1N4004
C3
330n
X3-2
out 5VDC
X3-1
SMKDSP_1,5/2
X2-1
SMKDSP_1,5/2
out U DC
X2-2
B1
GBJ2506
AC1
AC2
C1
1000uF
C2
100nF

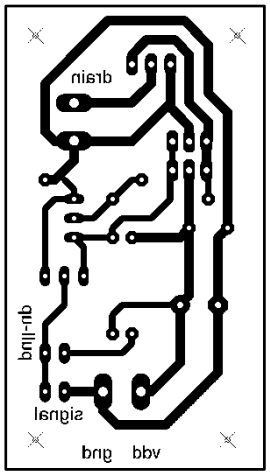
0.4.3


TUO
TUO

0.4.4



0.4.5



0.4.6

4.2 Výroba DPS

Všechny tři plošné spoje jsem vyrobil fotocestou ve školní laboratoři, v učebně D12. Nejprve jsem si nařezal cuprextit na požadované rozměry, s rezervou několika mm. Poté jsem tyto destičky vyčistil a vyleštil pomocí brusné vaty. Když byl měděný povrch dokonale lesklý, omyl jsem ho teplou mýdlovou vodou, kterou jsem ho zároveň i odmastil. Nyní jsem na měděnou plochu nanesl fotocitlivou vrstvu. To jsem provedl nástřikem spreje POSITIV. Po tomto nástřiku jsem cuprextitové destičky uložil do pece, kde jsem je nechal po dobu deseti minut a teplotě 75°C zaschnout. Během zasychání fotocitlivé vrstvy jsem si připravil předlohy plošných spojů. Ty jsem vytiskl laserovou tiskárnou na obyčejný kancelářský papír. Vytištěné předlohy jsem nyní přestříkal sprejem TRANSPARENT a nechal je několik minut oschnout. Tímto krokem jsem předlohy zprůhlednil a připravil je tak na krok následující. Tím je osvícení UV zářením. To jsem provedl tak, že jsem cuprextitové destičky se zaschlou fotocitlivou vrstvou naskládal na rovnou plochu, překryl je pečlivě vyrovnanými, zprůhledněnými předlohami a zakryl UV zářičem. Takto jsem nechal desky 6 minut osvítit. Když byly desky osvíceny, vyvolal jsem je v roztoku hydroxidu sodného (NaOH), čímž jsem

umyl fotocitlivou vrstvu na osvícených místech. Po vyvolání a umytí vodou jsem cuprextitové desky vložil do roztoku chloridu železitého (FeCl3). V tomto roztoku jsem desky nechal do chvíle, než byla fotocitlivou vrstvou neošetřená místa zcela odleptána. Po vyleptání a opláchnutí vodou jsem technickým lihem umyl zbytky fotocitlivé vrstvy. Bylo potřeba, aby byly nově vytvořené cesty chráněny, proto jsem je natřel kalafunovým lakem. Tento lak je ochrání proti oxidaci a navíc mi usnadní pozdější pájení. Posledním krokem ve výrobě DPS bylo vyvrtání otvorů pro vývody součástek. To jsem provedl pomocí malé, vysokootáčkové, stojanové vrtačky ve školní dílně.

4.3 Tvorba programu pro MCU

Hned v úvodu této kapitoly se přiznám, že jsem tento program nevytvářel sám. Mé znalosti v oboru programování nebyly pro tento projekt dostatečné, a proto jsem vyhledal pomoc.

Ovládací program pro přípravek generátoru PWM byl vypracován v prostředí ATMEL Studio (**screen 1**). Zdrojový kód byl napsán v jazyce C. Výhodou jazyka C je snazší orientace v kódu díky práci se symbolickými konstantami („#define") a možností uzavření souvisejících proměnných do struktury („struct"). V porovnání s assemblerem je zde také jednodušší tvorba podmíněných větvení programu, výpočtů (např. dělení) i případných cyklů (především cyklů „for"). I tak je jazyk C ještě dostatečně nízko úrovňový, aby umožnil případné optimalizace zápisu kódu podobně jako v assembleru (mimo něj umožňuje ATMEL Studio psát kód také v C++). Z těchto důvodů byl jazyk C vybrán při tvorbě tohoto programu.

Struktura programu je zvolena tak, aby se obsluhovaly jednotlivé procesy na základě vystavených vlajek. Ve výsledku se vlastně jedná o stavový automat. Výhodou této struktury je snadné rozšiřování programu i jeho postupné odlaďování, protože je vždy řešen jen jeden určitý stav automatu. Zároveň se tímto způsobem zrychlí průchod hlavní programovou smyčkou.

Samotný zdrojový kód se skládá z hlavičkového „PWMBoard.h" a zdrojového souboru PWMBoard.c. V hlavičkovém souboru lze najít nově definované konstantní výrazy a deklaraci jediné vytvořené funkce „InicializaceHardware()", která slouží pro počáteční nastavení použitých periférií (viz dále) při resetu procesoru. Mezi nejdůležitější symbolické konstanty tady patří definice bitových masek vlajek stavu PWM (začínají „STAVPWM") a stavu systému (začínají „STAVSYS"). Ty jsou dále používány v programu pro řízení chodu stavového automatu.

Na začátku zdrojového souboru jsou zavedeny knihovny <avr/io.h> a <avr/interrupt.h>, které slouží pro práci s periferiemi, zahrnují knihovny, kde jsou definovány datové typy „uint8_t" a umožňují práci s vektory přerušení. Jako třetí je zaveden hlavičkový soubor „PWMBoard.h". Následují deklarace a zároveň definice datových struktur „stavSys" a „stavPWM". Struktura „stavSys" zahrnuje proměnnou sloužící jako registr vlajek momentálního stavu programu a

proměnné počitadel časů pro obsluhu tlačítek a LED signalizace. Proměnná „stavSys.stavReg“ je definována jako „volatile“ jelikož její hodnota je měněna v přerušení (viz dále). Význam jejích bitů je následující:

7	6	5	4	3	2	1	0
			LD1S	LD0S	LED1	LED0	TIK

TIK… vystavení tohoto bitu značí, že došlo k přerušení a má být proveden jeden tik hodin systému

LED0… tento bit slouží pro informování stavového automatu, že má být provedena obsluha funkce signalizace LED0

LED1… význam bitu je podobný jako v případě LED0, nicméně v programu není použit

LD0S… tento bit ukazuje poslední stav LED0 (svítí/ nesvítí)

LD1S… stejný význam jako LD0S, ale pro LED1

Druhá struktura „stavPWM“ zahrnuje proměnné určující hodnotu, která má být zobrazena na sedmi segmentovém displeji, střídu PWM, počítadlo stisků tlačítek a proměnou „stavPWM.stavReg“, která slouží pro sledování stavu PWM. Význam jednotlivých bitů této proměnné je následující:

7	6	5	4	3	2	1	0
	BCD	SGID	SG	NHST	TLDS	TLON	TL

TL… bit je vystaven, pokud má dojít k obsluze tlačítek

TLON… signalizace, že při poslední obsluze tlačítek bylo nejméně jedno tlačítko stisknuto

TLDS… pokud je tento bit vystaven, pak uživatel drží tlačítko po určitou delší dobu (dlouhý stisk) – rychlá inkrementace nebo dekrementace střídy

NHST… bit, který signalizuje, že je k dispozici nová hodnota střídy a je ji třeba nastavit v příslušném registru periférie TIMER1

SG… při vystavení je požadována obsluha sedmi segmentového displeje

SGID… identifikace, zda má být obsloužen SEGMENT0 (řád jednotek) nebo SEGMENT1 (řád desítek)

BCD… signalizace, že má být proveden přepočet hodnoty střídy do kódu BCD, který lze zobrazit na sedmi segmentovém displeji

Jako poslední je deklarováno a definováno pole „segmentNum[]“, které slouží pro převod čísla BCD na číslo pro sedmi segment. Indexy tohoto pole přesně odpovídají číslu, které se má zobrazit na displeji.

Dále je ve zdrojovém kódu definována trojice funkcí. Funkce „main()“ je volána při resetu procesoru a obsahuje hlavní programovou smyčku. Ve funkci „InicializaceHardware()“ jsou provedena počáteční nastavení vstupně výstupních portů procesoru a periférií časovačů TIMER0 a TIMER1. TIMER0 je nastaven tak, aby po přetečení generoval přerušení „TIMER0_OVF“. To nastává při použitém krystalu 8 MHz s frekvencí 31,25 kHz a slouží pro účely sledování systémového času, tj. s touto frekvencí nastává v našem programu jeden tik hodin. Druhý časovač TIMER1 je využit v režimu PWM. Před časovač je zařazena dělička frekvence 8, tudíž inkrementace jeho hodnoty nastává s frekvencí 1 MHz. Rozlišení střídy je nastaveno na hodnotu 1024, čímž je určen výsledná frekvence PWM na zhruba 1 kHz. Jedno procento změny střídy ve výsledném programu odpovídá přibližně době sepnutí 10 µs.

Třetí funkcí je obsluha přerušení po přetečení časovače TIMER0 „ISR(TIMER0_OVF_vect)“. Jediné co se při této obsluze děje je vystavení vlajky tiku hodin v proměnné „stavSys.stavReg“.

Na začátku funkce „main()“ je zavolána funkce „InicializaceHardware()“ a jsou nastaveny počáteční hodnoty proměnných ve strukturách „stavPWM“ a „stavSys“. Dále je již hlavní programová smyčka, která se skládá ze dvou částí, časování obsluhy jednotlivých procesů a stavového automatu. Časování obsluhy jednotlivých procesů je vykonáváno v závislosti na vystavení vlajky STAVSYS_TIK v přerušení TIMER0. Pro měření času je uvnitř podmínkového větvení pomocí proměnné „stavSys.casPred“ programově vytvořen další dělící poměr 31. Díky tomu dochází k inkrementaci časů obsluhy tlačítek „stavSys.casTLAC“ a obsluhy LED „stavSys.casLED0“ přibližně po 1 ms. Cílové časy obsluhy jednotlivých procesů tak lze v dalších podmínkových větveních zadávat přímo v ms. Konkrétně obsluha tlačítek je volána jednou za 100 ms (viz symbolická konstanta OBSLUHA_TLAC_MS) a obsluha LED jednou za 200 ms (viz symbolická konstanta OBSLUHA_LED0_MS). Vlajka pro obsluhu displeje je vystavena při každém tiku hodin, tj. displej se obsluhuje přibližně s frekvencí 31,25 kHz.

Jelikož nastavení vlajek obsluhy jednotlivých procesů se děje ve výše zmíněném podmínkovém větvení a nikoliv přímo v přerušení, je vhodné, aby byl běh programu co nejméně zdržen následujícím průchodem stavovým automatem. Z toho důvodu se při každém průchodu stavovým automatem, provádí jen jeden jeho stav a všechny stavy jsou řešeny tak, aby obsahovaly jen několik operací. Díky tomu dochází k časté kontrole vystavené vlajky tiku hodin a není tak narušeno časování obsluh jednotlivých procesů. Kritická je v tomto případě především obsluha zobrazení displeje, která je volána nejčastěji a zároveň by měla být vykonávána co nejpravidelněji.

Stavový automat je vytvořen pomocí podmínkového větvení „if - else if“. Tím je zajištěna i priorita obsluhy jednotlivých procesů, protože se dané podmínky vyhodnocují postupně od shora (první podmínka v „if“) dolů (poslední podmínka s „elseif“). Základní pravidlo v podobných případech zní, že nejvyšší prioritu mají mít procesy, které se vykonávají nejčastěji. Proto se jako první kontroluje vystavení vlajky obsluhy zobrazení displeje STAVPWM_SG.Displej je provozován v dznamickém režimu, proto se v tomto stavu nejdříve oba segmenty zhasnou a podle vlajky STAVPWM_SGID se rozhodne, který segment

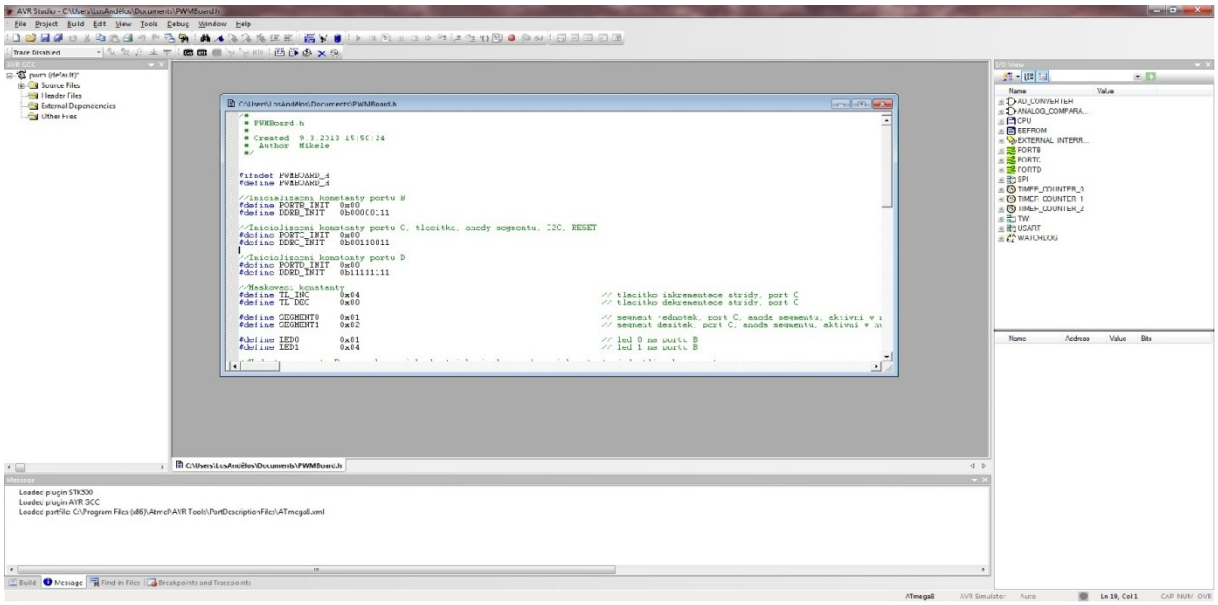
má být rozsvícen. Na port D se následně přesune zobrazovaná hodnota, příslušný segment se rozsvítí a změní se stav vlajky STAVPWM_SGID na druhý z obou segmentů.

Druhou nejvyšší prioritu má obsluha tlačítek, která se dělí cekem na tři stavy automatu. V prvním základním stavu (vlajka STAVPWM_TL) se provede načtení hodnot tlačítek a případně provede inkrementace nebo dekrementace střídy v proměnné „stavPWM.stridaPWM“. Pokud došlo ke stisku některého tlačítka, tak je vystavena vlajka STAVPWM_NHST. V tomto stavu se také provádí měření doby stisku tlačítek na základě vlajek STAVPWM_TLON a STAVPWM_TLDS. Při prvním stisku jakéhokoliv tlačítka nedojde k další změně střídy, dokud se obsluha nezavolá počtem definovaným v symbolické konstantě DLOUHY_STISK. Pokud je tlačítko drženo přes tuto dobu je následně střída měněna dle počtu volání obsluhy v symbolické konstantě KRATKY_STISK. V době psaní tohoto dokumentu byly obě hodnoty nastaveny tak, aby po prvním stisku tlačítka bylo ignorováno 10 volání obsluhy (odpovídá 1 s) a po uplynutí dané doby byla střída měněna s každým čtvrtým voláním (odpovídá 0,4 s).

Ve stavu detekujícím vlajku STAVPWM_NHST se provádí přepočet procentní hodnoty střídy na údaj, který má být zadán do registru „OCR1A“. Přepočet vychází z faktu, že 100 % odpovídá hodnota 1024, jak již bylo uvedenodříve. Zároveň je vystavena vlajka STAVPWM_BCD, na základě které se vybaví následující stav, ve kterém se provede převod hodnoty střídy do kódu BCD. Převedené hodnoty jsou uloženy v proměnných stavPWM.segment0BCD a stavPWM.segment1BCD a ty jsou následně využívány při obsluze displeje jako indexy pole „segmentNum[]“.

Posledním stav, který má zároveň nejnižší prioritu, je obsluha LED0. Zde je pouze měněn stav diody z rozsvíceno na zhasnuto a naopak. Uvedená funkce má význam především při signalizaci chyby zařízení, jelikož detekuje chod systémových hodin (vystavení přerušení od TIMER0) a úspěšný průchod programovou smyčkou.

Když byl program hotový, vyexportoval jsem soubor hex, který je za potřebí pro naprogramování čipu. To jsem provedl kliknutím na záložku Build, čímž se mi soubor hex vytvořil ve složce projektu.



Screen 1

4.4 Osazení součástkami, oživení obvodu, programování čipu

Všechny plošné spoje jsem osazoval, za pomoci stolní mikropájky a trubičkového cínu, ve své modelářské dílně podle osazovacích plánů (**obr. 0.4.7; 0.4.8; 0.4.9**).

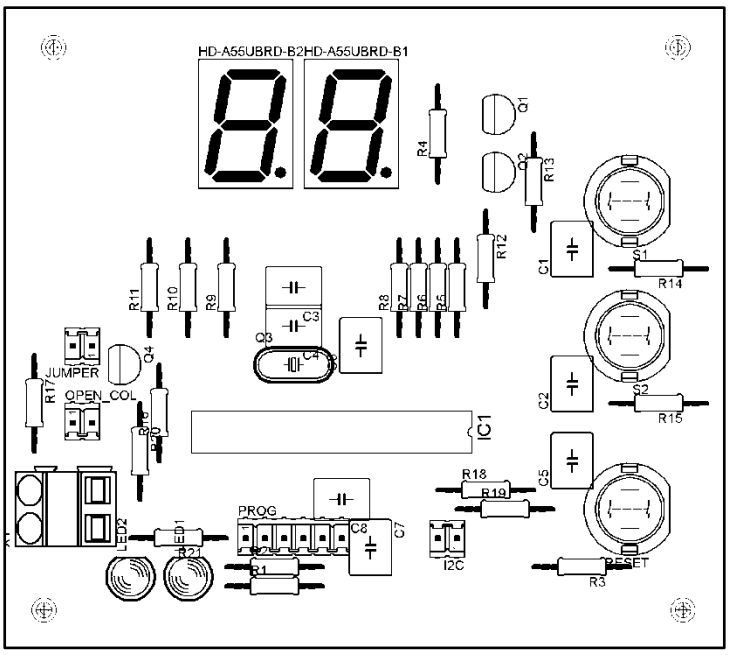
Jako první jsem začal plošné spoje osazovat rezistory, diodami a podobně nízkými součástkami. Záměrem tohoto postupu je, aby mi součástky při otočení plošného spoje pájecími body vzhůru nevypadávaly, a aby se o pracovní plochu přitiskávaly ke cuprextitu. Takto jsem pokračoval a osazoval stále většími a většími součástkami, až jsem plošné spoje zcela osadil. Samotné pájení jsem prováděl tak, že jsem hrot mikropájky přiložil k vývodu součástky, počkal, až se vývod dostatečně ohřeje, přitiskl konec trubičkového cínu a nechal ho volně roztéct po pájecí plošce. Dbal jsem na to, aby cínu nebylo použito zbytečně mnoho či naopak příliš málo.

Při návrhu krabičky výrobku, jsem si všiml, že by byly sedmi segmentové displeje příliš zapuštěné a bylo by na ně špatně vidět. Proto jsem se rozhodl, že displeje posunu výš. Nejprve jsem se o to pokusil za pomoci „dutinek“, které však displeje přizvedly pouze o pár mm. Přistoupil jsem tedy k prodloužení vývodu sedmi segmentovek. Ty jsem prodloužil připájením odstřihaných vývodů z rezistorů. Když byly všechny vývody prodlouženy, navlékl jsem na ně kousky bílé bužírky. To jsem udělal jednak z estetických důvodů a také kvůli izolaci, aby se jednotlivé vývody displeje nemohly navzájem vyzkratovat. Tímto jsem problém vyřešil (**obr. 0.4.10**).

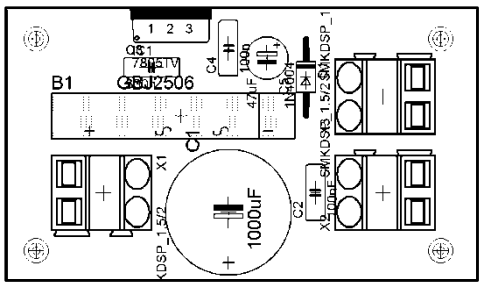
Když byly všechny plošné spoje osazené, podrobil jsem je zkoušce. Nejprve jsem vyzkoušel zdroj, zda je na výstupu stabilizátoru opravdu 5V. Vše bylo tak, jak má být. Poté jsem k tomuto napájení připojil desku s MCU a zahájil programování.

Samotné programování nebylo nikterak těžké. Nejprve jsem pomocí USB kabelu připojil programátor PRESTO k mému PC a spustil program ASIX UP (**screen 2**). Poté jsem ve správním pořadí připojil 6/7 možných vodičů, vedoucích z programátoru, na programovací

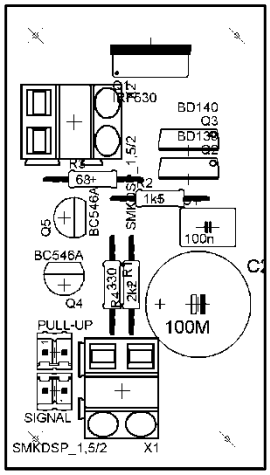
konektor (**obr. 0.4.11**). V programu ASIX UP jsem nejprve navolil správný typ mikroprocesoru, frekvenci oscilátoru (**obr. 0.4.12**) a otevřel jsem si uložený hex (**screen 3**). Posledním krokem programování bylo spuštění programování, při kterém programátor nahrál data do paměti mikročipu. Po dokončení programování se obvod ihned rozeběhl a fungoval správně. Obvod spínače jsem bohužel nemohl vyzkoušet, proto jsem s ním vyčkal, než bude při kompletaci propojen s řídícím odvodem.



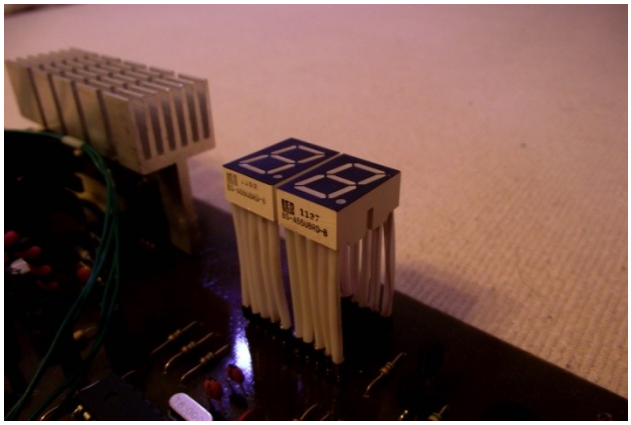
0.4.7



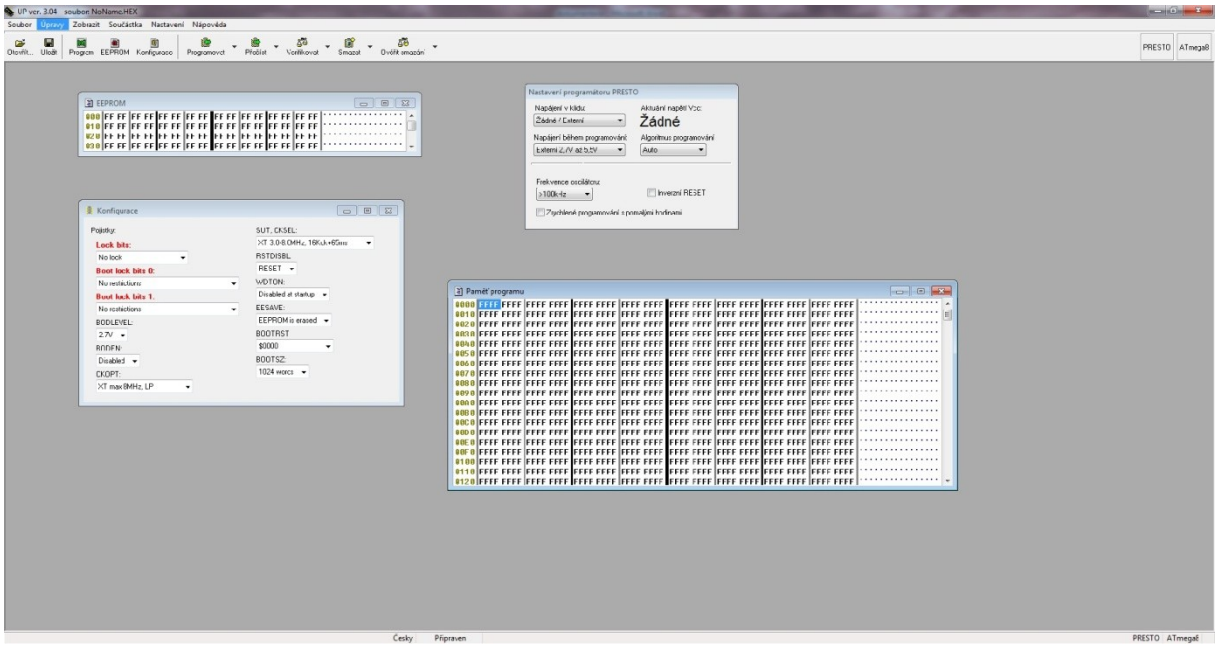
0.4.8



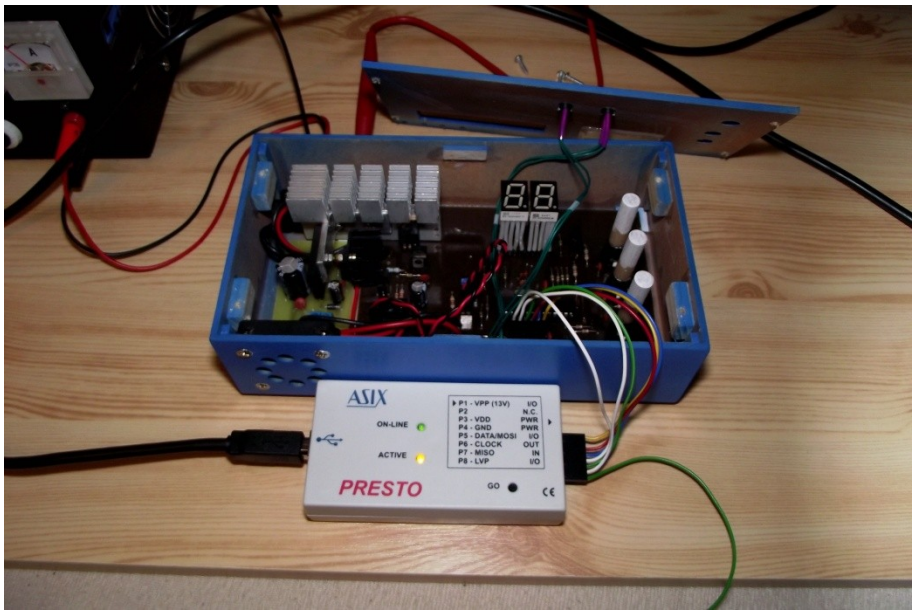
0.4.9



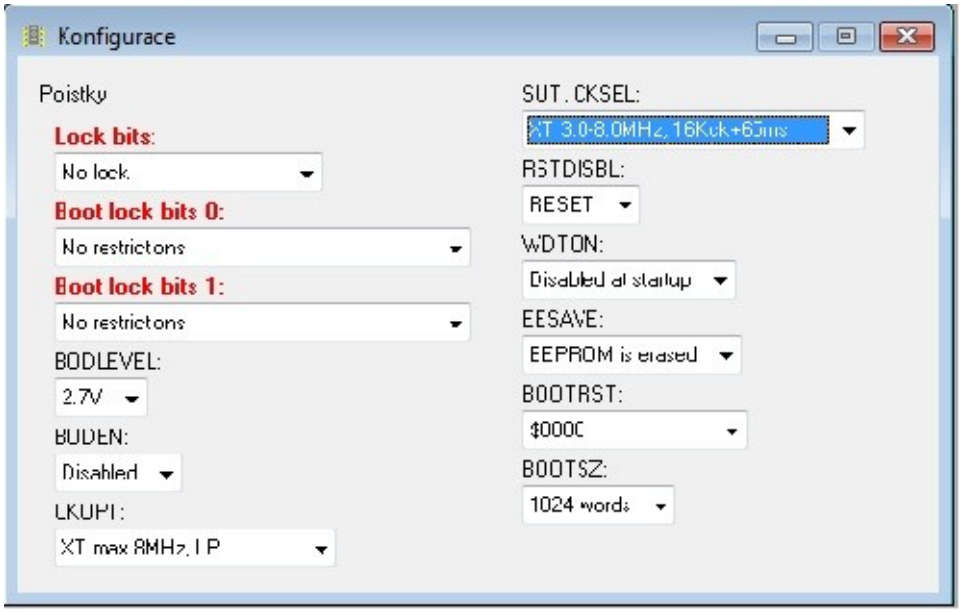
0.4.10



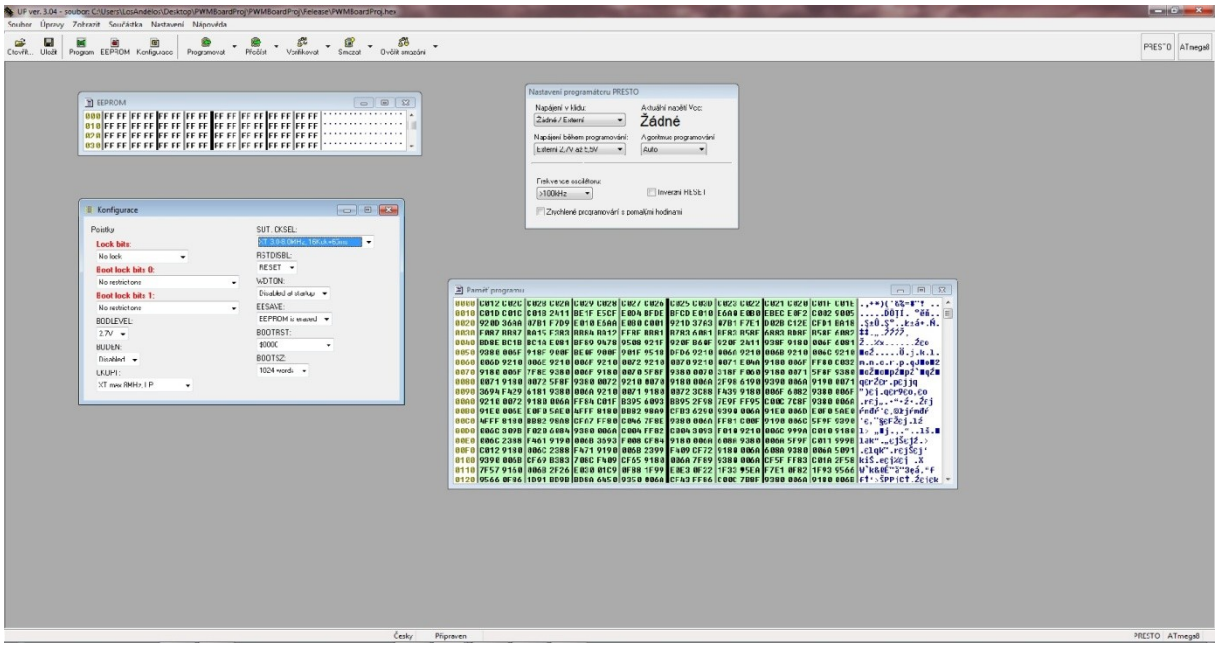
Screen 2


ASIX
ON-LINE
ACTIVE
PRESTO
GO

0.4.11


Konfigurace
Poistky
Lock bits:
No lock
Boot lock bits 0:
No restrictons
Boot lock bits 1:
No restrictons
BODLEVEL:
2.7V
BUDEN:
Disabled
LKUPI:
XT max 8MHz, I P
SUT. CKSEL:
XT 3.0-8.0MHz, 16Kck+65ms
RSTDISBL:
RESET
WDTON:
Disabled at startup
EESAVE:
EEPROM is erased
BOOTRST:
$000C
BOOTSZ:
1024 words

0.4.12



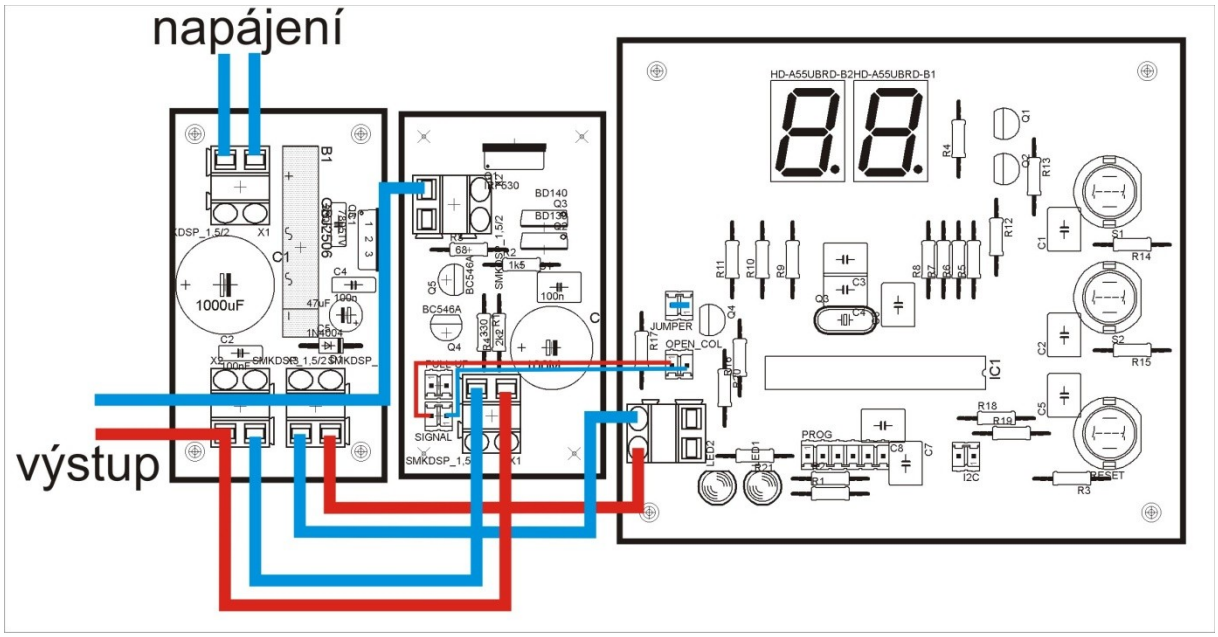
Screen 3

4.5 Kompletace, výroba krabičky

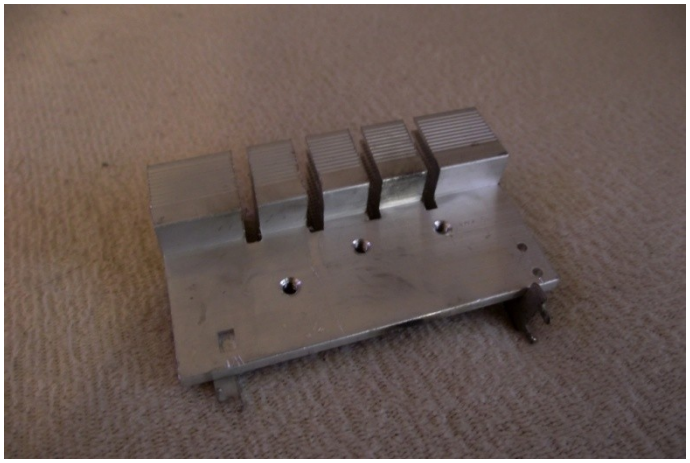
Vyzkoušené, osazené a připravené plošné spoje bylo nutno zkompletovat do jednoho velkého obvodu. To jsem provedl propojením jednotlivých desek měkkými, ohebnými, měděnými vodiči. Jednotlivé desky jsem mezi sebou propojil, viz **obr. 0.4.13**. Když byly plošné spoje propojeny, začal jsem s řešením umístění chladiče. Vzhledem k omezenému výběru jsem pro svůj obvod zvolil chladič vymontovaný ze starého PC zdroje (**obr. 0.4.14**). Tento chladič jsem umístil tak, abych na něho mohl přímo připevnit spínací výkonový tranzistor IRF 530. Tento tranzistor však není jedinou součástkou, která potřebuje být připevněna na chladiči. Chladič totiž potřebuje i výkonový usměrňovací můstek. Ten je ale dosti předimenzovaný, proto jsem se rozhodl, že ho k chladiči připojím pouze pomocí kusu 3mm hliníkového plechu, ohnutého do tvaru L.

Nyní nastala chvíle pro výrobu montážní krabičky. Tu jsem vyrobil z 3mm tlustého sololitu, používaného též při výrobě nábytku. Nejprve jsem si celý obvod i s chladičem řádně změřil, aby vše sedělo jak má. Poté jsem všechny díly krabičky naměřil, nařezal a očistil smirkovým papírem. Do připravených dílů jsem poté vyvrtal a vybrousil všechny potřebné otvory pro displej, tlačítka, chladič, signalizační LED, svorkovnici a další. Všechny díly, kromě horního krytu, jsem poté slepil lepidlem HERKULES (**obr. 0.4.15**). Když lepidlo dostatečně zaschlo, celou krabičku jsem znovu obrousil smirkovým papírem a přestříkal dvěma vrstvami modré barvy. Konečný vzhled krabičky (bez polepů) lze vidět na obrázku **0.4.16**.

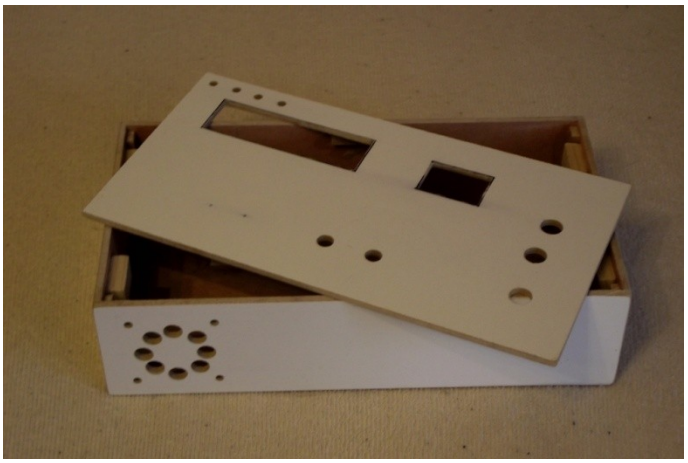
Když byla montážní krabička hotova, mohl jsem do ní umístit plošné spoje. Ty jsou ke dnu krabičky připevněny pomocí M3 šroubů s matičkami. Dále jsem ještě přišrouboval na vnitřní stranu krabičky 40mm ventilátor, který jsem do krabičky umístil kvůli lepšímu odvodu tepla z chladiče. Posledním úkolem bylo prodloužit tlačítka tak, aby jejich konce koukaly z horního krytu. S tím jsem si hravě poradil a tlačítka prodloužil přilepenými, bukovými válečky. Tímto jsem projekt celkově dokončil a připravil k prvnímu použití.



0.4.13



0.4.14



0.4.15

0.4.16

5)OVĚŘENÍ FUNKČNOSTI, TEST

5.1 Ovládání regulátoru, popis signalizace

Na horní ploše výrobku jsou umístěny 3 ovládací tlačítka, 2 LED diody a dvoumístný sedmi segmentový displej. Pomocí prvního a druhého tlačítka shora lze ovládat výstupní výkon regulátoru. Po připojení napájecího napětí se regulátor automaticky nastaví do počátečního stavu = výstupní výkon 0%. Nyní lze pomocí stisků prvního tlačítka od shora přidávat výstupní výkon regulátoru po 1%, nebo tlačítko podržet, a po uplynutí času cca 1s se začne výstupní výkon zvyšovat rychleji. Druhé tlačítko od shora funguje obráceně a výkon

s ním snižujeme. Aktuálně nastavený výkon, v rozmezí 0-99%, lze odečítat z displeje. Třetí a také poslední tlačítko je reset, kterým regulátor uvedeme do počáteční polohy. Lze ho tedy využít, jako rychlé STOP tlačítko, kterým snížíme jakýkoli nastavený výkon ihned na 0%. Použité LED diody složí jako kontrolky. První LED z leva se po připojení regulátoru na napájení rozsvítí a po odpojení zhasne. Druhá LED signalizuje správný běh programu mikroprocesoru. Pokud LED bliká, program probíhá bezchybně. Pokud LED zhasne, nebo zůstane trvale svítit, nastala chyba a je nutný reset regulátoru.

5.2 Ověření funkčnosti

Při prvním spuštění regulátoru jsem pro jistu změřil, zda je mikroprocesor opravdu napájen 5V a zda se některá součástka podezřele nezahřívá. Jako první ověření správné činnosti PWM jsem přímo na signálový výstup z desky s MCU připojil LED diodu s rezistorem. Zde vše fungovalo správně, a proto jsem signál přivedl i na spínací koncový obvod. Na ten jsem připojil malou 12V auto žárovku a sledoval, jak se s přidáváním výkonu postupně rozsvěcuje vlákno žárovky. Obvod tedy fungoval správně a já se uchýlil k prvnímu výkonovému testu.

5.3 Test

Při prvním testu jsem jako zátěž použil 12V SS motor z ventilátoru zn. Škoda. Po přidání výkonu začal motor již od prvního % slabě bzučet. Tento zvuk se zesiloval zhruba do 15% výkonu, kdy se motor začal pomalu otáčet. S přidávaným výkonem motor lineárně zvyšoval své otáčky až do maxima. Při maximálních otáčkách motor na prázdno odebíral proud 1,3A. Při tomto proudu se chladič regulátoru ani po několika minutách nezahřál.

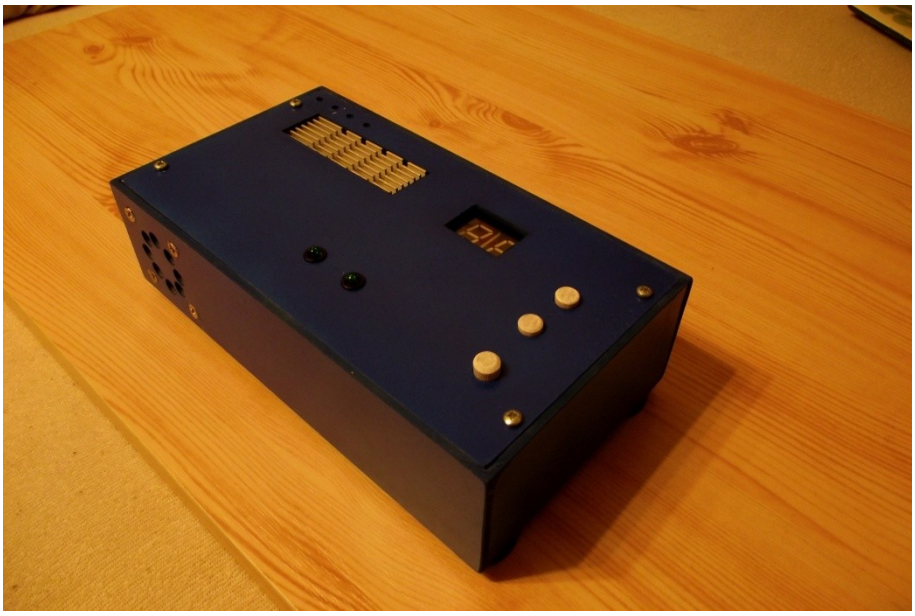
Při druhém testu jsem jako zátěž regulátoru použil cca 30cm dlouhý kus odporové struny. S touto zátěží protékal regulátorem při plném nastaveném výkonu proud 5,4A. Při nastaveném výkonu 50%, měl protékající proud hodnotu 2,8A. Tímto jsem dokázal, že nastavený procentuální výkon regulátoru odpovídá zhruba skutečnému výkonu. Při spínaném proudu 5,4A bylo i po 10min provozu možné na chladiči regulátoru udržet dlaň. Z toho usuzuji, že by regulátor dokázal spínat i daleko větší proudy po delší dobu a přitom by se nepřehřál. V žádném případě bych však nedoporučoval dlouhodobě přesahovat proud 10A.

6) ZÁVĚR

Důvodem toho projektu byl fakt, že jsem chtěl zkonstruovat regulátor pro odporovou pilu, kterou hojně využívám ve své modelářské dílně při stavbě letadel. Tento projekt je již několikátý mnou postavený regulátor, avšak zatím ten nejlepší. Při návrhu ani stavbě tohoto regulátoru se nevyskytly žádné větší obtíže, které by mou práci ztěžovaly. Závěrečné testy regulátoru dopadly také dobře, a proto se nebojím říci, že tento projekt dopadl výborně. Vím,

že tento PWM regulátor s ATmega8 není nejlepší z nejlepších, a že by nebyl vhodný například k prodeji, ale já jsem s ním velice spokojený. Myslím si, že se velice povedl a pro účely, pro které byl navrhován, je perfektní.

Pokud bych chtěl tento projekt v budoucnu ještě vylepšovat, určitě by přibyly novinky, jako například automatické řízení ventilátoru, paměť nastavených hodnot výkonu atd. To je ovšem otázka dalšího vývoje.



Obrázek hotového výrobku

7) ODKAZY

Informace o mikroprocesoru:
http://www.gme.cz/mikroprocesory-atmel-avr-mega/atmega8-16pu-p432-201/

Informace o programátoru ASIX PRESTO:
http://www.asix.cz/prg_presto.htm