VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH A REALIZACE REGULÁTORU OTÁČEK STEJNOSMĚRNÉHO MOTORU PRO MOBILNÍ ROBOT

DESIGN AND REALIZATION OF THE DC MOTOR CONTROLLER FOR MOBILE ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Jan Hrbáček

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jiří Krejsa, Ph.D.

BRNO 2008

Zadání

Cíle, kterých má být dosaženo:

1. Proveďte rešerši tématu a obdobných zařízení.
2. Navrhněte HW řešení, vytvořte schéma zapojení a návrh plošného spoje.
3. Navržený regulátor realizujte.
4. Zhodnoťte výsledné řešení vzhledem k výsledkům reálného nasazení.

Charakteristika problematiky úkolu:

Navrhněte a realizujte obousměrný elektronický regulátor otáček stejnosměrného motoru určený pro nasazení v mobilním robotu. Zvolte vhodnou platformu řídicí jednotky, přiměřené řešení výkonového stupně dimenzované pro stálý příkon motoru do 250 W při napětí do 30 V. Vytvořte programové vybavení řídicí elektroniky. Zvolené řešení prakticky realizujte a odlaďte jeho funkčnost.

Abstrakt

Bakalářská práce se zabývá návrhem obousměrného regulátoru otáček určeného pro stejnosměrný komutátorový motor do výkonu 250 W. V rešeršní části jsou diskutovány různé způsoby realizace jednotlivých subsystémů regulátoru a proveden výběr technologií použitých při realizaci regulátoru dle zadání. V části popisující návrh je rozvedeno provedení důležitých konstrukčních uzlů, činnost vyvinutého firmware i provedení prototypu a finálního výrobku.

Výsledkem této práce jsou funkční exempláře regulátoru použité pro regulaci pohonu v kolovém robotu Bender 2. Při provozu robotu nebyla shledána žádná významná závada a byla ověřena spolehlivost ve všech provozních podmínkách.

Abstract

The goal of the bachelor's thesis is to design a bidirectional electronic speed controller for a brushed DC motor with the power up to 250 W. The background research part introduces various ways of each subsystem's implementation and selects proper technologies to be used to build the controller according to the specification. The description of all important design details, developed firmware functionality as well as design of the prototype and the end product is mentioned in the thesis' part concerning construction.

The final controller was successfully tested equipped in the wheeled robot Bender 2. It has been found fully functional and reliable in all operating conditions.

Bibliografická citace

HRBÁČEK, J. *Návrh a realizace regulátoru otáček stejnosměrného motoru pro mobilní robot.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 49 s. Vedoucí bakalářské práce Ing. Jiří Krejsa, Ph.D.

Čestné prohlášení

Čestně prohlašuji, že bakalářskou práci na téma *Návrh a realizace regulátoru otáček stejnosměrného motoru pro mobilní robot* jsem vypracoval samostatně pod vedením svého vedoucího bakalářské práce s použitím odborné literatury, kterou jsem všechnu citoval v seznamu literatury.

V Brně dne 21.5.2008

…….

Jan Hrbáček

Poděkování

Na tomto místě chci poděkovat Ing. Jiřímu Krejsovi, Ph.D. a Ing. Stanislavu Věchetovi, Ph.D. za vedení, cenné rady a přátelský přístup během zpracování této bakalářské práce.

Stejně tak bych chtěl poděkovat své rodině a přátelům za podporu v průběhu celého studia.

Obsah

1. Úvod

Řízení a regulace coby inženýrská disciplína má své nezastupitelné místo mezi ostatními obory již od dob počátků mechanizace průmyslu. Od původních mechanických regulátorů se sice postupně přešlo přes elektromechaniku k elektronice, avšak původní idea zůstává: působit na řízenou soustavu takovým způsobem, aby se chovala podle našich představ.

Obecnou soustavu můžeme řídit principiálně dvěma způsoby, rozlišenými přítomností nebo nepřítomností *zpětné vazby*. Řídíme-li systém bez zpětné vazby, mluvíme o tzv. *ovládání*. V takovém případě nevíme, co se v soustavě děje a jaké důsledky naše zásahy do soustavy mají, můžeme se tedy pouze spoléhat na předpoklady vycházející z naší znalosti dané soustavy. Pokud však libovolným způsobem sledujeme, jak soustava na řízení reaguje, a tyto poznatky využíváme pro změnu parametrů řízení, nazýváme takovéto řízení *regulací*.

Regulace otáček motoru je jednou ze základních a nejrozšířenějších regulačních úloh. Bylo uveřejněno mnoho článků zabývajících se konstrukcí regulátoru zejména stejnosměrného motoru (tedy prakticky zadáním mé práce). Výsledné regulátory jsou určeny převážně pro součinnost s pohony kategorie mikromotorů[1] a použití v robotice. Přestože se jejich konstrukce kus od kusu liší, lze ji rozdělit na několik subsystémů, které obecně nalezneme v prakticky každém takovém regulátoru. Má práce je proto dělena podobně.

Oblíbenost stejnosměrného (DC) motoru pramení především z jeho snadného řízení a přijatelného poměru výkon/hmotnost. Z těchto důvodu byl také vybrán pro pohon kolového robotu *Bender 2* vyvíjeného na Ústavu mechaniky těles, mechatroniky a biomechaniky Fakulty strojního inženýrství Vysokého učení technického v Brně, pro nějž je popisovaný regulátor vyvíjen. V robotu jsou použity tyto motory hned dva – pro oddělený pohon obou zadních kol – čímž je odstraněna nutnost použití konstrukčně náročného diferenciálu, který je tak realizován softwarově – nastavením různých rychlostí jednotlivým regulátorům.

Mým úkolem je regulátor požadovaných vlastností navrhnout, sestavit a otestovat jeho funkčnost v reálném nasazení. Cílem této práce je pak shrnout informace o mé konstrukci regulátoru i ostatních vhodných alternativách realizace jednotlivých subsystémů.

1 Mikropohonem je zpravidla myšlen stroj o výkonu do 400 W (viz [1]).

2. Rozbor zadání, formulace cílů

Ze zadání vyplývají tyto cíle:

1. **Proveďte rešerši tématu a obdobných zařízení** – vzhledem k tomu, že obdobných zařízení, které však potřeby zadání nesplňují zcela, existuje velké množství, budu se zabývat pouze jejich částmi odpovídajícími požadavkům na některý ze subsystémů regulátoru.
2. **Navrhněte HW řešení, vytvořte schéma zapojení a návrh plošného spoje** – v kapitole 4.6.1. Požadavky na mechanickou konstrukci uvádím omezení, která na konstrukci regulátoru klade mechanické provedení robotu *Bender 2*. Dle těchto požadavků je hardware regulátoru navržen.
3. **Navržený regulátor realizujte** – tento bod zadání je poměrně zásadní, neboť výroba dvou exemplářů regulátoru je cílem celé práce.
4. **Zhodnoťte výsledné řešení vzhledem k výsledkům reálného nasazení** – zkušenosti z provozu shrnu s ohledem na vhodnost zvolených technologií a celkovou spolehlivost.

Zadání dále specifikuje žádané parametry regulátoru:

Příkon motoru:	do **250 W**
Napájecí napětí motoru:	do **30 V**

3. Rešerše tématu, možná řešení problému

Základem úspěšného vyřešení libovolného technického problému je zpravidla kvalitní rešerše zpracovávaného tématu a dobrý přehled o již existujících řešeních. Především v případě komplexnějších problémů je mnohdy výhodnější zvolit známé a vyzkoušené řešení, než se pouštět do vlastního návrhu všech subsystémů. Právě kvalitní rešerše pak umožňuje vybrat nejlépe vyhovující dostupnou technologii, případně rozhodnout o vývoji řešení vlastního.

Přestože patří regulace otáček stejnosměrného motoru mezi základní regulační úlohy, skládá se regulační smyčka z poměrně velkého množství konstrukčních uzlů, které je možné realizovat různými způsoby. Jejich základní přehled uvádím v této kapitole, a to včetně nejrelevantnějších možností realizace.

3.1. Snímání otáček

Nutnou podmínkou optimální zpětnovazební regulace je co možná nejaktuálnější a nejméně zkreslená informace o parametrech soustavy, které regulujeme. V našem případě to jsou otáčky na hřídeli řízeného motoru.

Kvalita této informace je závislá na mnoha faktorech a prakticky každý z těchto faktorů ji dokáže naprosto znehodnotit. Záleží tak na kvalitě celého měřicího řetězce, samotným snímačem počínaje a (v případě číslicových systémů, tedy i v našem případě) A/D převodníkem konče.

Regulace motorů je specifická vysokou úrovní rušivých signálů, které mohou při špatné volbě technologie měření otáček a „dopravy“ elektrického signálu ze senzorů k převodníkům naměřený údaj značně zkreslit. Určitou výhodou by mohla být velmi malá vzdálenost řízeného motoru a samotného regulátoru, která zmíněnou nevýhodu do jisté míry kompenzuje.

Snímače otáček se dají rozdělit podle typu informace, kterou produkují, na snímače *spojité* (výstupní veličina je spojitého charakteru; zde tachodynamo) a *diskrétní* (výstupní veličina je kvantována; ostatní zde zmíněné snímače).

Diskrétní snímače samy o sobě obvykle neposkytují informaci o směru otáčení. Tato informace je ale pro konstrukci obousměrného regulátoru nezbytná. Užívá se proto nejčastěji tzv. *kvadraturního uspořádání*, kdy je třeba dvou senzorů nastavených tak, aby byl jejich signál vzájemně posunutý o čtvrt periody. Z posloupnosti příchodu úrovní obou signálů lze posléze rekonstruovat směr otáčení. Dále o tomto tématu pojednávám v kapitole 4.2.1. Dekódování signálu IRC.

Pro realizaci mého regulátoru jsem z uvedených možností vybral optický kvadraturní snímač (především kvůli přijatelnému poměru mezi rozlišením snímače a jeho složitostí), nicméně vstup regulátoru je prakticky omezen pouze na typ snímače (diskrétní impulsní), lze tedy použít např. i Hallovu sondu nebo indukčnostní snímač (obé v kvadraturním zapojení).

V této části rešerše budu problematice měření rychlosti otáčení věnovat jen poměrně omezený prostor. Mnoho dalších informací (které ale přímo nesouvisí s tématem mé práce nebo jdou pro mé potřeby zbytečně do hloubky tématu) lze nalézt zvláště na anglické Wikipedii, viz např. [2]. Lze také doporučit pramen [3], který v příslušné kapitole na vhodné úrovni shrnuje dostupné technologie měření otáček.

3.1.1. Tachodynamo

Jedná se o prakticky nejstarší a nejznámější metodu měření otáček. Výstupním signálem tachodynama je stejnosměrné napětí přímo úměrné měřeným otáčkám, přičemž je možné z polarity výstupního napětí zjistit i smysl otáčení.

Jak již bylo zmíněno, tachodynamo poskytuje spojitou informaci o rychlosti otáčení sledovaného stroje (pochopitelně nemusí jít jen o motor). Tato vlastnost je výhodná v oblasti nízkých otáček, kde je při rozumném rozlišení A/D převodníku zaručena stálost kvality informace nezávisle na vzorkovací frekvenci diskrétního regulátoru.

Mezi nevýhody tachodynama by se dala zařadit určitá náchylnost k zarušení signálu, protože filtrace analogových signálů je obecně mnohem složitější a nejistější, než je tomu u signálů digitálních (hlavním prvkem analogové informace je přesná velikost úrovně signálu, zatímco u digitálního signálu je úroveň rozdělena pouze na dvě skupiny o různé logické hodnotě).

Vzhledem k tomu, že tachodynamo (resp. obecně spojitý senzor) nebylo zvoleno pro snímání otáček v mé konstrukci regulátoru, bylo by nad rámec této práce rozepisovat se šířeji o jeho konstrukci.

3.1.2. Optické senzory

Optické senzory pracují zpravidla na principu zdroje a detektoru světla, mezi nimiž se pohybuje dírkovaná clona fixovaná na sledované hřídeli. Poměrně obvyklé jsou také konstrukce senzorů, pracujících na principu odrazu – v tomto případě je zdroj i detektor na stejné straně a nad nimi se otáčí disk s proužky o dvou různých odrazivostech (zpravidla stačí bílý a černý nátěr).

Společný pro obě provedení je typ výstupu – tedy impulsy šířkou odpovídající dírkám v cloně nebo reflexním proužkům. Pro zjištění rychlosti otáčení tedy stačí čítat impulsy přicházející ze senzoru po dobu nějaké vzorkovací periody a zjištěný počet podělit počtem děr clony (počtem reflexních proužků). Získáme tak počet otáček za jednu periodu vzorkování, který není těžké převést například na úhlovou rychlost v radiánech za sekundu.

Stejně jako všechny dále zmíněné diskrétní senzory trpí optické senzory jednou nectností – v oblasti nízkých otáček a vysokých vzorkovacích frekvencí poskytují poměrně špatné výsledky. Nepříjemně se zde projevuje kvantování informace o otočení sledované hřídele do diskrétních pulsů. Při nízkém rozlišení clony a vysoké vzorkovací frekvenci se může dokonce stát, že bude navzorkováno nula impulsů za periodu při nenulové rychlosti otáčení. Tento problém se minimalizuje dostatečným rozlišením clony, běžně kolem 500 impulsů na otáčku.

3.1.3. Hallova sonda

V podobném uspořádání jako průchozí optický senzor obvykle pracuje i senzor otáček s Hallovou sondou – s tím rozdílem, že není použit zdroj světla, ale permanentní magnet. Clona pak musí být kovová a její funkce spočívá ve zkratování magnetického toku, který tak ovlivňuje Hallovu sondu jen v místech otvorů.

Tvar signálu a jeho zpracování je prakticky stejný jako u optických senzorů.

3.1.4. Indukčnostní snímače

Snímače toho typu sestávají z cívky (indukčnosti), která je součástí oscilátoru. Cívka je zpravidla navinuta na feritovém jádře, které pomáhá směrovat jí generované elektromagnetické pole do oblasti, ve které se pohybuje kovová clona. Jejím přiblížením do oblasti cívky se oscilátor zatíží tvorbou vířivých proudů v kovovém materiálu clony a změní se tak pracovní bod oscilátoru. Tuto změnu detekuje spínací obvod, který patřičně upraví stav výstupu.

Signálem se tak tato skupina snímačů opět neliší od optických a Hallových senzorů. Na rozdíl (především) od optických snímačů je však těžké dosáhnout vyššího rozlišení (počtu impulsů na otáčku).

3.1.5. Indukční snímače

Podobně jako u indukčnostního snímače je hlavním prvkem senzoru cívka, která však má jádro z permanentního magnetu. Otáčením kovové clony s výřezy v její blízkosti se na ní indukuje napětí úměrné rychlosti otáčení clony. Toto napětí však není vyhlazené, je impulsního charakteru s počtem impulsů na otáčku odpovídajícím počtu výřezů v cloně.

Nevýhodou je právě závislost výstupního napětí na rychlosti otáčení clony, snímač je tak obtížně použitelný v oblasti nízkých otáček, kdy je obtížné rozeznat jednotlivé pulsy.

Jinak se signál charakterem příliš neliší od dříve zmíněných diskrétních snímačů.

3.2. Výkonová část

Řídit výkon dodávaný stejnosměrnému spotřebiči (v našem případě DC motoru) ze stejnosměrného zdroje (akumulátorů) je možné několika způsoby. Principiálně je možné je rozdělit do dvou kategorií:

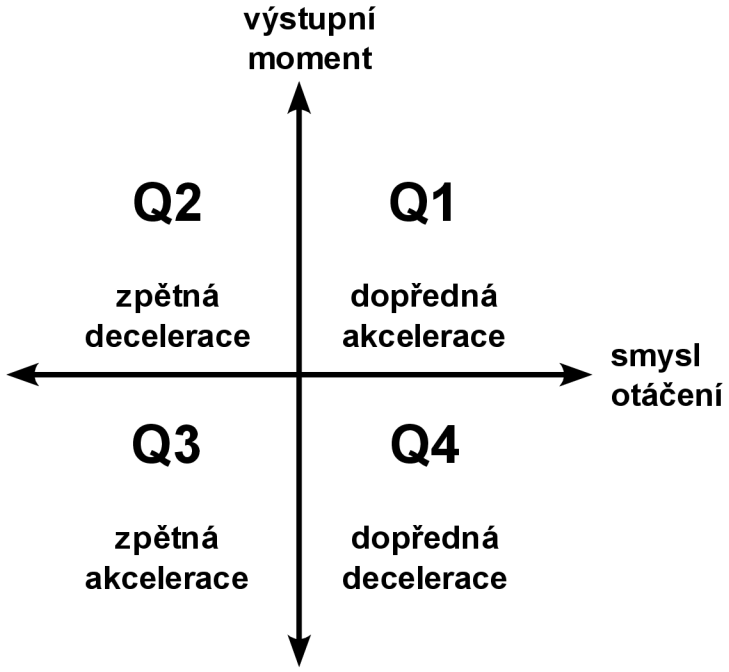
- ztrátové řízení – výkonový stupeň tvoří napěťový dělič, na němž zůstává část napětí zdroje a napětí na zátěži je tak příslušně menší. Zdroj tak dodává prakticky konstantní výkon, jehož určitá část (podle žádaného výkonu na zátěži) je disipována na výkonovém stupni. Napětí na spotřebiči je regulováno spojitě, ale za cenu plýtvání energií a velkých nároků na chlazení výkonových prvků.
- „bezztrátové“ řízení – výkonové prvky pracují ve spínacím režimu, kdy na nich vzniká minimální úbytek napětí[2]. Pro řízení výkonu se pak použije některá z pulsních modulací, nejčastěji pulsně-šířková (PWM).

Vzhledem k tomu, že při použití ztrátového řízení motoru o výkonu 250 W daném zadáním bych musel vyřešit odvod značného množství Jouleova tepla disipovaného na výkonovém stupni, připadá v úvahu pouze řízení bezztrátové.

Dalším kritériem dělení metod řízení je schopnost řídit spotřebič v různém počtu kvadrantů, které jsou zachyceny na obrázku 3.1. Jednotlivými případy se zabývám v následujících podkapitolách.

Velmi dobrým zdrojem v oblasti návrhu výkonové elektroniky regulátoru je pramen [4].

2 pochopitelně nikdy nulový – proto je označení tohoto řízení v uvozovkách. Výkon disipovaný na výkonovém stupni je však výrazně menší než v případě ztrátového lineárního řízení.



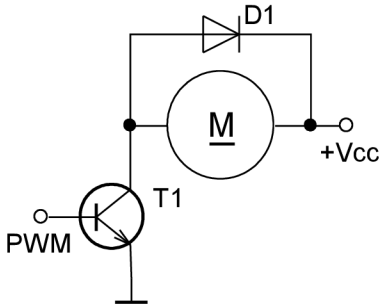
Ilustrace 3.1: Kvadranty řízení motoru

3.2.1. Jednokvadrantové řízení

Jednokvadrantový budič dokáže motor řídit pouze v jednom směru, nedokáže motor brzdit.

Jak je zřejmé z obrázku 3.2, budič pracuje prakticky pouze jako elektronický „vypínač“, který přerušuje a uzavírá obvod zahrnující oba póly napájecího zdroje a motor.

Vzhledem k tomu, že v zadání je regulátor specifikován jako obousměrný, nelze čistě jednokvadrantové řízení použít.



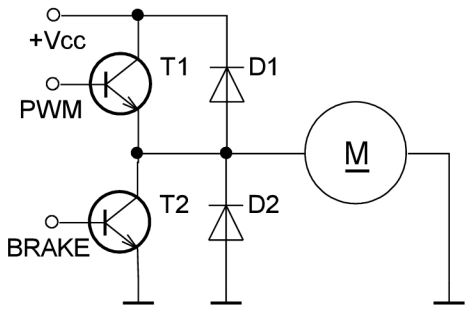
Ilustrace 3.2: Princip 1Q budiče

Dioda D1 je přítomna z důvodu ochrany tranzistoru T1 před proudovými špičkami vznikajícími při uzavření tranzistoru. Dioda umožní průchod těchto proudů zpět motorem a zamezuje tak průrazu tranzistoru.

3.2.2. Dvoukvadrantové řízení

Zapojení jednokvadrantového budiče doplněné o druhý spínač zajišťující brzdění (viz obrázek 3.3) je obvykle nazýváno dvoukvadrantovým budičem, zvládá totiž motor řídit v kvadrantech Q1 a Q4.

Stejně tak je za dvoukvadrantový budič možné považovat jednokvadrantové zapojení doplněné o relé přepínající polaritu napětí na svorkách motoru. Řízenými kvadranty jsou pak Q1 a Q3.



Ilustrace 3.3: Princip 2Q budiče

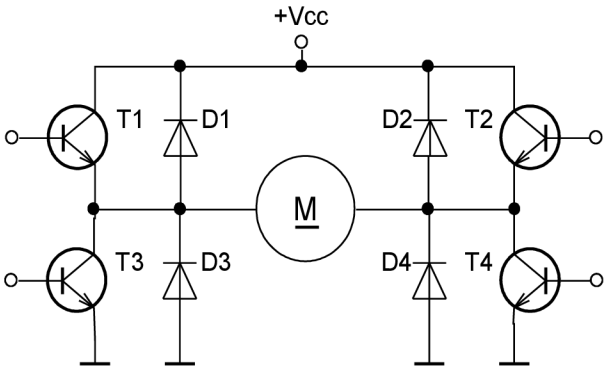
Dvoukvadrantový budič schopný řízení motoru v kvadrantech Q1 a Q3 by požadavkům zadání vyhověl. Jeho realizace s pomocí elektromechanického prvku (relé) však není příliš vhodná

z důvodu, že přechodový odpor kontaktů relé by se časem a působením nepříznivých vlivů pravděpodobně zvětšoval a zhoršoval tak parametry regulátoru, o jiskření při spínání proudů běžných při napájení výkonného DC motoru nemluvě. I tento typ budiče jsem tedy zavrhl.

3.2.3. Čtyřkvadrantové řízení

Řízením motoru ve všech jeho režimech vyniká čtyřkvadrantový budič. Jeho jiným a obvyklejším názvem souvisejícím s tvarem uspořádání je tzv. H-můstek.

Za čtyřkvadrantový budič by se dalo považovat i zapojení z obrázku 3.3 doplněné o relé přepínající polaritu napětí na svorkách motoru, ale z důvodů zmíněných v předchozí podkapitole nebudu toto zapojení uvažovat.



Ilustrace 3.4: Princip 4Q budiče

Řízení H-můstku není už tak snadné jako v případě méněkvadrantových budičů, protože je třeba sepnout vždy dva spínače. Přináší to ale výhodu možnosti čistě elektronicky obrátit polaritu napětí na svorkách motoru. Brzdné schopnosti jsou přitom zachovány.

Abychom docílili chodu motoru v jednom/druhém směru, je třeba sepnout dvojici tranzistorů T1+T4/T2+T3. Jedna svorka motoru se tak připojí ke kladnému pólu zdroje a druhá k zápornému.

Existují tři typy brzdění stejnosměrného motoru (viz pramen [5]), přičemž všech tří je možné za pomoci čtyřkvadrantového budiče dosáhnout:

- *generátorické brzdění* nastane v případě, že otáčky motoru v důsledku vnějšího zatížení přesáhnou otáčky na prázdno a indukované napětí na kotvě motoru přesáhne napětí svorkové. Směr proudu se obrátí a na zátěž začne působit brzdný moment – motor se stává generátorem (dynamem).
- *dynamické brzdění* – na svorky motoru je místo napájecího zdroje připojen odpor a motor tak pracuje opět v režimu generátoru do něj. V případě impulsně řízeného H-můstku je situace specifická v tom, že připojovaný odpor má nulovou velikost – svorky se sepnutím tranzistorů T1+T2 nebo T3+T4 zkratují. Míru brzdění je pak možno regulovat střídou řídicího PWM signálu (případně parametrem jiné impulsní modulace).
- *brzdění protiproudem* – nejúčinnější metoda, kdy je na svorkách motoru obrácena polarita napájecího napětí a motor se tak snaží obrátit smysl otáčení svého rotoru. Z momentové charakteristiky DC motoru vyplývá extrémní velikost výstupního momentu, z čehož vyplývá také extrémní velikost protékajícího proudu. Je třeba proto výkonové prvky dimenzovat i na takovouto zátěž (naštěstí brzdná zátěž je málokdy trvalého charakteru, jde tedy spíše o jednodušeji zvladatelné špičkové zatížení).

Pro realizaci výkonového stupně svého regulátoru jsem vybral čtyřkvadrantové zapojení budiče, a to především z důvodu jeho velké univerzality. Postrádá také nespolehlivé a pomalé elektromechanické prvky. Díky impulsnímu řízení tranzistorů ve spínacím režimu je zaručena vysoká proudová zatížitelnost můstku při nepatrných nárocích na chlazení.

3.3. Řídicí jednotka – platforma

Řídicí jednotkou je v tomto textu myšlen centrální mikrokontrolér a jeho podpůrné obvody, které ovšem nespadají do ostatních subsystémů regulátoru (výkonový stupeň, IRC dekodér).

Úvodem bych vysvětlil termíny „mikroprocesor“, „mikrokontrolér“ a „signálový procesor“, se kterými v dalším textu často operuji. *Mikroprocesor* je zařízení integrující většinu funkcí CPU do jednoho čipu. *Mikrokontrolér* je typem mikroprocesoru, který navíc integruje i typické periferie – paměť programu, operační paměť, I/O vývody, A/D převodníky a další speciální rozhraní. *Signálový procesor* je pak mikroprocesor nebo mikrokontrolér specializovaný na oblast real-time zpracování digitalizovaného analogového signálu.

Nároky, které na řídicí mikrokontrolér klade konstrukce relativně jednoduchého regulátoru otáček, splňují jak „obyčejné“ osmibitové typy, tak i (s rezervou) signálové procesory. Jejich výhodou je fakt, že disponují velkým výkonem a instrukcemi speciálně určenými pro real-time práci se signálem. Není tak technologicky a výkonově problém implementovat některý z digitálních filtrů měřené veličiny (např. proudu) a potlačit tak šum v signálu.

Uvědomuji si, že v této části tak trochu porovnávám neporovnatelné – detailně se zabývám dvěma rodinami, které ovšem patří každá do jiné kategorie mikroprocesorů. Je tomu tak proto, že obě jsou v oblasti regulační techniky často používány, i když každá z jiných důvodů. Obě platformy také postoupily do užšího výběru z rozsáhlé množiny moderních mikroprocesorů (namátkou uveďme např. PIC od Microchipu, všemožné klony Intel 8051 nebo oblíbené Freescale HC(S)08).

Vítězem výběru se stala rodina mikrokontrolérů Atmel AVR, především z důvodu vývojového prostředí integrujícího open-source kompilátor, velkého množství dostupných informací a široké komunity vývojářů. Navíc existuje mnoho otevřených a mnohdy i nenáročných konstrukcí programátorů podporujících tuto rodinu.

3.3.1. Atmel AVR

Jedná se o rodinu osmibitových mikrokontrolérů Harvardské architektury (oddělená paměť programu a dat). CPU těchto procesorů je typu RISC s instrukcemi prováděnými během jednoho cyklu a dosahuje tak výkonu 1 MIPS na 1 MHz taktovací frekvence. Více viz např. pramen [6].

Výhody:

- bezproblémová dostupnost běžných typů, nízká cena
- existence kvalitního open-source kompilátoru *avr-gcc* a standardní knihovny *avr-libc*
- „obrovská“ komunita uživatelů a autorů knihoven; není problém najít knihovnu pro obsluhu prakticky libovolné části hardware mikrokontroléru
- široká paleta různě vybavených typů – je možné vybrat mikrokontrolér poměrně přesně na míru aplikaci
- dostupnost některých typů také v provedení DIL (výrazně ulehčuje vývoj při použití nepájivého kontaktního pole)
- mnoho konstrukcí programátorů od nejjednodušších pro paralelní port po komplexní USB řešení

Nevýhody:

- „pouze“ 8bitová architektura – nižší výkon zejména při aritmetických operacích s vícebytovými čísly a při operacích v plovoucí desetinné čárce (floating-point)
- „obyčejné“ časovače – mnohem menší možnosti nastavení, než u DSP56800

3.3.2. Freescale DSP56800

DSP56800 je rodinou šestnáctibitových signálových procesorů od firmy Freescale (dříve Motorola). Jsou konstruovány jako kombinace mikrokontroléru a signálového procesoru a spojují tak výhody obou přístupů – vysoký výkon potřebný pro zpracování zpětnovazebních signálů z čidel a velké množství snadno použitelných periferií.

Výhody:

- časovače optimalizované pro použití při přímém řízení výkonového můstku a přímé čítaní kvadraturních impulsů z IRC
- velký výpočetní výkon (16bitová architektura, instrukce kategorie DSP, podpora plovoucí desetinné čárky) – možnost implementace náročných filtrů vstupních veličin

Nevýhody:

- špatná dostupnost v obchodech (částečně vyváženo možností nechat si poslat vzorky)
- nedostupnost neplaceného vývojového prostředí (IDE CodeWarrior je zdarma do velikosti přeloženého kódu 32 KB, což zvláště při použití knihoven nemusí stačit)
- z předchozího bodu vyplývá menší dostupnost volných zdrojových kódů a knihoven (i když některé obecnější je možno díky jazyku C portovat z jiné architektury)
- nedostupnost volné konstrukce programátoru

3.4. Komunikační sběrnice

V úvodu jsem nastínil, že cílem mého vývoje bylo zkonstruovat regulátor otáček motoru pro kolový mobilní robot *Bender 2*. Otázka volby sběrnice pro komunikaci s ostatními zařízeními proto byla řešena celým týmem podílejícím se na konstrukci robotu.

Zvolena byla nakonec sběrnice RS 485, zejména pro svou odolnost vůči rušení a nenáročnost implementace. Je však třeba podotknout, že pokud by se v budoucnu objevila potřeba regulátor přizpůsobit jiné komunikační technologii, neměl by to být problém vzhledem k hardwarové podpoře nejběžnějších rozhraní řídicím mikroprocesorem.

V tomto stručném popisu vlastností několika standardů, které byly vzhledem k zaměření projektu relevantní, se zaměřím především na vítěze výběru, sběrnici RS 485.

3.4.1. I^2C

Jak již z názvu vyplývá, Inter Integrated Circuit (I^2C) je sběrnice určená primárně pro komunikaci integrovaných obvodů (např. v rámci jedné desky plošných spojů). Její přednosti spočívají v jednoduché implementaci, malé hardwarové náročnosti (není třeba žádný budič)

a velké podpoře ze strany výrobců mikroprocesorů[3] a ostatního hardware.

Naopak není příliš odolná vůči rušení, což byl také jeden z hlavních argumentů proti použití tohoto standardu v našem robotu.

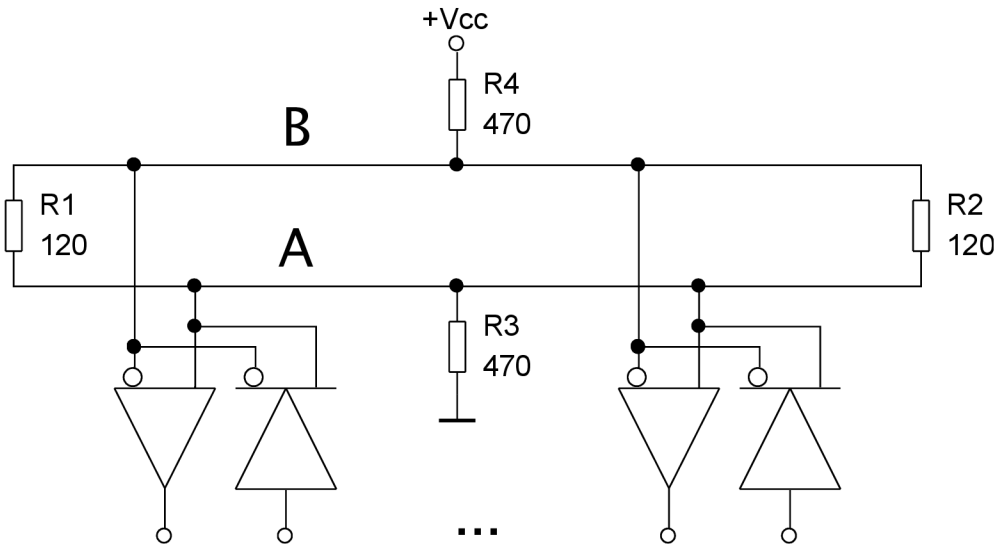
Sběrnice využívá dvou vodičů pro přenos signálů SDA (datový vodič) a SCL (hodinový signál udávaný masterem), zemní vodič a napájecí napětí přivedené přes pull-up rezistory pro definici klidového stavu vodičů. Standardními rychlostmi komunikace je 100 kHz a 400 kHz, ale problémem by neměla být ani komunikace na 1 MHz. Typickou je na sběrnici připojen jeden master, který generuje hodinový signál a řídí komunikaci, a několik slave přístrojů; možný je však také multi-master režim.

Zařazením a vlastnostmi existuje několik podobných standardů – např. SPI (Serial Peripheral Interface) nebo Dallas 1-Wire.

Více viz např. [7].

3.4.2. RS 485

Standard EIA-485, známý také jako RS 485, je poloduplexní vícebodová sériová komunikační sběrnice. Vyznačuje se diferenciálním napěťovým přenosem, kdy logické stavy jsou definovány polaritou rozdílu napětí na datových vodičích. Tyto jsou označeny A a B, přičemž signál B je negací signálu A. V klidu je tedy signál B kladnější než signál A.



Ilustrace 3.5: Schematické zapojení linky RS-485

Velmi důležitou záležitostí zejména v případě dlouhých linek je terminace a definování klidového stavu sběrnice. Používají se terminační rezistory na obou koncích sběrnice (R1, R2) a pull-up/pull-down rezistory (R3, R4). EIA-485 dále předepisuje vstupní impedanci budiče 12 kΩ, což umožňuje na jednu linku připojit až 32 přístrojů (viz [8]).

Na rozdíl od známějšího sériového rozhraní RS 232 je RS 485 použitelné i na delší vzdálenosti (až 1200 m) a díky diferenciálnímu principu přenosu je mnohem odolnější vůči naindukovanému rušení (rušivé napětí se naindukuje na oba vodiče stejně, čímž se rozdíl napětí

3 Z licenčních důvodů se sběrnice I^2C někdy skrývá i pod jinými názvy – např. Atmel označuje mírně rozšířenou verzi tohoto standardu jako TWI – Two Wire Interface.

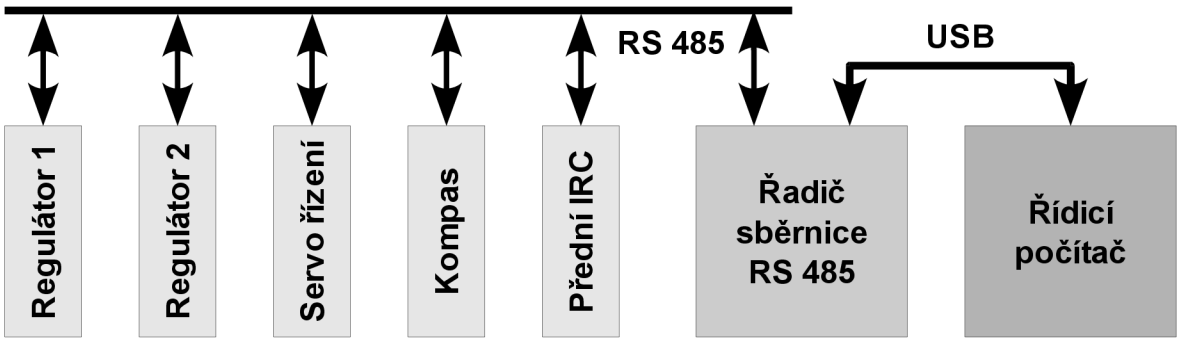
nezmění). Rozdílový vstup také do značné míry eliminuje rozdíl zemních potenciálů na straně jednotlivých zařízení. Pro naše použití je však důležitá zejména odolnost vůči rušení, protože vzdálenosti jsou v rámci robotu do půl metru a zemní potenciál je společný všem zařízením.

Maximální rychlost komunikace uváděná normou se pohybuje kolem hodnoty 10 Mbit/s.

Pro obsluhu sběrnice RS 485 existuje několik typů sériově vyráběných integrovaných budičů, které v jednom pouzdře integrují obvody diferenciálního vysílače i přijímače. Dle normy EIA-485 také zajišťují nadproudovou ochranu. V nabídce je má mnoho výrobců, z nejznámějších a nejpoužívanějších jmenujme MAXIM MAX485, Texas Instruments 75176 nebo DS3695A od National Semiconductors.

3.4.2.1. Komunikační protokol

Norma EIA-485 nijak neupravuje komunikační protokol, který pro komunikaci po sběrnici mají zařízení používat. Je proto na každém, aby se popasoval s vlastním návrhem protokolu, případně zvolil některé z již existujících řešení. Jedním z takovýchto protokolů je ROBIN – ROBot Independent Network ([9]). Jeho koncepce je velmi podobná té, kterou jsme zvolili my.



Ilustrace 3.6: Topologie sběrnice RS 485 robotu Bender 2

Základem, na kterém stojí návrh našeho komunikačního protokolu, je master/slave[4] komunikace s jedním řídicím zařízením a několika podřízenými jednotkami. Master je řídicí počítač, resp. jím ovládaný řadič sběrnice, slave potom všechna ostatní palubní elektronika (viz obrázek 3.6).

Každé zařízení slave na sběrnici má svou unikátní adresu. Řadič sběrnice coby master adresu mít nemusí, neboť žádné jiné zařízení nesmí začít samo od sebe vysílat, ale pro případnou budoucí potřebu je pro něj vyhrazena adresa 0x00. Dále je definována broadcast[5] adresa 0xFF.

Paket dat vyslaný řadičem sběrnice potom vypadá následovně:

```
<adresa> <příkaz> <počet datových bytů N> <data1> … <dataN> <CRC>
```

a paket odpovědi:

```
<adresa> <počet datových bytů N> <data1> … <dataN> <CRC>
```

Počet datových bytů je omezen jen rychlostí jejich zpracování.

4 Komunikace typu master/slave je založena na jednom přístroji, který řídí veškerý provoz na sběrnici, a jednom či více zařízeních podřízených. Podřízené zařízení nesmí zahájit komunikaci, není-li k tomu vyzváno.

5 Broadcast je vysílání zařízení master, které je určeno pro všechna ostatní zařízení.

3.4.3. CAN-bus

Písmena CAN jsou zkratkou slov Controller Area Network ([10]) a označují sériovou datovou sběrnici vyvinutou firmami Bosch a Intel v osmdesátých a devadesátých letech 20. století (práce na sběrnici však pokračují dále a na trh jsou uváděny jeho různé mutace, např. TTCAN).

Fyzická vrstva sběrnice je tvořena vlastním přenosovým médiem (nejčastěji diferenciální linka dle standardu ISO 11898-2), CAN kontrolérem a budičem sběrnice (může být i součástí kontroléru). Jsou definovány dva rychlostní profily – při vzdálenostech do 40 m je maximální rychlost sběrnice 1 Mbit/s nebo 125 Kbit/s.

Komunikace probíhá prostřednictvím rámců sestavených ze skupin bitů vyjadřujících různé části přenášené informace (identifikátor zprávy, vlastní data, CRC, ACK atd.). Existují čtyři typy rámců: *data frame*, *remote frame*, *error frame* a *overload frame*. Podle délky identifikátoru se dělí na základní (identifikátor dlouhý 11 bitů) a rozšířené (29 bitů).

CAN je známý vysokou spolehlivostí přenosu a indikací několika různých přenosových chyb. Mezi algoritmy používané pro tento účel patří:

- *CRC* – cyklická kontrola přenášených datových bytů
- *ACK* – nastavování příznaku úspěšného přijetí rámce
- *Bit stuffing* – při výskytu pěti bitů stejné polarity je za ně vložen bit opačné polarity; při příjmu šesti bitů stejné polarity je hlášena chyba
- vyšší algoritmy kontroly vycházející z významu jednotlivých částí zprávy

Protokolů pro sběrnici CAN byla vyvinuta různými firmami již řada (mezi jinými například DeviceNet, SDS nebo SafetyBUS p), v poslední době se však čím dál více prosazuje protokol CANopen definovaný organizací CiA (CAN in Automation).

Díky svým dobrým vlastnostem a rozsáhlé standardizaci je sběrnice CAN používána v širokém spektru aplikací, zahrnujícím automobilový průmysl (CAN-bus byl vyvinut jako automobilová komunikační sběrnice) a průmyslovou automatizaci (jako fieldbus).

4. Konstrukce

Hardware regulátoru je možno rozdělit na několik logických konstrukčních celků. Každý takovýto celek je možné navrhnout a realizovat mnoha způsoby, z nichž některé (pro mé zadání nejrelevantnější) byly popsány v rešerši, kde jsem také uvedl, kterou z popisovaných technologií jsem vybral pro použití v tomto projektu.

Tato kapitola se bude věnovat upřesnění faktů uvedených v rešerši každého z konstrukčních celků a samotné konkrétní realizaci celého regulátoru.

4.1. Řídicí část

4.1.1. Volba platformy

V rešeršní části zabývající se popisem použitelných mikroprocesorových platforem jsem uvedl, že zvolena byla v souladu s požadavky rodina Atmel AVR. Úkolem této kapitoly bude popsat nároky na centrální mikroprocesor, zajišťující vlastní regulaci otáček motoru i komunikaci regulátoru s okolím.

Primárním požadavkem byl počet šestnáctibitových časovačů/čítačů – pro řízení vzorkovací periody o velikosti řádově desítek až stovek milisekund a čítaní impulsů z IRC je kapacita osmibitového akumulátoru nedostatečná. Hned toto první kritérium z výběru diskvalifikovalo podstatnou většinu zařízení rodiny ATmega. Z dostupných typů mu vyhověla ATmega162, ATmega64 a ATmega128 (což je verze ATmega64 disponující větší pamětí).

Vzhledem k tomu, že jsem původně měl v plánu zkonstruovat regulátor jako kaskádní s podřízenou smyčkou regulace proudu, byl A/D převodník další potřebou periferií. Díky tomuto požadavku (a také díky absenci HW podpory rozhraní I^2C/TWI, které bylo stále ve hře coby komunikační rozhraní) vypadl z výběru typ ATmega162.

Zbylé nároky splnila ATmega64 naprosto s přehledem – disponuje dvěma dalšími osmibitovými časovači (z nichž jeden je použit pro generování PWM signálu), dvěma rozhraními USART, jedním TWI, jedním SPI, 10bitovým A/D převodníkem atd.

Výběr řídicího mikrokontroléru tak byl nakonec paradoxně velmi snadný, mé požadavky přesně splňoval jediný dostupný obvod.

4.1.2. Komunikace s okolím

Hlavními požadavky na komunikační sběrnici robotu *Bender 2* byla spolehlivost, odolnost vůči rušení a jednoduchost implementace. Těmto nárokům nejlépe odpovídá rozhraní RS 485, které bylo také nakonec zvoleno.

Ideální topologie této sběrnice je lineární, tedy jde o jedno elektrické vedení, na kterém jsou „navěšeny“ jednotlivé přístroje. V případě našeho robotu tento účel plní kroucená dvoulinka UTP. V případě problémů s rušením by se dal použít i stíněný twist, ale k problémům tohoto typu zatím nedošlo, zůstali jsme tedy u UTP.

Našim potřebám nejlépe odpovídala architektura single master – multi slave, tedy architektura s jednou centrální jednotkou, která řídí provoz na sběrnici, a více podřízenými zařízeními. Je

tomu tak proto, že veškerou inteligenci a autonomii robotu zajišťuje řídicí počítač (notebook nebo embedded systém) a ostatní zařízení slouží pouze jako senzory a aktuátory bez autonomní inteligence. Počítač sběrnici obsluhuje za pomoci řadiče zvlášť vyvinutého pro naše potřeby. Regulátor je tedy zařízením typu slave – podřízeným.

Workflow komunikace v rámci robotu pak vypadá následovně: počítač například zjistí, že potřebuje znát údaj z palubního digitálního kompasu. Prostřednictvím sběrnice USB se tedy spojí s řadičem sběrnice, kterému předá přesné znění příkazu. Řadič tento příkaz zapíše na jím řízenou palubní sběrnici a čeká na odpověď od zařízení, kterému je příkaz adresován. Jakmile odpověď obdrží, předá ji opět přes USB nadřízenému počítači. Jinak řečeno, jde o half-duplex.

Mikroprocesory Atmel AVR hardwarově podporují sériovou komunikaci zabudovaným rozhraním UART/USART. Mnou zvolený mikrokontrolér ATmega64 obsahuje dva moduly USART, z nichž jeden jsem použil. Přístup k těmto rozhraním je řízen plně pomocí přerušení, jejich obsluha tedy prakticky nezatěžuje výpočetní kapacitu mikrokontroléru a o veškeré časování je postaráno hardwarově.

Pro připojení ke sběrnici standardu RS 422/485 je nejvýhodnější použít integrovaný budič (např. 75176 nebo MAX485), který převádí diferenciální napěťové úrovně sběrnice na běžné logické signály. Oba zmíněné budiče mají stejné rozložení vývodů a jsou tak vzájemně plně nahraditelné (v případě, že jeden typ není dočasně k dostání, je možné bez problémů použít ten druhý).

Budič sběrnice je připojen k pinům RXD a TXD rozhraní USART0 (vývody portu E PE0 a PE1), přepínání směru komunikace je zajištěno vývodem PB7.

4.1.3. Firmware

Celé programové vybavení regulátoru je psáno v jazyce C za použití verze *avr-gcc* open-source kompilátoru *gcc* a standardní knihovny *avr-libc*. Výhodou tohoto řešení oproti vývoji v assembleru nebo přímo strojovém kódu je nejen velká rychlost vývoje software a přehlednost napsaného kódu (!), ale i dostupnost velkého množství knihoven zapouzdřujících přístup k periferiím mikrokontroléru, což vývoj dále zjednodušuje.

Pro vývoj veškerého firmware jsem použil IDE *AVR Studio* dodávané přímo firmou Atmel. Bezproblémově integruje jak překladač assembleru, tak právě open-source kompilátor *avr-gcc* a knihovny *avr-libc*. Podporuje také připojení několika typů programátorů, mimo jiné i typu *AVRISP mkII*. Klon tohoto USB programátoru (*USBprog* – [11]) jsem také použil pro nahrávání přeloženého kódu do mikrokontrolérů.

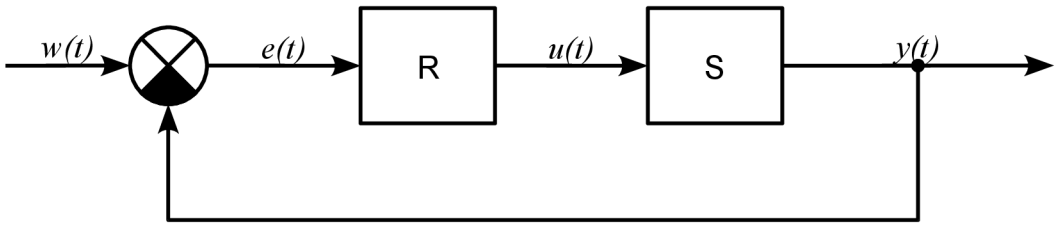
Během ladění se ukázal být velkou pomocí LCD 16x2 znaků, který jsem připojil na port F centrálního mikrokontroléru. Obsluhuje ho knihovna *LCD Library* od Petera Fleuryho ([12]).

4.1.3.1. Rychlostní regulátor

Regulace otáček připojeného motoru probíhá v tradiční zpětnovazební smyčce, kdy na vstup regulátoru R je přiváděna regulační odchylka $e(t)$ coby rozdíl řídicí veličiny $w(t)$ a regulované veličiny $y(t)$ (viz obrázek 4.1 a pramen [13]):

$$e(t) = w(t) - y(t)$$

Regulační odchylka je pak zpracována algoritmem regulátoru (nejčastěji nějakou variantou číslicového PSD regulátoru) a je stanovena hodnota akční veličiny $u(t)$ ovlivňující soustavu S.



Ilustrace 4.1: Schéma obecné regulační úlohy

Použitím modelu stejnosměrného motoru, jehož parametry (indukčnost a odpor vinutí, moment setrvačnosti rotoru atd.) jsem změřil nebo přibližně spočítal, jsem v programu Matlab za použití metody optimálního modulu navrhl přenos PI regulátoru vhodného pro řízení použitého motoru. Tento přenos jsem posléze použil jako počáteční nastavení konstant diskrétního PS regulátoru implementovaného v řídicím mikrokontroléru. Konstanty jsem empiricky doladil tak, aby měl motor s regulátorem co nejlepší odezvu na skok řízení. Použitý diskrétní regulátor nyní popíšu podrobněji.

Nejdříve jsou definovány konstanty a parametry regulátoru:

```
// constants
#define SAMPLE_PERIOD_R 0.1   // rotation
#define IRC_TICKS 20          // holes per disc
#define IRC_MODE 4            // ticks per one hole

// rotation PS
#define COEF_P_R 0.02
#define COEF_S_R 0.3
```

Konstanta `SAMPLE_PERIOD_R` určuje vzorkovací periodu rychlostního regulátoru v sekundách, konstanty `IRC_TICKS` a `IRC_MODE` popisují použitý inkrementální snímač – jeho rozlišení a režim (více viz kapitola 4.2.1. Dekódování signálu IRC).

Dále je třeba definovat globální proměnné:

```
// global variables
volatile double setup_rpm = 0;     // rpm set by master
volatile double sampled_rpm = 0;   // rpm from IRC
volatile int64_t sampled_irc = 0;  // IRC, a backup to the main front IRC
volatile unsigned char rotation_dir = 0;  // rotation direction from IRC

// rotation PS
volatile int64_t dev_suma = 0;     // control deviation suma
```

Proměnná `setup_rpm` uchovává rychlost nastavenou po komunikační sběrnici (tedy řídicí veličinu), do proměnné `sampled_rpm` je ukládána okamžitá rychlost otáčení hřídele motoru (regulovaná veličina). Pro účely měření ujeté vzdálenosti je možné použít obsah proměnné `sampled_irc`, tedy čítače navzorkovaných impulsů z IRC. V proměnné `rotation_dir` je aktuální směr otáčení hřídele motoru a `dev_suma` uchovává sumaci regulační odchylky.

Regulační algoritmus je periodicky spouštěn s periodou danou konstantou `SAMPLE_PERIOD_R` za pomoci Timer/Counter3 – jeho Compare Interruptem.

Prvním krokem je úprava žádané hodnoty – při umístění regulátoru na levou stranu robotu musíme reverzovat otáčky motoru. Volba strany, na kterou je regulátor umístěn, je provedena definicí makra `SIDE_LEFT`, o volbě strany je tedy rozhodnuto již v preprocesoru kompilátoru.

Dále je zjištěno. zda není regulátor zastaven vnějším příznakem STOP, který je přiveden na pin číslo 4 portu E:

```
if (!bit_is_set(PINE, PE4))
```

Ve STOP-stavu je proměnná dev_suma nastavena na nulu, čímž se regulátor prakticky uvede do stavu před nastavením nenulové řídicí veličiny. Zabrání to „škubnutí“ motoru při návratu do provozního stavu.

Pokud STOP-stav nenastal, pokračuje se načtením impulsů z IRC (popsáno níže v kapitole 4.1.3.2. Komunikační subsystém) a výpočtem okamžité rychlosti otáčení na hřídeli motoru:

```
sampled_rpm = pow(-1, rotation_dir) * sampled * 60 / SAMPLE_PERIOD_R /
     IRC_TICKS / IRC_MODE;
```

Jde o programové vyjádření vztahu, který převádí navzorkovaný počet impulsů sampled na otáčky za minutu (pro nastavování rychlosti otáčení motorů jsme se rozhodli používat vyjádření v otáčkách za minutu z důvodu větší názornosti tohoto údaje):

$$\text{sampled\_rpm} = (-1)^{\text{rotation\_dir}} \cdot \frac{60}{\text{SAMPLE\_PERIOD\_R}} \cdot \frac{sampled}{\text{IRC\_TICKS} \cdot \text{IRC\_MODE}}$$

V této fázi jsou připraveny všechny veličiny vstupující do regulátoru, přichází tedy na řadu samotný výpočet regulačního zásahu (akční veličiny) pwm_ratio:

```
// the main rotation control algorithm...
// PS-regulator
deviation = real_rpm - sampled_rpm; // e(kT)
dev_suma += deviation * SAMPLE_PERIOD_R;

if (dev_suma > INT64_MAX - 10000) dev_suma = INT64_MAX - 10000;
if (dev_suma < INT64_MIN + 10000) dev_suma = INT64_MIN + 10000;

pwm_ratio = floor(COEF_P_R * deviation + COEF_P_R * SAMPLE_PERIOD_R /
     COEF_S_R * dev_suma); // PS-algorithm

if (pwm_ratio > 255) pwm_ratio = 255; // PWM ratio normalization...
if (pwm_ratio < -255) pwm_ratio = -255;
```

Proměnnou deviation je označena regulační odchylka. Z té je následně vypočten přírůstek sumace (resp. numerické integrace – je používána jednoduchá obdélníková metoda) přičtený ke globální proměnné dev_suma. Ta je ošetřena proti přetečení (i když jde jen o formální záležitost – přetečení 64-bitového integeru je za běžných okolností nepravděpodobné; test přetečení bude pravděpodobně nahrazen anti-windup opatřením) a použita ve výpočtu regulačního zásahu pwm_ratio podle algoritmu odpovídajícího vztahu

$$u(k) = K_P \left(e(k) + \frac{T}{T_S} \sum_{i=1}^{k} e(k) \right)$$

Hodnota regulačního zásahu je omezena na interval <-255; 255>, neboť rozlišení PWM modulátoru je jeden byte. Jeho absolutní hodnota (která vyjadřuje amplitudu regulačního zásahu) je po této úpravě zapsána do registru OCR0, který určuje střídu výsledného pulsně-šířkově modulovaného signálu:

```
OCR0 = abs(pwm_ratio); // update the timer
```

Znaménko akčního zásahu určuje směr a je podle něj nastaven pin 1 portu C:

```
if (pwm_ratio < 0) // choose right output direction
{
     PORTC |= _BV(PC1);
}
else
{
     PORTC &= ~(_BV(PC1));
}
```

Zmíněná proměnná `rotation_dir` uchovávající informaci o směru otáčení hřídele motoru je aktualizována při změně na pinu 5 portu E:

```
// External interrupt 5 (rotation direction change)
ISR(INT5_vect)
{
     if (PINE & _BV(PE5))
     {
          rotation_dir = 1;
     }
     else
     {
          rotation_dir = 0;
     }
} // ISR(INT5_vect)
```

Kompletní zdrojový kód pro centrální mikroprocesor i dekodér signálu IRC najdete na přiloženém CD.

4.1.3.2. Komunikační subsystém

Komunikace regulátoru po lince RS 485 s řadičem sběrnice probíhá za pomocí upravené knihovny UART Library od Petera Fleuryho ([14]). Originál této knihovny je založen na Application Note AVR306 firmy Atmel ([15]) a podporuje mnoho obvodů řady ATmega, ATtiny i AT90s.

Úprava spočívala v přidání podpory MPCM, výstupního pinu pro přepínání směru komunikace (což je při komunikaci po RS 485 nutné) a několika pomocných funkcí. Originální i upravenou knihovnu naleznete na přiloženém CD, její popis by byl zřejmě nad rámec zpracovávaného tématu.

Obsluha zabudovaného rozhraní USART mikroprocesoru se díky použití knihovny a jazyka C stává velice přehlednou a snadnou záležitostí. Na začátku programu je třeba vložit samotnou knihovnu a definovat parametry komunikace:

```
#include "uart.h"

#define UART_BAUD_RATE 9600

#ifdef SIDE_LEFT
     #define UART_MY_ADDRESS 0x31
#else
     #define UART_MY_ADDRESS 0x30
#endif

#define UART_BROADCAST_ADDRESS 0xFF
```

Konstanta UART_BAUD_RATE specifikuje komunikační rychlost v baudech za sekundu (USART umožňuje při použití krystalu 14,7456 MHz rychlost až 1,8432 Mbit/s). V závislosti na poloze regulátoru (pravé/levé kolo) je pak stanovena adresa UART_MY_ADDRESS a nakonec adresa broadcastu společná pro celou sběrnici UART_BROADCAST_ADDRESS.

V inicializační rutině je knihovna nastavena (zvolená rychlost, 9 bitů ve slově, bez paritního bitu a povoleno MPCM):

```
// init the USART0 - RS485: 9bit, no parity, MPCM
uart_init(UART_BAUD_SELECT(UART_BAUD_RATE, F_CPU), 9, 0, 1);
```

V této chvíli je možné knihovnu začít používat. Komunikaci řídí nekonečná smyčka ve funkci main(). Při každé iteraci se přečte jeden znak ze sběrnice:

```
c = uart_getc();
```

V případě, že buffer dat je prázdný, inkrementuje se čítač komunikačního „watchdogu“:

```
if (c & UART_NO_DATA)
{
    rst++;
}
```

V opačném případě je nejprve provedena kontrola, zda nedošlo při přijetí nebo zpracování daného slova k chybě:

```
if (c & UART_FRAME_ERROR || c & UART_OVERRUN_ERROR || c &
    UART_BUFFER_OVERFLOW)
{
    uart_mpcm_on(); // set MPCM

    i = 0;
    command = 0;
    order = 0;

    continue;
}
```

Pokud ano, je resetován stav komunikace (proměnné i, command a order) a program skočí do další iterace nekonečné smyčky.

Je-li vše v pořádku, smaže se čítač watchdogu a pokračuje se zpracováním přijatých dat:

```
rst = 0; // clear the "watchdog"

switch (order)
{
    case 0:
        if ((unsigned char) c == UART_MY_ADDRESS ||
            (unsigned char) c == UART_BROADCAST_ADDRESS)
        {
            uart_mpcm_off(); // clear MPCM

            order++;
        }

        break;
```

```
        case 1:
             command = (unsigned char) c;
             order++;

             break;
        case 2:
             data_num = (unsigned char) c;

             if (data_num > 0)
             {
                  order++;
             }
             else
             {
                  order += 2;
             }

             break;
        case 3:
             order++;

             break;
   }
```

Proměnná `order` udržuje počet dosud přijatých znaků a určuje tak význam právě přijatého znaku (viz popis protokolu v kapitole 3.4.2.1. Komunikační protokol; jedinou odchylkou je, že zatím nebyla implementována kontrola paketu bytem CRC). Podle toho je pak znak:

- porovnán s vlastní a broadcastovou adresou (shoduje-li se, je inkrementována proměnná `order` a pokračuje se tak v příjmu dat)
- přiřazen do proměnné `command` a určuje tak identifikátor instrukce, která má být provedena
- přiřazen do proměnné `data_num` a jeho významem je počet datových znaků, které budou následovat

Je-li komunikace ve fázi příjmu datových znaků, jsou tyto zapsány do datového bufferu inicializovaného na maximální počet očekávaných znaků `DATA_BUFF_SIZE`. Je také inkrementována proměnná `i`, která udržuje počet doposud přijatých znaků:

```
   if (order > 3 && data_num > 0 && i < data_num && i < DATA_BUFF_SIZE)
   {
        data[i] = (unsigned char) c;

        i++;
   }
```

Z podmínky vykonání kódu uložení datových znaků je zřejmé, že k uložení nedojde, je-li překročena kapacita bufferu – algoritmus je tedy chráněn proti přetečení.

V případě, že je příjem dokončen:

```
   if (order > 3 && (i == data_num || i == DATA_BUFF_SIZE))
```

je provedena samotná instrukce identifikovaná číslem uloženým v proměnné `command`. Aktuální verze firmware podporuje tři příkazy:

- `0x01` – nastavení otáček. Program očekává dva datové znaky obsahující 16b číslo – žádanou hodnotu výstupních otáček motoru. V případě úspěšného nastavení odpoví

regulátor řadiči zasláním své adresy a nuly coby počtu datových znaků.

- `0x10` – čtení čítače impulsů z IRC čidla. Program neočekává žádný datový znak. Odpověď obsahuje vlastní adresu, dvojku (počet znaků odpovědi) a dva znaky se 16b reprezentací počtu navzorkovaných impulsů.
- `0xFF` – reset nastavení regulátoru (nastavení nulových otáček). Program nic neočekává ani nevrací.

Kód obsluhy jednotlivých instrukcí naleznete také na přiloženém CD.

4.2. Měření otáček motoru

4.2.1. Dekódování signálu IRC

Vzhledem k volbě rodiny Atmel AVR coby platformy řídicí jednotky bylo nutné vyřešit vzorkování velikosti otáček na hřídeli motoru. Tyto mikroprocesory totiž neobsahují zabudovaný dekodér kvadraturního signálu (na rozdíl například od DSP firmy Freescale), pouze vstup externích hodin časovače. Signál vstupující do procesoru tedy musí být charakteru impulsů odpovídajících impulsům signálu IRC a samostatné informace o směru otáčení.

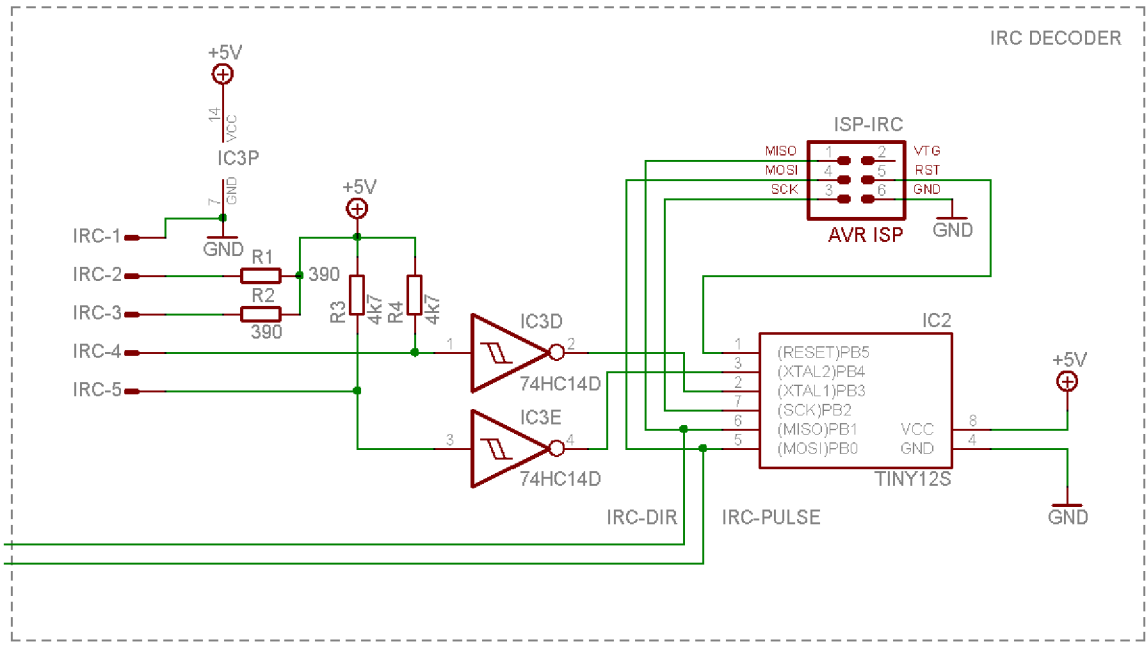
Možností, jak toto realizovat, je několik – zmíním alespoň obvody LS7083 a LS7084 popsané v [16], které jsem na počátku vývoje plánoval využít. Jejich nasazení však ztroskotalo na vysoké ceně a obtížné dostupnosti v České republice.

Nakonec se jako nejprůchodnější ukázala varianta vlastní výroby takovéhoto dekodéru. Vzhledem k tomu, že nároky na maximální frekvenci dekódovatelných signálů nejsou až tak vysoké (pokud budeme uvažovat jako mezní otáčky např. 10 000 ot/min, bude při použití enkodéru s 500 impulsy na otáčku nejvyšší zpracovávanou frekvencí vstupních impulsů cca 330 kHz), bylo možné využít i nejmenší a nejlevnější 8-pinové verze procesorů rodiny ATtiny. Prakticky jediným nárokem (který se však ukázal být dosti omezujícím) tak byla přítomnost paměti typu SRAM v mikrokontroléru, protože právě ji využívá kompilátor *avr-gcc* pro definici proměnných a pro zařízení, která jí nejsou vybavena, není schopen program v jazyce C zkompilovat[6]. Vybraným mikrokontrolérem se stal typ ATtiny13 (ve schématu je uveden z důvodu nepřítomnosti ATtiny13 v knihovně pinově kompatibilní typ ATtiny12).

Enkodér vyrobený pro použití s původními motory vybranými pro robot *Bender 2* byl schopen dodat pouze 20 impulsů na otáčku. Rozhodl jsem se proto, že dekodér bude pracovat v režimu X4, tzn. že bude reagovat výstupním impulsem na všechny změny kvadraturního signálu, čímž de facto čtyřikrát znásobí rozlišení enkodéru. Z hlediska programování algoritmu je to mimo jiné také nejjednodušší. Při použití enkodéru s větším rozlišením je třeba pouze zkontrolovat, zda nedojde k překročení mezní vstupní frekvence čítače hlavního mikrokontroléru (viz kapitola 4.2.2. Vzorkování řídicím mikroprocesorem).

Enkodér je k regulátoru připojen přes pětipinový konektor, který zároveň zaručuje napájení LED enkodéru. Signál je nejprve tvarován pomocí Schmittova invertoru. Na vstupu invertoru je také definována klidová úroveň signálu tak, že při nepřipojeném enkodéru nedochází

6 Toto tvrzení není zcela pravda, existuje způsob, jak toto omezení obejít, ale nebylo v mém zájmu se tím zabývat. Také by vzhledem k jednoduchosti algoritmu bylo možné program napsat v assembleru, což by umožnilo použít jakýkoliv typ z rodiny ATtiny, jednodušším se ale ukázalo vybrat a sehnat vhodný typ mikrokontroléru.



Ilustrace 4.2: Schéma tvarovače a dekodéru kvadraturního signálu

k pronikání šumu na vstup dekodéru. Tvarovaný signál je již přiveden na vstupy PB3 a PB4 dekodéru. Výstup je na pinech PB0 a PB1.

Otázka generování výstupního impulsu byla vyřešena velmi jednoduše: tím, že na výstup dekodéru navazuje vstup čítače hlavního procesoru regulátoru, není nutné dbát na přesné časování impulsu. Podle datasheetu mikrokontroléru má být frekvence externích hodin minimálně 2,5 krát menší než frekvence system clocku, což dává procesoru určitý čas na detekování impulsu. Teoreticky by tedy stačily výstupní impulsy o šířce cca 3 periody system clocku hlavního procesoru. Dekodér to řeší jednoduše – na začátku zpracování externího přerušení nastaví pin výstupního portu na úroveň 1 a na konci zpracování na úroveň 0, čímž vytvoří impuls o délce zaručeně větší, než je požadované minimum, a zároveň ne tak velké, aby vadila při vyšších frekvencích výstupních impulsů (aby docházelo ke „slévání“ impulsů do sebe).

Prakticky celou funkčnost dekodéru realizuje následující kód:

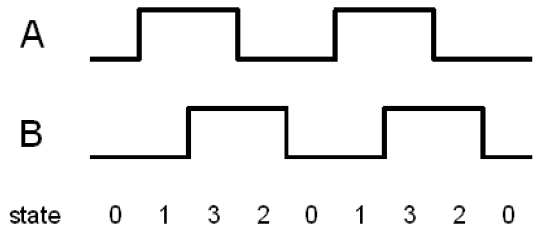
```
// pin change interrupt handling routine
ISR(PCINT0_vect)
{
     char state = 0;

     // generate an impuls
     OUTPUT_PORT |= _BV(PULSE);

     // fill the state variable
     if (bit_is_set(INPUT_PORT, INPUT_A))
     {
          state |= _BV(0);
     }

     if (bit_is_set(INPUT_PORT, INPUT_B))
     {
          state |= _BV(1);
     }
```

```
        // decide the direction
        if ((last_state == 0 && state == 1) ||
            (last_state == 1 && state == 3) ||
            (last_state == 3 && state == 2) ||
            (last_state == 2 && state == 0))
        {
            // we're moving forward, clear the output...
            OUTPUT_PORT &= ~(_BV(DIRECTION));
        }
        else
        {
            // we're moving backward, set the output...
            OUTPUT_PORT |= _BV(DIRECTION);
        }

        // save this state
        last_state = state;

        // end of the impuls
        OUTPUT_PORT &= ~(_BV(PULSE));
    }
```

Proměnná `state` obsahuje detekovaný stav vstupů z IRC enkodéru. Jsou čtyři možné kombinace:

- oba vstupy v 0 – `state = 0`
- vstup A v 1, vstup B v 0 – `state = 1`
- vstup A v 0, vstup B v 1 – `state = 2`
- oba vstupy v 1 – `state = 3`,

což lze také vyjádřit jako 2-bitové slovo BA.



Ilustrace 4.3: Stavy složek IRC signálu

Podle posloupnosti stavu předchozího a současného se pak určí směr pohybu[7] a zapíše na pin `DIRECTION` portu `OUTPUT_PORT`, což jsou ve skutečnosti makra preprocesoru definovaná na začátku zdrojového souboru.

Zbytek zdrojového kódu pro mikrokontrolér dekodéru obsahuje pouze příkazy pro inicializaci periferií mikrokontroléru a definici proměnných a maker. V hlavní smyčce programu se nic neprovádí; v případě potřeby snížení příkonu mikrokontroléru by bylo možné vřadit na toto místo příkazy pro uvedení do některého z režimů spánku. Probuzení by zajišťovala přímo změna na některém ze vstupních pinů.

4.2.2. Vzorkování řídicím mikroprocesorem

Načítání impulsů z dekodéru IRC hlavním procesorem regulátoru je řešeno (jak již bylo zmíněno) vstupem hodin pro 16-bitový Timer/Counter1. Inicializaci časovače zajišťuje následující kód:

```
    // timer1: external clk, rising edge
    TCCR1B = _BV(CS12) | _BV(CS11) | _BV(CS10);
```

Jde v podstatě jen o výběr zdroje hodinového signálu časovače, všechna ostatní nastavení jsou ponechána jako výchozí.

7 Zde je vhodné doplnit, že dekodér de facto o skutečném směru otáčení nemá ani potuchy, pouze přiřazuje jedné sekvenci vstupních impulsů logickou nulu na směrovém výstupu a opačné sekvenci logickou jedničku...

Šestnáctibitový čítač byl zvolen s ohledem na větší možnost volby vzorkovací periody: při použití pouze osmibitového čítače by bylo nutné vyčítat jeho obsah nejpozději při 255. načteném impulsu, což by při uvažované mezní frekvenci vstupních impulsů 330 kHz určovalo maximální vzorkovací periodu $7{,}7 \cdot 10^{-4}\,\mathrm{s}$, která je hluboko pod rozumným rozsahem volby vzorkovací periody. Znamenalo by to také, že při použití enkodéru s nízkým rozlišením (což byl i případ původního pohonu našeho robotu) v kombinaci s extrémně malou periodou vzorkování by měření rychlosti otáčení hřídele motoru vykazovalo zoufale nízkou a nevyhovující přesnost.

Samotné vyčítání navzorkovaných impulsů řeší následující kód:

```
// restart the counter
sreg = SREG;
cli();
TCNT3 = 0;
sampled = TCNT1; // read sampled ticks
TCNT1 = 0;
SREG = sreg;

sampled_irc += sampled;
if (sampled_irc > INT64_MAX - 10000) sampled_irc = INT64_MAX - 10000;
```

Vzhledem k tomu, že architektura AVR je osmibitová, zatímco akumulátor čítače šestnáctibitový, provádí se kopírování obsahu akumulátoru ve dvou strojových cyklech. Existuje zde ovšem nebezpečí, že mezi vyčtením jednotlivých bytů proběhne přerušení, během kterého se hodnota uložená v akumulátoru může změnit. Výsledkem by pak byla chybná hodnota přečtená z akumulátoru. Přímo v datasheetu ATmega64 ([17]) je proto doporučován postup, kdy je do proměnné `sreg` nejprve uložen obsah status registru `SREG`, následně jsou příkazem `cli()` zakázána přerušení a teprve poté je bezpečné číst šestnáctibitový registr. V našem případě je nejdříve restartován Timer/Counter3, který určuje vzorkovací periodu, pak je z registru `TCNT1` přečten počet navzorkovaných impulsů z IRC a tento registr je také smazán. Nyní je už možné přerušení opět povolit, a to nahráním původního obsahu registru `SREG`.

Počet impulsů za aktuální vzorkovací periodu uložený v proměnné `sampled` je přičten do 64bitové celočíselné proměnné `sampled_irc`, která shromažďuje navzorkovaný počet impulsů až do jejich vyčtení po komunikační sběrnici. Kód na posledním řádku zabraňuje přetečení proměnné `sampled_irc`.

4.3. Výkonový stupeň

Pro napájení našeho robotu byla zvolena dvojice olověných gelových akumulátorů Banner GiVC 12-7,5 spojená do série. Při požadovaném maximálním příkonu motoru do 250 W a nominálním napětí dvojice baterií 24 V vychází maximální proud výkonovým stupněm přibližně v úrovni 10 A.

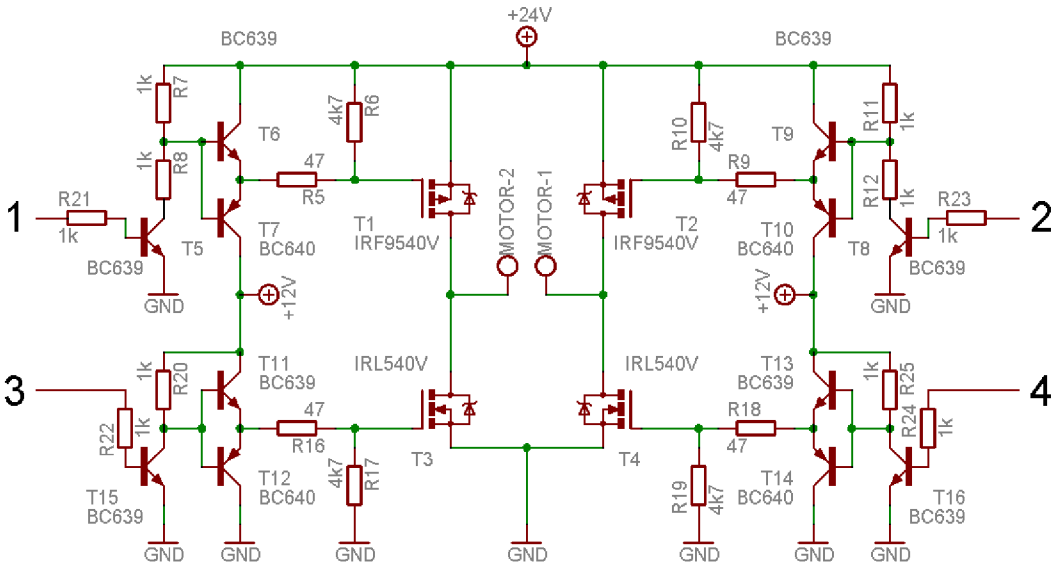
Jako architekturu výkonového stupně jsem zvolil H-můstek (čtyřkvadrantový budič). Je tvořen čtyřmi tranzistory typu MOSFET, přičemž horní část je vodivosti P a dolní část vodivosti N. Pro plné otevření potřebují tranzistory této technologie napětí U_{GS} větší než přibližně 10 V (viz [18] a [19]). Proudy při nabíjení a vybíjení náboje řádové velikosti desítek nanocoulombů na elektrodě G při rychlém přechodovém ději také mohou dosáhnout hodnot až v řádu jednotek ampér. Z těchto důvodů není možné tranzistor řídit přímo výstupem logického obvodu, je proto nutné použít pro řízení každého výkonového tranzistoru budič.

4.3.1. Budič FET

Většina výrobců polovodičů nabízí integrované budiče FET, někdy kombinované s nábojovou pumpou, která dostatečné řídicí napětí dokáže „vyrobit“ i z nižšího napájecího napětí. Bohužel, dostupnost takovýchto součástek na českém trhu v kusovém množství není velká a dovoz ze zahraničí není v takovýchto množstvích rentabilní. Rozhodl jsem se proto budič sestavit vlastní.

Moje konstrukce vychází z principiálního zapojení uvedeného na obrázku 4.7c v pramenu [20]. Sestává z dvojčinného emitorového sledovače tvořeného tranzistory T6 a T7 (T9, T10, T11, T12, T13, T14 v případě ostatních budičů) a rezistorů R5 a R6 (R9, R10, R16, R17, R18, R19) definujících nabíjecí/vybíjecí proud a klidový stav elektrody G (nesmí zůstat za žádných okolností „ve vzduchu“).

Tranzistor T5 (T8, T15, T16) spolu se zbývajícími rezistory pak slouží jako převodník mezi napěťovými úrovněmi CMOS logiky a vlastního budiče.



Ilustrace 4.4: Schéma výkonového stupně

4.3.2. Volba součástek

Pro realizaci H-můstku bylo třeba vybrat technologicky a parametricky podobné MOSFET tranzistory opačných vodivostí. Tento výběr jsem si zjednodušil, neboť jsem převzal typy použité v můstku regulátoru z pramenu [21]. K tomuto kroku jsem se rozhodl, protože jsem usoudil, že nejbezpečnější bude volba typů, jejichž koexistence je ověřená praxí. Jde o tranzistory IRF9540 (vodivost P, maximální proud 23 A, odpor v sepnutém stavu 0,117 Ω) a IRL540 (vodivost N, 36 A, 0,044 Ω).

Oba použité typy mají v pouzdře integrovánu antiparalelní diodu, není proto nutné ji přidávat v samostatném pouzdře.

V zapojení budiče jsou využity univerzální tranzistory, a to typy BC639 (NPN) a BC640 (PNP). Oproti běžnějším typům umožňují krátkodobé kolektorové proudy až 1 A, což je při konstrukci budiče výhodné až nutné.

4.4. Logika výkonového stupně

Navržený H-můstek výkonového stupně je třeba řídit tak, aby jednotlivé tranzistory spínaly ve vhodnou dobu vzhledem k momentálnímu směru otáčení a stavu PWM signálu. Tuto logiku ovládání tranzistorů je možné implementovat jak programově přímo v mikrokontroléru, tak i „natvrdo“ pomocí logických obvodů.

Každé řešení má své výhody a nevýhody – řešení softwarové spoří místo na plošném spoji i náklady na výrobu (což nás při kusové výrobě netrápí, jde o korunové položky), ale více využívá zdrojů mikroprocesoru (zabere více časovačů). Hardwarové řešení je sice řešením natrvalo a obvykle není možné nijak upravit jednou určenou logiku ovládání můstku, ale vystačí si na straně mikroprocesoru s jedním časovačem, který generuje PWM, a jedním výstupem směrovým.

Po určitém váhání jsem nakonec zvolil HW řešení, které nechává více zdrojů mikroprocesoru volných pro jiné použití. Navíc jsem implementoval funkci brzdy motoru, kdy dolní část výkonového můstku „zkratuje“ svorky motoru a ten tak přechází do režimu dynamického brzdění (viz [5]). Regulátor má opticky oddělený vstup, který je logickou funkcí OR připojen

PULSE	DIR	BRAKE	T1	T2	T3	T4
0	0	0	0	0	0	1
1	0	0	1	0	0	1
0	1	0	0	0	1	0
1	1	0	0	1	1	0
0	0	1	0	0	1	1
1	0	1	0	0	1	1
0	1	1	0	0	1	1
1	1	1	0	0	1	1

Tabulka 4.1: Pravdivostní tabulka logiky výkonového stupně

k ovládacímu výstupu brzdy z mikroprocesoru. Lze tak snadno robot v nouzovém případě zabrzdit, a to bez ohledu na momentální stav palubní elektroniky (tedy např. i při teoretickém hazardním stavu programu regulátoru). Jednotlivé signály jsem pojmenoval PULSE (PWM signál), DIR (informace o směru) a BRAKE (brzda).

Ovládací elektronika je postavena z obvodů řady 74HC (High speed CMOS), tedy rychlé nízkopříkonové CMOS logiky. Návrh spočíval v definici vstupů, výstupů a pravdivostní tabulky zachycující všechny možné stavy vstupů a reakce výstupů na ně. Výstupy jsem posléze vyjádřil ve tvaru logických funkcí závislých na vstupech a co nejvíce je zjednodušil. Minimalizaci jsem následně ověřil za pomoci Karnaughových map (viz např. [22]).

Pro výstupy T1 a T2 je minimalizace triviální – jsou aktivní pouze v jediném případě. Pro jejich realizaci byl tedy zvolen třívstupový člen AND.

I v případě výstupů T3 a T4 je minimalizace nenáročná, jejich aktivita je vázána pouze na signály DIR a BRAKE. Výstup T3 je v log. 1 v případě, že je v log. 1 vstup DIR nebo vstup BRAKE. T4 je v log. 1, jestliže je v log. 0 vstup DIR nebo je v log. 1 vstup BRAKE. Vzhledem

k tomu, že jsem se rozhodl logiku implementovat pomocí obvodů AND a NAND, bylo třeba provést převod dle de Morganových pravidel na logický součin.

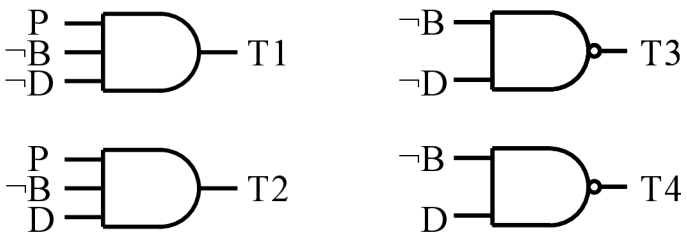
Jednotlivé logické funkce:

$$T1 = P \cdot \overline{B} \cdot \overline{D}$$

$$T2 = P \cdot \overline{B} \cdot D$$

$$T3 = B + D = \overline{\overline{B}} + \overline{\overline{D}} = \overline{\overline{B} \cdot \overline{D}}$$

$$T4 = B + \overline{D} = \overline{\overline{B}} + \overline{\overline{D}} = \overline{\overline{B} \cdot D}$$



Ilustrace 4.5: Zapojení hradel logiky výkonového stupně

Negované vstupní signály $\overline{B}$ a $\overline{D}$ jsou vytvořeny invertory z nenegovaných signálů B a D.

4.5. Prototyp

Vývoj regulátoru probíhal postupně od částí zajišťujících měření vstupních veličin po návrh akčních členů. Vznikl tak nejdříve otáčkoměr s dekodérem IRC signálu, následně výkonový stupeň a jeho řídicí logika. Nakonec bylo vše propojeno naprogramováním diskrétního regulátoru realizovaného v hlavním procesoru zařízení.

Na tomto prototypu, který díky použití nepájivého kontaktního pole nabyl velikosti takřka desetinásobku finálního produktu, bylo celé zapojení v reálu otestováno na částečně osazeném robotu řízené RC soupravou a poháněné pouze jedním motorem (měl jsem k dispozici pouze jeden prototyp regulátoru). Projevila se jediná závada, a to špatné mechanické upevnění silových vodičů k výkonovému stupni, díky kterému polovina výkonového stupně prototypu doslova vyhořela. Tato vada se na finálních výrobcích naštěstí neopakovala.

4.5.1. Řídicí jednotka

Prototyp řídicí jednotky byl postaven na nepájivém kontaktním poli. Tato technologie se ukázala být velice vhodnou pro vývoj, neboť je možné provádět libovolný počet změn v zapojení a vše řádně otestovat dříve, než se přikročí k výrobě plošného spoje (na němž je už případná větší úprava zapojení velice nepohodlná či přímo nemožná).

Jedinou nevýhodou se ukázalo být připojení mikrokontroléru, který je vyráběn pouze v pouzdrech pro povrchovou montáž. Vyřešil jsem to výrobou redukce TQFP64-DIL64, s pomocí které již nebyl problém mikrokontrolér zapojit.

4.5.2. Výkonová část

Vzhledem k očekávaným proudům, tekoucím výkonovou částí regulátoru, bylo nutné postavit prototyp této části mimo nepájivé kontaktní pole, které by takovou zátěž nejspíš neslo špatně. Realizoval jsem jej proto na kousku univerzálního plošného spoje pomocí drátových propojek.

4.6. Finální provedení regulátoru

4.6.1. Požadavky na mechanickou konstrukci

Hlavním požadavkem byla velikost přibližně 9,5 cm x 5 cm, daná potřebou umístit regulátory do prostoru mezi obě pohonné jednotky robotu. S přihlédnutím k počtu součástek, kterými měl být hardware regulátoru realizován, byla evidentní nutnost použití součástek pro povrchovou montáž (SMD). V současné době jsou však již klasické součástky na ústupu a technologie osazování SMD součástek je čím dál dostupnější i pro neprůmyslové a malosériové aplikace. Z tohoto pohledu přešla tato nutnost spíše ve vhodnou alternativu, jejíž použití konstrukci nezkomplikuje.

Podvozek robotu *Bender 2* neobsahuje žádné pružení či tlumení nárazů (pro vyrovnávání nerovností cesty je vybaven kyvnou zadní nápravou). Dalším požadavkem se proto stala přiměřená odolnost regulátorů proti otřesům a menším nárazům. I v této oblasti jsou součástky SMD vhodnější volbou, neboť mají mnohem menší rozměry a hmotnost než součástky klasické a trpí tím pádem při popisovaném namáhání znatelně méně.

4.6.2. Návrh DPS

I přes volbu SMT pro konstrukci finálního hardware regulátoru bylo nutné kvůli rozměrovým omezením navrhnout desku plošných spojů jako oboustrannou s prokovením. Vzhledem k tomu, že jsem se chtěl pokusit o výrobu svépomocí (DPS ostatních zařízení robotu jsem za pomoci modré nažehlovací fólie distribuované firmou GES-ELECTRONICS vyrobil), snažil jsem se počet prokovek minimalizovat. Nakonec se však deska stala nad očekávání složitou a domácí výrobu jsem po jednom nepříliš zdárném pokusu zavrhl – zhotovení bylo zadáno profesionálnímu výrobci.

Organizace součástek na desce je rozdělena na dvě části – řídicí a výkonovou. Zejména pod řídicí částí je na obou stranách desky vytvořena plocha „rozlité mědi“, která přispívá k odolnosti regulátoru vůči rušení (vzhledem k fyzickému umístění regulátorů mezi dva DC motory jde o poměrně důležitou vlastnost).

Celý port F centrálního mikrokontroléru je vyveden na kolíkovou lištu, která umožňuje připojení případných rozšiřujících zařízení bez úpravy DPS regulátoru. Pro účely ladění jsem přes tento port připojoval znakový LCD.

Navržená DPS regulátoru je k dispozici společně s kompletním schématem na CD přiloženém k této práci.

5. Reálné nasazení

Jak již bylo v celé práci mnohokrát zmíněno, popisovaný regulátor byl vyvinut pro použití v autonomním kolovém robotu *Bender 2* určeném pro soutěž Robotour 2007 (viz [23]). Hlavním cílem soutěže je zvládnout řízení a navigaci robotu tak, aby byl schopen projet cca kilometr dlouhou trasu sestavenou z parkových cestiček v pražské Stromovce. Ročník 2007, tedy druhý ročník soutěže, byl navíc obdařen přídomkem „outdoor delivery challenge" – byl tedy rozvinut požadavek na schopnost robotu uvézt užitečný náklad (etalonem se stal 5l soudek piva). Náš robot jsme konstruovali s přihlédnutím k tomuto faktu a tak se jeho hmotnost vyšplhala na necelých 20 kg.

Že není snadné levně pořídit kvalitní pohon robotu této hmotnostní kategorie, náš tým tušil již od počátku. Vinou nesprávně dimenzovaného převodového poměru mezi motory a hnanými koly a nepříliš robustní převodovce došlo velmi brzy (ještě před začátkem soutěže...) k enormnímu opotřebení silonových kol převodovek. Přičinily se na tom taktéž určité potíže s komunikační sběrnicí, kdy instrukci ke změně otáček dostal někdy pouze jeden z regulátorů, zatímco druhý se stále snažil vyregulovat předchozí žádanou hodnotu (obvykle nulové otáčky). Nekvalitní převody tak vzaly rychle za své. Naštěstí i s poškozenými převodovkami byl robot schopen soutěž částečně absolvovat a neskončit na posledním místě.

Regulace motorů však byla kvalitní a spolehlivá, jedinou výtku si zaslouží určitá pomalost regulátoru při reakci na výraznou změnu zatížení – použitý PS regulátor navržený pomocí metody optimálního modulu postrádá diferenciální složku výrazně ovlivňující rychlost regulace. Původní pohon navíc díky provozu v nevhodné oblasti momentové charakteristiky a malému převodovému poměru disponoval nevelkým přebytkem krouticího momentu. Tuto nectnost regulátoru odstraní implementace jiného číslicového regulátoru (PSD nebo S-PD).

Destrukcí převodovek vznikla nutnost pořídit pohon nový. Zvolena byla dvojice DC motorů RE 40 (výkon 150 W) s převodovkami GP 42 C (poměr 43:1) a enkodéry HEDS 5540 (500 impulsů na otáčku), vše od švýcarské firmy Maxon. V době dopisování této práce jsou v plném proudu úpravy mechanické části robotu nutné pro montáž nového pohonu. Regulátory budou novým motorům přizpůsobeny v nejbližší době – velké úpravy však třeba nebudou, stačí pozměnit příslušné konstanty a překompilovat firmware.

6. Závěr

Úkoly stanovené zadáním se mi podařilo v plné míře splnit – z velké škály dostupných řešení jsem vybral vhodné subsystémy, spojil je do jednoho celku a – a to je nejdůležitější – tento celek úspěšně v praxi otestoval.

Možností, jak navrženou konstrukci dále zdokonalit, je celá řada – pokud bych se tomuto tématu dále věnoval, pravděpodobně bych v první řadě realizoval v textu zmíněnou kaskádnost regulátoru. Znamenalo by to přidání snímače proudu tekoucího motorem, benefitem by byla jak kvalitnější regulace, tak zvýšená ochrana motoru i regulátoru vůči nadměrné zátěži. Další vylepšení by se mohla týkat komunikačních schopností regulátoru – ať už implementací některého ze standardních protokolů určených pro sběrnice RS 485, nebo přímo změnou rozhraní na podstatně komplexnější CAN-bus například za použití protokolu CANopen. Regulátor by tak bylo možno snadno připojit k profesionálním řídicím systémům.

Význam této práce pro mě nespočívá pouze v rovině splnění požadavku na vývoj regulátoru pro robot *Bender 2*. V průběhu vývoje jsem narazil na mnoho menších i větších problémů, které jsem musel překonat a regulátor „dotáhnout“ v co nejkratším čase do provozuschopného stavu. Získal jsem tak cenné zkušenosti jak v oboru návrhu a ladění mikroprocesorové a výkonové elektroniky, tak i v oblasti časově efektivní práce a týmové spolupráce.

7. Seznam použitých zdrojů

[1] Koláčný, J.. *Elektrické mikropohony*. Brno: UVEE FEKT VUT, ?. 173 s. ISBN nemá

[2] Různí autoři. *Rotary encoder*. http://en.wikipedia.org/wiki/Rotary_encoder

[3] Beneš, Havránek, Kopecký, Krupa. *Měření fyzikálních veličin - návody do laboratorních cvičení*. Brno: UAMT FEKT VUT, 2006. 102 s. ISBN nemá

[4] Simreal. *Custom Motor Driver*. http://www.simreal.com/mediawiki/index.php?title=CustomMotorDriver

[5] Ondrůšek, Č. *Elektromechanická přeměna energie*. Brno: UVEE FEKT VUT, ?. 60 s. ISBN nemá

[6] Atmel Corporation. *AVR® 8-Bit RISC*. http://www.atmel.com/products/avr/default.asp

[7] Různí autoři. *I2C*. http://cs.wikipedia.org/wiki/I%C2%B2C

[8] Staněk, J.. *RS 485 & 422*. http://hw.cz/Teorie-a-praxe/Dokumentace/ART821-RS-485-422.html

[9] BDMICRO. *Robin - ROBot Independent Network*. http://www.bdmicro.com/code/robin/

[10] Různí autoři. *Controller Area Network*. http://en.wikipedia.org/wiki/Controller_Area_Network

[11] Sauter, B.. *USBprog - Open-Source Universalwerkzeug*. http://www.embedded-projects.net/index.php?page_id=135

[12] Fleury, P.. *LCD Library*. http://homepage.hispeed.ch/peterfleury/group__pfleury__lcd.html

[13] Blaha, P., Vavřín, P.. *Řízení a regulace I*. Brno: UAMT FEKT VUT, 2006. 215 s. ISBN nemá

[14] Fleury, P.. *UART Library*. http://homepage.hispeed.ch/peterfleury/group__pfleury__uart.html

[15] Atmel Corporation. *AVR306: Using the AVR UART in C*. http://www.atmel.com/dyn/resources/prod_documents/doc1451.pdf

[16] Novák, P.. *Mobilní roboty - pohony, senzory, řízení*. Praha: BEN - technická literatura, 2005. 248 s. ISBN 80-7300-141-1

[17] Atmel COrporation. *ATmega64(L)*. http://www.atmel.com/dyn/resources/prod_documents/doc2490.pdf

[18] International Rectifier. *IRL540N*. www.irf.com/product-info/datasheets/data/irl540n.pdf

[19] International Rectifier. *IRF9540N*. www.irf.com/product-info/datasheets/data/irf9540n.pdf

[20] Patočka M., Vorel P.. *Řídicí elektronika - aktivní obvody*. Brno: UVEE FEKT VUT, 2004. 154 s. ISBN nemá

[21] Heatherington, D.. *Invertabot: 12 pound Combat Robot*. http://www.wa4dsy.net/robot/invertabot/

[22] Různí autoři. *Karnaugh map*. http://en.wikipedia.org/wiki/Karnaugh_map

[23] Dlouhý, M.. *Robotour 2007*. http://robotika.cz/competitions/robotour2007/cs

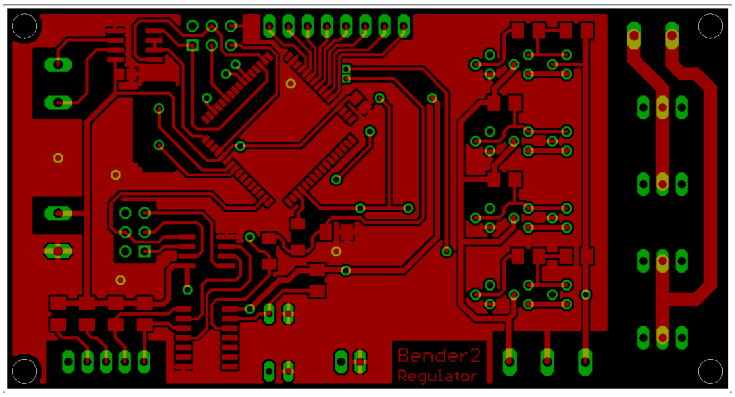
8. Seznam použitých zkratek a symbolů

AC	Alternating Current, střídavý proud
ACK	Acknowledge, příznak úspěšného přijetí dat
CPU	Central Processing Unit, procesor
CRC	Cyclic Redundancy Check, jeden z algoritmů kontroly integrity dat
DC	Direct Current, stejnosměrný proud
DPS	Deska plošných spojů
DSP	Digital Signal Processor, digitální signálový procesor
GCC	GNU Compiler Collection, open-source balík kompilátoru a dalších nástrojů pro jazyk C a C++. V našem kontextu je řeč zejména o větvi *avr-gcc* určené pro rodinu Atmel AVR.
IRC	Incremental Rotary Encoder, inkrementální snímač otáčení
MIPS	Millions of Instructions Per Second, miliony instrukcí za sekundu, jedno z měřítek rychlosti procesorů
MOSFET	Metal Oxide Semiconductor Field Effect Transistor – polem řízený tranzistor se strukturou kov-oxid-polovodič
MPCM	Multi-processor Communication Mode, rozlišování příchozích rámců na adresní a datové. Při nastaveném příznaku MPCMn v registru UCSRnA jsou ignorovány datové rámce a čeká se na příchod adresního rámce.
PCB	Printed Circuit Board, deska plošných spojů
PWM	Pulse-Width Modulation, pulsně-šířková modulace
RISC	Reduced Instruction Set Computer, počítač s redukovanou instrukční sadou
SMD	Surface Mount Device, součástka pro povrchovou montáž
SMT	Surface Mount Technology, technologie povrchové montáže
U(S)ART	Universal (Synchronous and) Asynchronous serial Receiver and Transmitter, (synchronní a) asynchronní transceiver pro všeobecné použití
UTP	Unshielded Twisted Pair, nestíněná kroucená dvoulinka

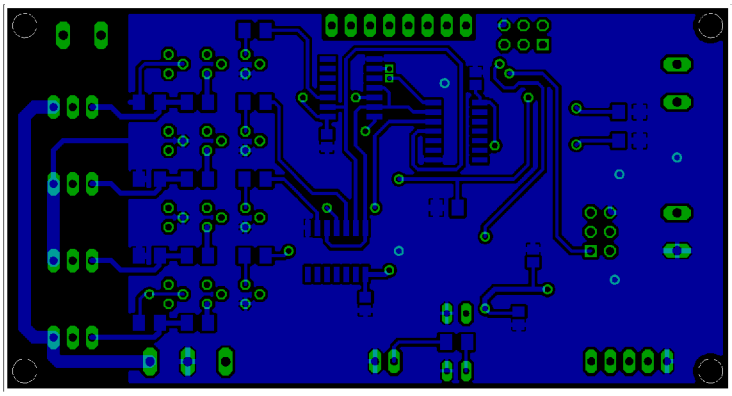
U_{GS}	napětí mezi elektrodami Gate a Source tranzistoru typu FET	[V]

9. Přílohy

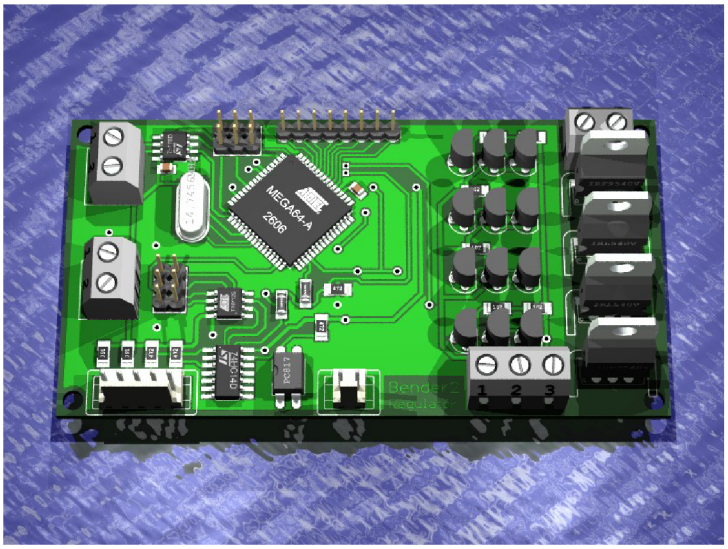
Příloha A: Návrh DPS



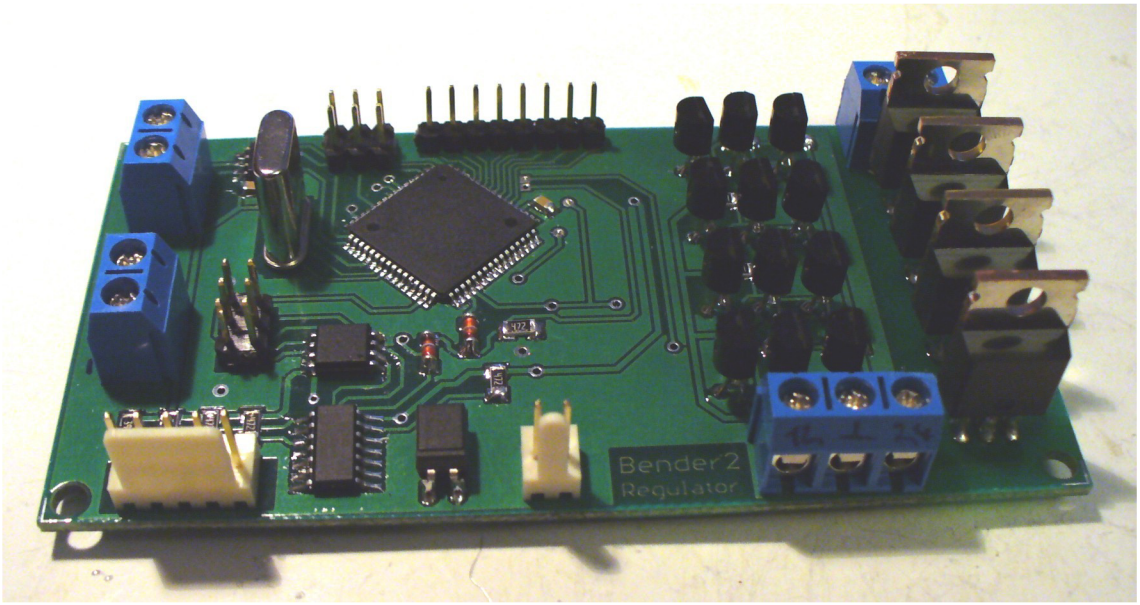
Ilustrace 9.1: Horní strana DPS regulátoru



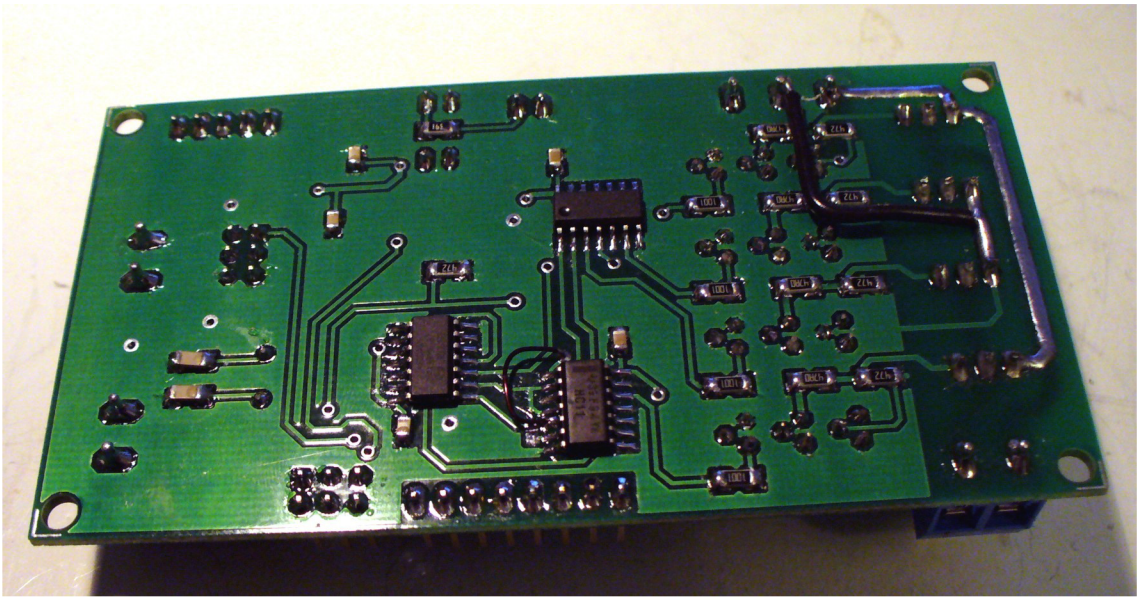
Ilustrace 9.2: Dolní strana DPS regulátoru



Ilustrace 9.3: 3D vizualizace DPS regulátoru



Ilustrace 9.4: Osazený regulátor – pohled shora



Ilustrace 9.5: Osazený regulátor - pohled zdola - jsou patrny drátové propojky

Příloha B: Software použitý při návrhu

- **AVR Studio 4** – vývoj firmware regulátoru pro mikrokontroléry Atmel AVR
- **Eagle 4.15 Light** – kreslení schématu zapojení, návrh DPS
- **Eagle3D a POV-Ray 3.6** – 3D simulace desky plošných spojů
- **Matlab R2006b** – tvorba modelu DC motoru a návrh konstant regulátoru
- **Micro-Cap 8.0** – simulace poloviny můstku včetně budičů FET, kontrola funkce
- **OpenOffice.org 2.4** – tvorba samotné bakalářské práce včetně kreslení diagramů
- **ProfiCAD 4.6.4** – kreslení schemat použitých v rešeršní části práce
- **WinAVR** – soubor knihoven a utilit pro podporu vývoje na platformě Atmel AVR