UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA

Modul řízení otáček ventilátoru

Pavel HLADÍK

BAKALÁŘSKÁ PRÁCE

2012

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel Hladík**
Osobní číslo:	**D08205**
Studijní program:	**B3709 Dopravní technologie a spoje**
Studijní obor:	**Dopravní infrastruktura: Elektrotechnická zařízení v dopravě**
Název tématu:	**Modul řízení otáček ventilátoru**
Zadávající katedra:	**Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě**

Zásady pro vypracování:

Cílem práce je tvorba modulu regulace otáček ventilátoru k dynamometru pro zatěžování spalovacích motorů. Modul bude zapojen do systému řízení dynamometru přes sběrnici CAN. Modul bude ovládat frekvenční měnič, ke kterému je připojen ventilátor chlazení měřeného objektu (motocykl či samotný motor).
Úkolem bakaláře bude:
-Nastudovat ovládání frekvenčního měniče
-Nastudovat sběrnici CAN
-Kompletně navrhnout HW modulu (vytvořit prototyp)
-Vytvořit firmware pro navrhnutý HW (v programovacím jazyku C)
-Ověřit funkci zařízení

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

HEROUT, Pavel. Učebnice jazyka C, Nakladatelství KOPP, 2004, 978-80-7232-383-8, IV. přepracované vydání.
BURKHARD, Mann. C pro mikrokontroléry. BEN Praha, Technická literatura, 2003, ISBN 80-7300-077-6.
VÁŇA, Vladimír. Amtel AVR programování v jazyce C - Popis a práce ve vývojovém prostředí CodeVisionAVR C, ISBN 80-7300-102-0.
HERAIN, Jaroslav. Softwarová nástavba frekvenčního měniče. Bakalářská práce UPCE DFJP KEEZ. 2006. D15665.
JAVŮREK, Jiří. Regulace moderních elektrických pohonů. Grada Publishing, 2003. ISBN 80-247-0507-9.
PAVELKA, Jiří a kol. Elektrické pohony. ČVUT Praha, 1996, ISBN 80-01-01411-8.
NOVÁK,Jaroslav. Elektromechanické systémy v dopravě a ve strojírenství. ČVUT Praha, 2002, ISBN 80-01-02457-1.

Vedoucí bakalářské práce: **Ing. Zdeněk Mašek**
Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě

Datum zadání bakalářské práce: **7. prosince 2011**
Termín odevzdání bakalářské práce: **31. května 2012**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Radovan Doleček, Ph.D.
vedoucí katedry

V Pardubicích dne 22. února 2012

Prohlášení:

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 2. 4. 2012 Pavel Hladík

Poděkování:

Nejdříve bych chtěl velice poděkovat vedoucímu této bakalářské práce, panu Ing. Zdeňku Maškovi za účinnou, metodickou, odbornou, věcnou i formální pomoc, a to po celou dobu tvorby této práce, kdy mi poskytoval mnohdy velmi cenné rady pro vývoj, testování a zpracování mé bakalářské práce. Tímto mu také děkuji za trpělivost a čas, který obětoval mým dotazům. Rovněž bych chtěl poděkovat dopravní fakultě Jana Pernera, zvláště pak Katedře elektrotechniky, elektroniky a zabezpečovací techniky v dopravě za poskytnutí prostor a mechanického zajištění, které bylo nutné pro tvorbu mé bakalářské práce. V neposlední řadě bych chtěl poděkovat rodině za podporu při mém mnohaletém studiu, dále spolužákům a kamarádům za cenné podněty a informace týkající se mé bakalářské práce.

Anotace:

Bakalářská práce se zabývá návrhem, tvorbou a testováním modulu regulace otáček ventilátoru k dynamometru pro zatěžování spalovacích motorů. Modul bude zapojen do systému řízení dynamometru přes sběrnici CAN. Modul bude ovládat frekvenční měnič, ke kterému je připojen ventilátor chlazení měřeného objektu (motocykl či samostatný motor).

Klíčová slova:

frekvenční měnič, dálkové ovládání, ventilátor, mikrokontrolér, D/A převodník, CAN

Title:

The Module of Control Fan Speed

Annotation:

This bachelor’s study focuses on design, creation and testing of a fan speed control module of a dynamometer for a loading internal combustion engines. The module be connected to the dynamometer of a control system through CAN bus. The module will be controlled by a frequency converter, which is connected with the cooling fan of the object (a motorcycle or a separate engine).

Keywords:

frequency convertor, remote control, fan, microcontroller, D/A converter, CAN

Obsah

Úvod

Vlastnictví silničního motorového vozidla je v posledních letech v naší společnosti téměř samozřejmostí, ať už se jedná o vlastnictví automobilu či motocyklu. Pro kvalitní údržbu a opravu motorového vozidla je nutností mít dostatek zařízení a měřících přístrojů na správnou diagnostiku a opravu samotného vozidla. Současný trend v automobilovém průmyslu spěje k veškeré elektronizaci, což nám mnohdy usnadní diagnostikování závad na vozidle.

Jednou ze záležitostí, s kterou se musíme při diagnostice vozidel vypořádat je měření výkonu silničního vozidla. Pro měření výkonu vozidla využíváme takzvaných dynamometrů, které jsou schopny změřit jak výkon vozidla, či samotného motoru, tak jeho krouticí moment. Při měření výkonu na dynamometru nám vyvstává problém s chlazením vozidla, jelikož se nám vozidlo nepohybuje, ale je připevněno k rámu dynamometru, tak nelze využít proudění vzduchu k chlazení měřeného vozidla, ke kterému dochází při jízdě. Automobily a motocykly jsou konstruovány tak, že využívají náporové chlazení motoru, kde se počítá s pohybem vozidla, při kterém bude okolo chlazených míst proudit vzduch. Účelem chlazení je udržet provozní teplotu motoru a odvést přebytečné teplo, které není využito k mechanické práci.

Tato bakalářská práce se zabývá právě návrhem a sestavením funkčního prototypu modulu řízení otáček ventilátoru, který bude vřazen do systému dynamometru. Modul řízení otáček ventilátoru nám bude ovládat frekvenční měnič, který bude následně ovládat motor ventilátoru pro chlazení vozidla. Ventilátor nám bude vytvářet námi požadované proudění vzduchu pro chlazení vozidla. Zdrojem požadovaných otáček ventilátoru je rychlost vozidla, vyčítaná pomocí sběrnice CAN, nebo dálkové ovládání, pomocí něhož lze nastavit rychlost otáček ventilátoru. Navržený a zkonstruovaný modul bude sloužit v laboratořích Katedry elektrotechniky elektroniky a zabezpečovací techniky spadající pod Dopravní fakultu Jana Pernera, jako jedna z komponent nutných pro provoz tamějšího dynamometru.

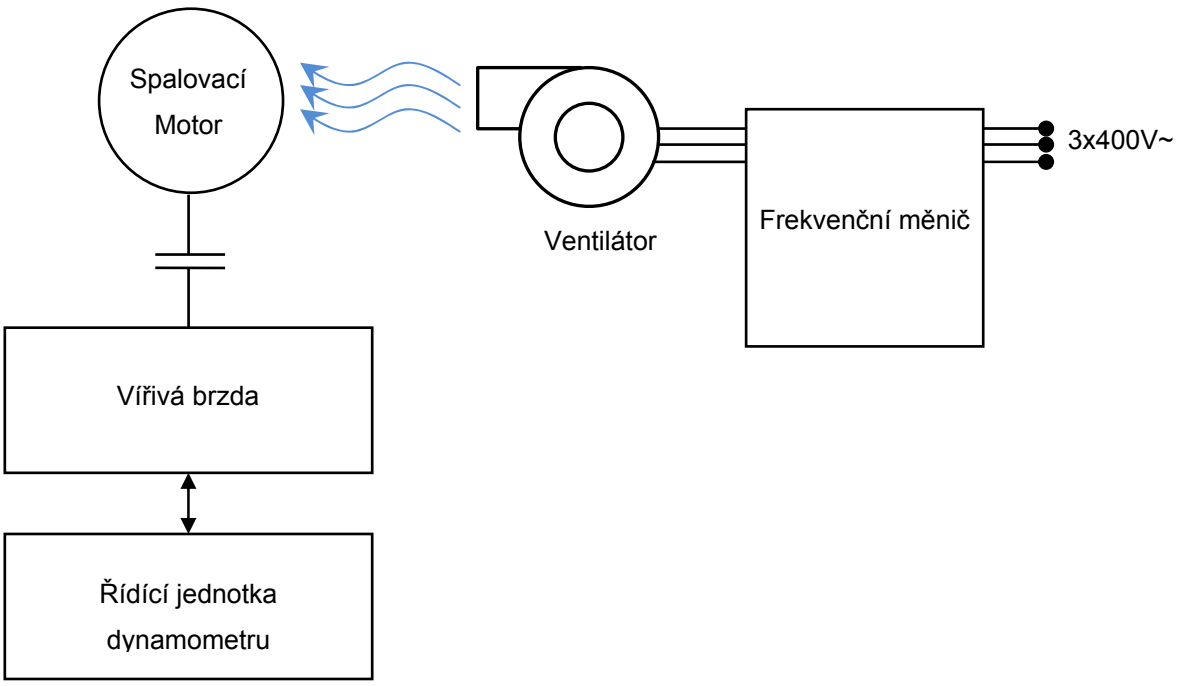
V mé bakalářské práci naleznete požadavky na modul řízení otáček ventilátoru a popis stávajícího stavu chlazení vozidel u dynamometru. Dále se práce věnuje hardwarovému řešení samotného modulu, kde se seznámíme s jednotlivými komponenty. Následně se práce zabývá tvorbou firmwaru pro modul řízení otáček ventilátoru, který je napsán v programovacím jazyku C. Poslední část práce se věnuje ověření funkčnosti modulu.

1 Specifikace modulu řízení otáček ventilátoru

Nyní se blíže seznámíme s koncepcí současného chlazení spalovacího motoru u válcového dynamometru. Také se seznámíme se souhrnem požadavků na modul řízení otáček ventilátoru, který ovládá frekvenční měnič a ten následně řídí motor ventilátoru. Obeznámíme se také s frekvenčním měničem, jeho ovládáním a parametry. Jedna z kapitol bude zaměřena na motor ventilátoru připojený k frekvenčnímu měniči a na jeho parametry.

1.1 Stávající stav chlazení motoru u válcového dynamometru

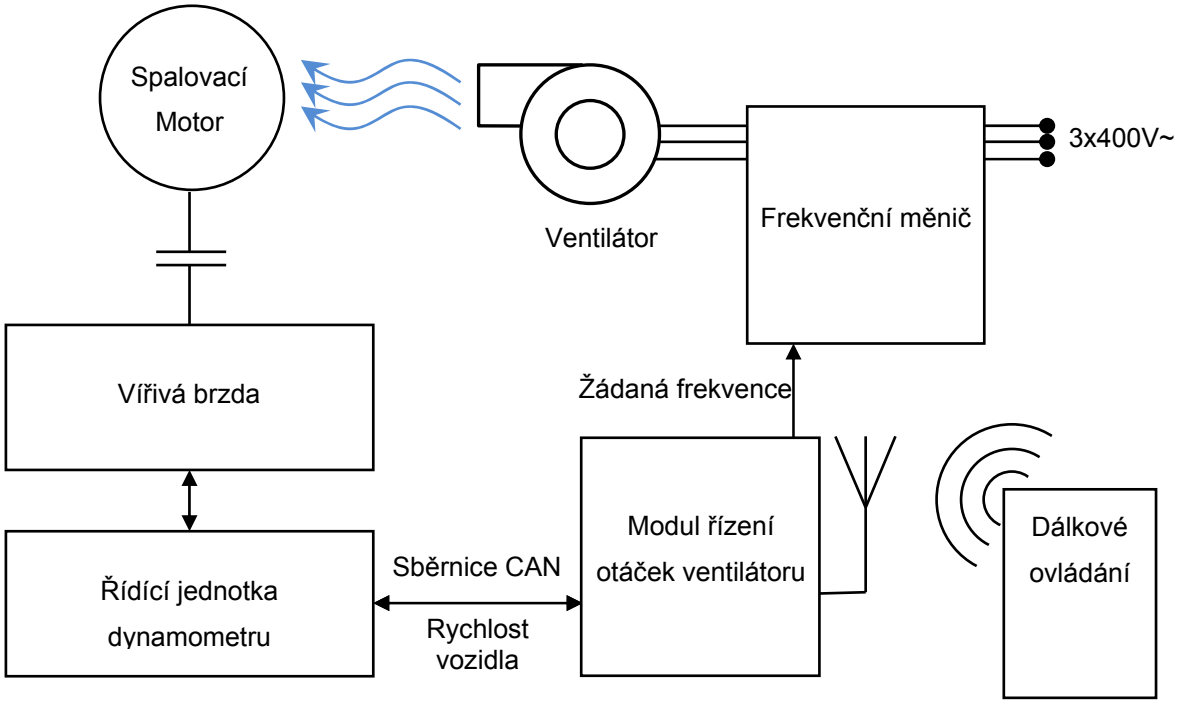
Chladicí systém vozidla či motoru se dosud skládá z ventilátoru a frekvenčního měniče, který pomocí pulzně šířkové modulace řídí motor ventilátoru. Doposud byl systém chlazení motoru měřeného vozidla konstruován tak, že nebyla možnost řídit otáčky ventilátoru. Frekvenční měnič nám sloužil pouze pro plynulý rozběh asynchronního motoru ventilátoru, tak aby nedošlo k výpadku elektrické energie, důsledkem proudového rázu, při zapojení motoru ventilátoru přímo na elektrickou síť. Doposud se celý systém chlazení před měřením na dynamometru zapnul a po ukončení měření vypnul a během samotného měření nebyla možnost regulace otáček ventilátoru.



Obr. 1.1 – Blokové schéma současné koncepce chlazení motoru

1.2 Budoucí podoba řízení ventilátoru pro chlazení motoru

V současné podobě chlazení, nelze regulovat proudění vzduchu v závislosti na rychlosti vozidla. Z tohoto důvodu se rozhodlo o úpravě chlazení, aby se navodili totožné podmínky, jako panují za provozu vozidla. Na základě dat z řídící jednotky dynamometru se bude regulovat rychlost otáček ventilátoru. Modul řízení otáček ventilátoru bude komunikovat s řídící jednotkou dynamometru po sběrnici CAN nebo bude ovládán pomocí dálkového ovládání. Při komunikaci po sběrnici CAN, bude do modulu periodicky zasílána informace o aktuální hodnotě rychlosti vozidla. Samotný modul nám posléze vyhodnotí požadovanou hodnotu otáček ventilátoru a pomocí frekvenčního měniče nám nastaví vhodné otáčky ventilátoru. Na obr. 1.2 je zobrazeno blokové schéma zařazení modulu otáček ventilátoru do současné koncepce dynamometru.



Obr. 1.2 – Zařazení modulu otáček ventilátoru do systému dynamometru

Dálkové ovládání modulu umožní ovládat ventilátor i z místa měřícího pracoviště a tak omezí neustálé přistupování k frekvenčnímu měniči. Pokud bude modul řízen pomocí sběrnice CAN, bude řízení otáček plně automatizováno.

1.3 Frekvenční měnič

Frekvenční měnič jindy také nazývaný měnič kmitočtu je zařízení, které slouží k přeměně elektrického proudu nebo napětí s určitým kmitočtem na elektrický proud nebo napětí s jiným kmitočtem. Velmi častým důvodem použití frekvenčního měniče bývá potřeba plynulé regulace otáček asynchronního motoru, jak je tomu i v tomto případě. Frekvenční měniče prodělaly velký vývoj techniky společně s vývojem mikroprocesorů a polovodičových součástek.

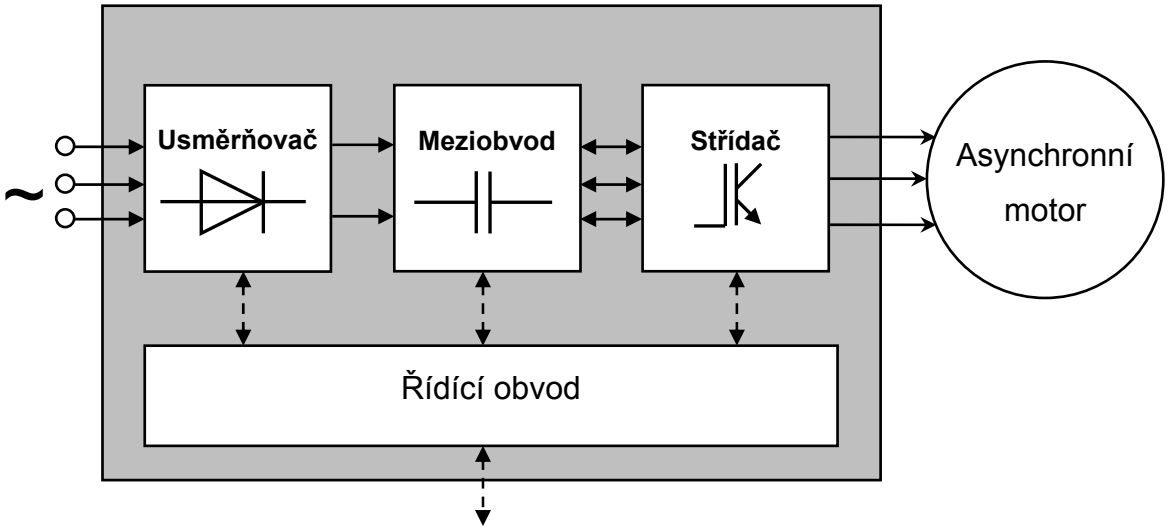
Frekvenční měnič mění konstantní napětí napájecí sítě na stejnosměrné napětí. Z tohoto stejnosměrného napětí, vytváří střídač novou síť s proměnlivým napětím a frekvencí pro motor. Hlavní části frekvenčního měniče lze rozdělit do čtyř částí usměrňovač, meziobvod, střídač a řídící obvod. Na vstup usměrňovače se připojí síťové trojfázové napětí, z něhož usměrňovač vytvoří pulsující stejnosměrné napětí. Existují dva základní typy usměrňovačů:

- Řízené – Umožňují řídit výstupní napětí z usměrňovače. Jde o aktivní řízený usměrňovač, jenž využívá u frekvenčních měničů IGBT tranzistorů, používá se pro rekuperační měniče.
- Neřízené – Neumožňují řízení výstupního napětí z usměrňovače. Chceme-li změnit výstupní napětí usměrňovače, musíme změnit vstupní napětí. K usměrnění se využívá polovodičové diody.

V stejnosměrném meziobvodu je filtrační kondenzátor pro vyhlazení usměrněného napětí a tlumivka pro tlumení proudového nárazu při zapojení nenabitého kondenzátoru. V meziobvodu se uchovává energie, která je potřebná pro střídač. Ve střídači se vytváří kmitočet výstupního motorového napětí. Přetváří tedy nejen stejnosměrné napětí na střídavé s proměnným kmitočtem, ale také reguluje velikost výstupního napětí. Řídicí obvod využívá mikroprocesor, jenž vysílá a přijímá signály z usměrňovače, meziobvodu i střídače, tedy nám otevírá a uzavírá polovodičové spínače řídícími signály. Blokové schéma frekvenčního měniče je vidět na obr. 1.3. Řízení připojeného motoru se provádí pomocí změny napětí motoru, současně se změnou frekvence, tedy pomocí pulzně šířkové modulace (PWM - Pulse Width Modulation). Motor je v každém pracovním bodě řízen tak, aby byl optimálně magneticky i elektricky využit.[3]

Frekvenční měniče se nejčastěji používají k řízení ventilátorů a zubových čerpadel. Jejich výhody jsou jednoduchost a přesnost ovládání připojeného stroje.

Další výhodou je odpadnutí komponent pro potřeby monitorování a ochrany motorů jako je tepelné relé. Nevýhodami je velké rušení, které frekvenční měnič při svém řízení vytváří a dosti často bývá vysoká pořizovací cena.



Obr. 1.3 – Blokové schéma frekvenčního měniče s asynchronním motorem

1.3.1 Ovládání frekvenčního měniče

V této bakalářské práci je využito frekvenčního měniče SINAMICS PM240 o výkonu 5,5 kW od firmy Siemens. Dále je použito řídící jednotky CU240S PN-F a ovládacího panelu SINAMICS G110, také od firmy Siemens. Ovládání frekvenčního měniče může být realizováno ručně pomocí ovládacího panelu, nebo pomocí analogových a digitálních vstupů. Také je možno pro ovládání využít sběrnice Profibus. V případě modulu řízení otáček ventilátoru je řízení provedeno pomocí analogových a digitálních vstupů na frekvenčním měniči. Na digitální vstupy jsou vysílány hodnoty logické jedničky a nuly pro určení požadovaného stavu. První digitální vstup nastavuje směr otáčení ventilátoru doleva či doprava a druhý digitální vstup nám určuje, zda má být ventilátor zapnut či nikoli. Dále se pomocí analogového vstupu frekvenčního měniče nastavuje požadovaná hodnota otáček ventilátoru. Kde hodnota 10 V odpovídá jmenovité frekvenci elektromotoru, v našem případě 50 Hz.

Pro ovládání pomocí analogových a digitálních výstupů bylo nutné vhodně spojit modul řízení otáček ventilátoru a frekvenční měnič. Na následujícím obrázku obr. 1.4 nalezneme vhodné spojení frekvenčního měniče a modulu řízení otáček ventilátoru, které je nutno provést pro správnou funkci řízení otáček.

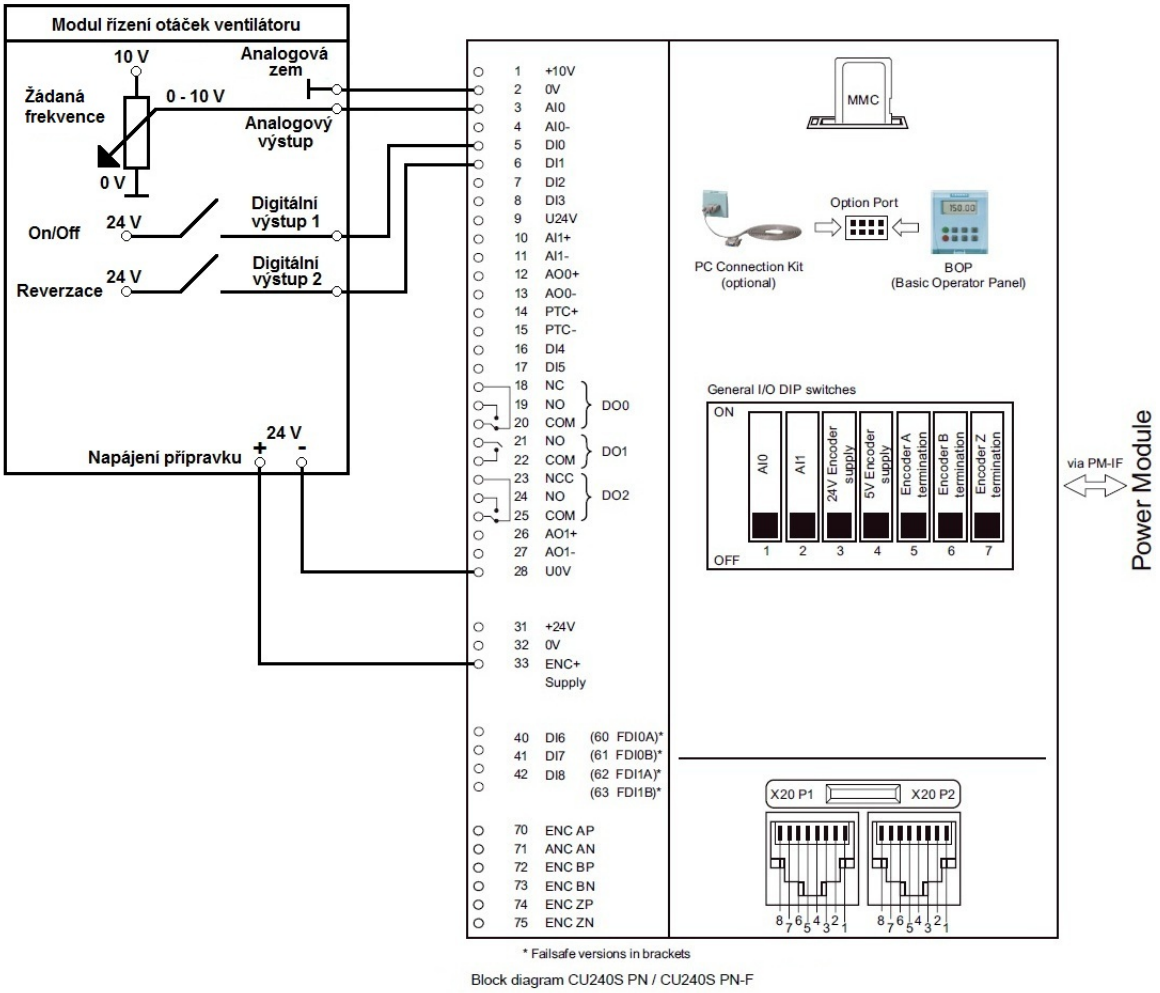
Parametry frekvenčního měniče:

Vstupy:

- Digitální vstupy 24 V
- Analogové vstupy 0-10 V

Výstupy:

- Pomocné napájení 24 V / 300 mA (viz. Příloha 1)



Obr. 1.4 – Spojení frekvenčního měniče a modulu řízení otáček ventilátoru

1.3.2 Parametrování frekvenčního měniče

Dříve než dojde k prvnímu spuštění motoru, je nutné nastavit parametry měniče v závislosti na konkrétních podmínkách, tato činnost se nazývá parametrování. Parametry zadávané do měniče určují, jak se má měnič chovat při chodu. Volby vhodných parametrů mají vliv na chování zařízení připojených k frekvenčnímu měniči. Při parametrizaci měnič dostává informaci o připojeném motoru na jeho

výstupu, volbou zadaných parametrů. Vlastnosti a způsob ovládání frekvenčních měničů lze velmi snadno měnit, záleží na nastavení parametrů frekvenčního měniče. Parametry zajišťují vlastně základní činnost a vlastnosti chování měniče, umožňující snadné začlenění do technologického systému, včetně nastavení různých omezení, rozběhových a doběhových ramp, křivek a podobně.

V našem případě s použitím modulu řízení otáček ventilátoru, spočívá parametrizace měniče v nastavení výkonové části měniče, s ohledem na použitý elektromotor a způsobu jeho řízení. Je využito skalárního U/f řízení. Dále musel být nastaven způsob ovládání frekvenčního měniče. Používá se ovládání, kde využíváme analogových a digitálních vstupů frekvenčního měniče. Analogová hodnota nám určuje žádanou hodnotu frekvence. Dva digitální vstupy nám udávají, zda má být měnič zapnut či nikoli (ON/OFF1) a směr otáčení (REV). Toto nastavení u našeho frekvenčního měniče je pojmenované ON/OFF and REV. Nastavení parametrů frekvenčního měniče v případě spojení s modulem řízení otáček ventilátoru nalezneme v příloze 1 této práce.

1.4 Motor ventilátoru

Jako pohon ventilátoru je použit elektromotor. Jde o motor od společnosti Siemens s označením 1LA7130-4AA60. Jedná se o trojfázový asynchronní motor nakrátko zapojen do trojúhelníka. Štítkové hodnoty motoru jsou uvedeny v následující tabulce.

Parametr	Hodnota
Váha	42 kg
Krytí	IP 55
Výkon	5,5 kW
Frekvence	50 Hz
Napětí	400 V
Proud	11,4 A
Cos φ	0,81
Otáčky	1455 ot/min

Tab. 1.1 – Štítkové hodnoty motoru

Více informací o motoru se dozvíme z katalogu [2]. Jedná se tedy o asynchronní motor, který je nejrozšířenějším pohonem v elektrotechnice. Jeho výhody jsou vysoká spolehlivost a jednoduchá konstrukce.

Obr. 1.5 - Ventilátor

1.5 Souhrn požadavků na řídicí systém

Doposud jsme se seznámili s frekvenčním měničem, jeho funkcí v systému, parametrech a ovládání. Také jsme se dozvěděli o motoru, který je použit jako pohon ventilátoru. Nyní se blíže seznámíme s požadavky na modul řízení otáček ventilátoru. Při návrhu zařízení bylo nutné vyhovět zadaným požadavkům na modul. Jedním z prvotních požadavků je komunikace s nadřazeným systémem, která probíhá po sběrnici CAN. Modul řízení otáček ventilátoru musí komunikovat s řídící jednotkou dynamometru, aby byl schopen okamžitě reagovat na změnu rychlosti měřeného objektu a mohl měřený objekt uchladit, ať už se jedná o motorové vozidlo či motor samostatný.

Ovládání frekvenčního měniče bude tedy probíhat pomocí analogových a digitálních vstupů, proto je nutné pro modul zvolit vhodné analogové a digitální výstupy. Pomocí digitálních výstupů se bude ovládat chod měniče a reverzace ventilátoru. Dále se pomocí analogového výstupu ovládá rychlost otáček ventilátoru a tudíž rychlost proudění vzduchu z ventilátoru. Velikost otáček ventilátoru se musí nechat nastavovat tak, aby se zvyšovala od počátku po 1 % až po jmenovitou hodnotu motoru. Je také požadována komunikace po sériové lince RS-232

s počítačem, pro případné odladění firmwaru zařízení a pro zobrazení dějů a veličin, s kterými modul aktuálně pracuje.

Dalším požadavkem na modul řízení otáček ventilátoru je možnost ho ovládat dálkovým ovládáním. Dálkové ovládání by mělo přispět ke komfortu ovládání. Slouží k hlavnímu ovládání modulu řízení otáček ventilátoru. Pomocí dálkového ovládání se zadává požadované chování ventilátoru, jako je rychlost otáčení či způsob ovládání.

Celý systém musí být galvanicky oddělený jak ze strany vstupů, tak ze strany výstupů aby nedocházelo k rušení, které frekvenční měnič vytváří dosti vysoké.

Napájení modulu řízení otáček ventilátoru lze provést z frekvenčního měniče po aktivaci pomocného napájení pro otáčkové čidlo. Modul tak bude možno napájet 24 V a maximální přípustný odběr modulu řízení otáček ventilátoru bude moct činit 300 mA. Více se o aktivaci pomocného napájení dozvíme v příloze 1. Modulu musí být také umožněno napájení z externího zdroje, pokud by napájení z frekvenčního měniče nebylo možné, proto musíme modul opatřit vhodným konektorem pro připojení externího napájení. Volba napájení modulu se bude provádět pomocí mechanické propojky (jumper), umístěné na desce plošného spoje, pomocí které bude zvolen jeden aktuálně vhodný způsob napájení. Souhrn hlavních požadavků na modul řízení otáček ventilátoru nalezneme v následující tabulce.

Parametr	Hodnota	Význam
Analogový výstup	0-10 V	Spojitá hodnota pro určení požadované frekvence
Digitální výstup 1	0/24 V	Měnič vypnut/zapnut
Digitální výstup 2	0/24 V	Směr otáčení ventilátoru doleva/doprava
Napájení	24 V / 300 mA	Napájení modulu řízení otáček ventilátoru
Ovládání modulu řízení otáček ventilátoru	-	Dálkové ovládání/CAN

Tab. 1.2 – Souhrn požadavků na modul řízení otáček ventilátoru

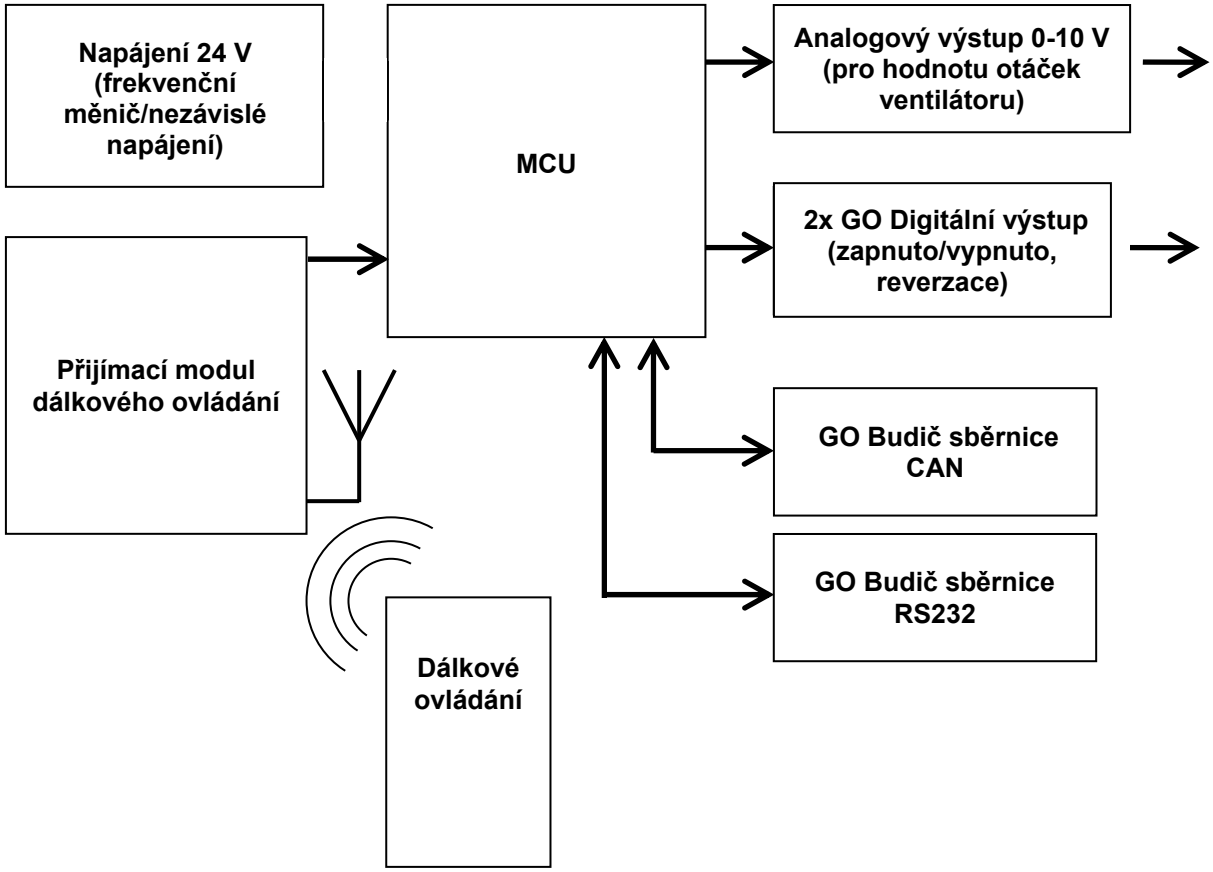
2 Hardware

Tato kapitola se zabývá volbou vhodného hardwaru modulu řízení otáček ventilátoru. V předchozích částech práce byly vytyčeny jasné cíle, jak by měl modul řízení otáček ventilátoru pracovat a jak bude zařazen v celém systému dynamometru. Nyní se věnujeme návrhu desky plošných spojů a volbě jednotlivých komponent potřebných pro vyhotovení zařízení a jeho správnou funkci.

Celé zařízení musí vhodně ovládat frekvenční měnič pomocí připojení analogových a digitálních vstupů na měniči. Tudíž naše zařízení musí obsahovat analogový výstup na určení požadované hodnoty frekvence a dva digitální výstupy pro volbu zapnutí a vypnutí frekvenčního měniče a pro reverzaci směru otáčení ventilátoru. Toto zařízení by se mohlo zdát na první pohled velmi jednoduché, a lze ho realizovat pouze pomocí dvou přepínačů a jednoho potenciometru připojeného k frekvenčnímu měniči jak, je naznačeno v manuálu pro frekvenční měnič [9]. Ovšem další z požadavků na modul řízení otáček ventilátoru je komunikace po sběrnici CAN s nadřazeným systémem, tak aby se regulace otáček ventilátoru prováděla automaticky a to v závislosti na rychlosti měřeného objektu. Podstatným požadavkem je také galvanické oddělení, které se musí realizovat jak na vstupu, tak na výstupu zařízení, aby nedocházelo například k rušení analogového signálu o požadované hodnotě otáček. Celé zařízení bude řízeno vhodným mikrokontrolérem (dále MCU).

2.1 Blokové schéma modulu řízení otáček ventilátoru

Nyní, když je znám způsob ovládání frekvenčního měniče pomocí modulu řízení otáček ventilátoru, je vhodné si uvést hlavní části, jež musí modul obsahovat. Pro názornost, je zde uvedeno blokové schéma modulu řízení otáček ventilátoru obr. 2.1, z kterého bude vycházet celý návrh a konstrukce. Blokové schéma je rozděleno do několika hlavních částí, jež si popíšeme v dalším textu. V blokovém schématu je rovněž vidět tok informací modulem. Dálkové ovládání vyšle informaci o zapnutí, vypnutí či jiné změně ve funkci ventilátoru, MCU následně informaci zpracuje a pomocí analogových a digitálních výstupů upraví chování ventilátoru. Sběrnice CAN by nám měla umožnit vyčítání rychlosti měřeného objektu, kterou bude zpracovávat MCU a následně plně automaticky řídit rychlost ventilátoru, opět pomocí analogových a digitálních výstupů.



***GO – Galvanické oddělení**

Obr. 2.1 – Blokové schéma modulu řízení otáček ventilátoru

2.2 Hlavní části modulu řízení otáček ventilátoru

Jelikož jsou už známy hlavní části modulu řízení otáček ventilátoru, je nutné si popsat a blíže vysvětlit funkci jednotlivých částí zařízení. Každé části se bude věnovat jedna z následujících kapitol.

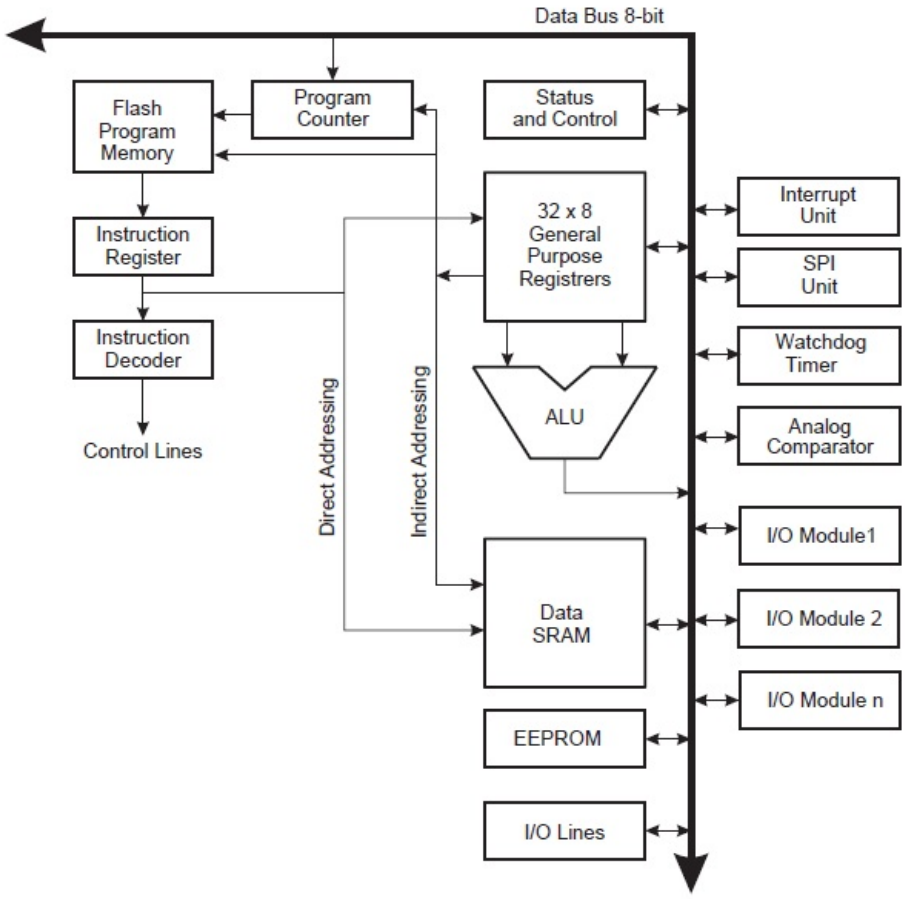
2.2.1 MCU

MCU má na starosti správné řízení a ovládání celého přípravku, často bývá srdcem každého složitějšího moderního elektronického zařízení. Při volbě vhodného MCU je nutné se zaměřit na požadavky, jenž jsou na modulu řízení otáček ventilátoru kladeny. Jedním z hlavních kritérií na volbu vhodného MCU je komunikace po sběrnici CAN. Bylo by vhodné, aby zvolený MCU obsahoval řadič CAN, tím se návrh celého zařízení značně zjednoduší. V úvahu je nutno brát, že po zkonstruování modulu řízení otáček ventilátoru, je nutno tento přípravek také odzkoušet a otestovat. Za tímto účelem bude přípravek obsahovat komunikační linku

RS-232. Z toho plyne, že vybrané MCU musí také obsahovat řadič UART. Dále je vhodné volit MCU tak, aby obsahoval komunikační rozhraní ISP nebo I2C pro komunikaci s okolními periferiemi (D/A převodníkem) na desce plošných spojů.

Požadavky na rychlost MCU nejsou zásadní vzhledem k aplikaci. Svým výkonem bude postačovat 8-bitový MCU, s architekturou vhodnou na programování ve vyšším programovacím jazyce (jazyk C). Ze souhrnu těchto požadavků byl zvolen MCU od firmy Atmel s označením AT90CAN128.

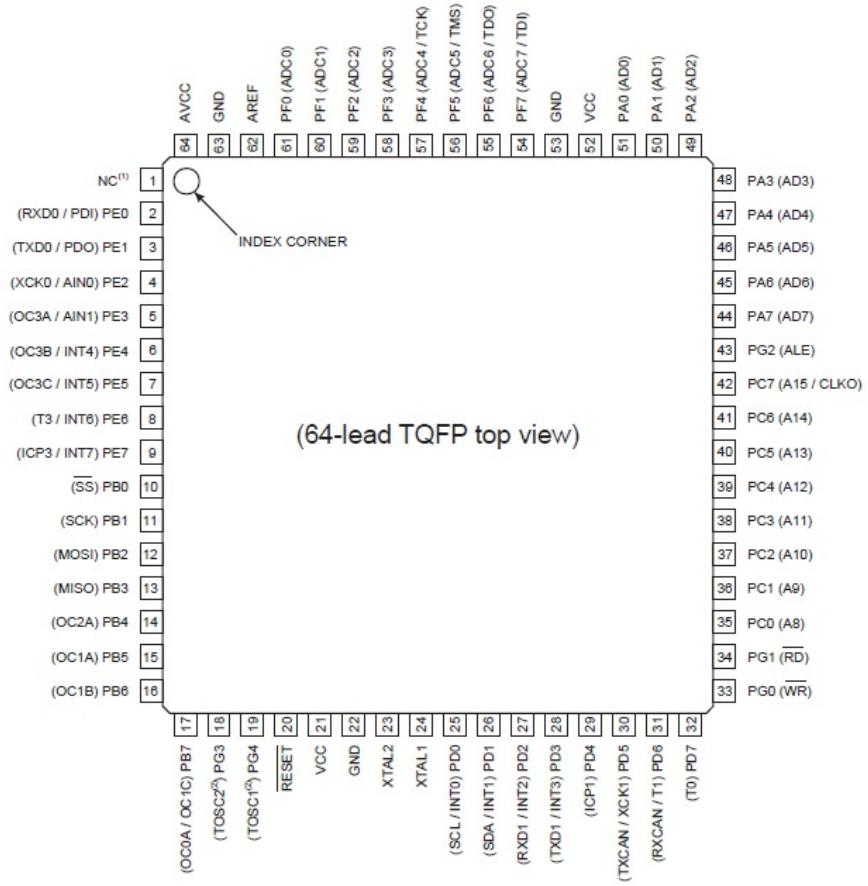
Jedná se o 8-bitový MCU s nízkým příkonem, jenž je založen na technologii CMOS a architektuře AVR. Obsahuje redukovanou sadu instrukcí RISC (Reduced Instruction Set Computer). Tento MCU využívá harvardskou architekturu, to znamená, že má oddělenou paměť dat od paměti programu. Paměť programu tvoří elektricky programovatelná FLASH paměť o velikosti 128 KB. Datová paměť má velikost 4 KB a je tvořena statickou polovodičovou pamětí SRAM. Programování FLASH paměti lze realizovat pomocí sběrnice SPI, nebo přes rozhraní JTAG jak je tomu v našem případě. Na obr 2.2 je zobrazena architektura AT90CAN128. V následujícím výčtu jsou uvedeny základní vlastnosti MCU.



Obr. 2.2 – Vnitřní architektura AT90CAN128

Přehled vlastností AT90CAN128:

- Periferie:
    - programovatelný Watchdog s On-chip oscilátorem
    - 8-bitový synchronní časovač/čítač (Timer/Counter-0): 10-bitová předdělička, externí Event Counter, výstup Compare nebo 8-bitový PWM výstup
    - 8-bitový asynchronní časovač/čítač (Timer/Counter-2): 10-bitová předdělička, externí Event Counter, výstup Compare nebo 8bitový PWM výstup, 32KHz oscilátor pro RTC operaci
    - duální 16-bitové synchronní čítače/časovače: 10-bitová předdělička, vstup Capture s rušením šumu, externí Event Counter, 3 výstupy Compare nebo 16bitový PWM výstup, modulace výstupu Compare
    - 8-kanálový 10-bitový SAR ADC
    - Two-wire sériové rozhraní (I2C)
    - duální programovatelný sériový USART
    - Master/Slave sériové rozhraní SPI
    - Řadič CAN 2.0A a 2.0B
    - On-chip analogový komparátor
- Speciální vlastnosti MCU:
    - Power-on Reset a programovatelná detekce Brown-out
    - Interní kalibrovaný RC oscilátor
    - 8 externích zdrojů přerušení
    - pět režimů spánku: Idle, ADC Noise Reduction, Power-save, Power-down, Standby
    - Global Pull-up Disable
- I/O: 53 programovatelných I/O linek
- Pracovní napětí: 2.7 až 5.5V
- Rozsah teplot -40 až 125°C
- Maximální frekvence: 8MHz při 2.7V, 16MHz při 4.5V



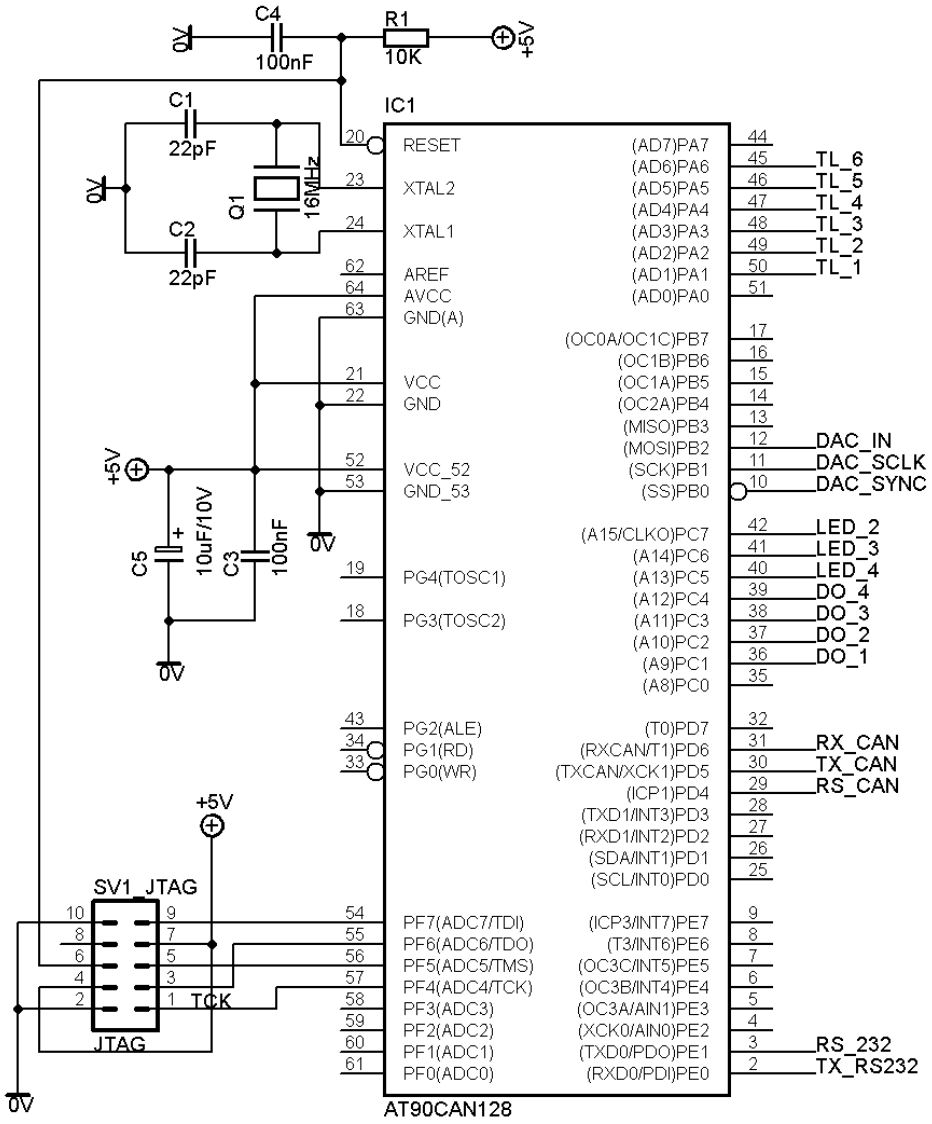
Obr. 2.3 – Rozmístění pinů na AT90CAN128

Podrobný popis vlastností MCU nalezneme v materiálech výrobce [11], rovněž tam nalezneme podrobnější popis pinů. Na obr. 2.3 je zobrazeno pouzdro MCU s označenými vývody. Využití MCU v modulu řízení otáček ventilátoru je zobrazeno na obr. 2.4 kde jsou mimo MCU zobrazeny i podpůrné součástky pro správnou funkci. Jedním z nejdůležitějších prvků je externí oscilátor s taktovací frekvencí 16 MHz. Dále jsou to blokovací kondenzátory pro pokrytí nárazového proudového odběru IO. Ve schématu je také vidět zapojení konektoru pro připojení programátoru pracujícího přes sběrnici JTAG. Největší výhodou ale bezesporu bylo to, že má v sobě tento MCU zabudovaný řadič sběrnice CAN, jež byla původně vymyšlena pro automobilový průmysl, avšak jeho použití se rozšířilo i do jiných průmyslových odvětví. Protokol linkové vrstvy našeho mikrokontroléru AT90128CAN je definován normou ISO 11898.

Volba tohoto mikrokontroléru byla také z důvodu dostupnosti vývojového prostředí, které je k dispozici na Katedře elektrotechniky, elektroniky a zabezpečovací techniky na Dopravní fakultě Jana Pernera. Bylo tak možné si před

samotnou konstrukcí odzkoušet různé funkce MCU jako reakci na tlačítka s následným ovládáním, komunikace s jinými periferiemi a samotné. Další výhodou bylo již dřívější použití stejného typu MCU v jiných aplikacích s kladnými ohlasy.

Na obr. 2.4 je zobrazen MCU, na jehož port A je připojen přijímací modul dálkového ovládání. Na portu B je připojen D/A převodník. K Portu C jsou připojeny kontrolní LED diody a optočleny. Na port D je připojen budič sběrnice CAN a k portu E budič RS-232.



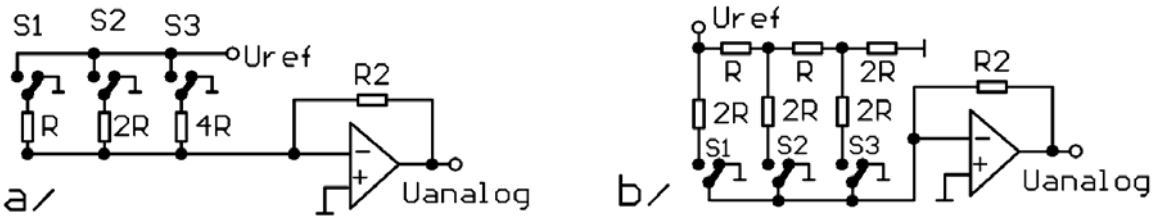
Obr. 2.4 – Schéma zapojení MCU v modulu řízení otáček ventilátoru

2.2.2 Analogový výstup

Slouží nám k určení námi požadované frekvence frekvenčního měniče, jež následně ovládá otáčky ventilátoru. Analogový výstup modulu řízení otáček ventilátoru je tvořen z několika hlavních komponent, které jsou zobrazeny na obr. 2.7, je to operační zesilovač, napěťová reference a digitálně analogový převodník.

Nejdříve se seznámíme všeobecně s funkcí D/A převodníků. Jedná se o elektronické součástky, jež jsou určeny pro převod digitálního (nespojitého) signálu na signál analogový (spojitý). Převádí tedy vstupní číslicový signál na analogové napětí. To může nabývat pouze určitých diskrétních hodnot. Základní část převodníku tvoří blok spínačů, který přivádí přesné referenční napětí na vstupy odporové sítě. Odporovou sítí je následně skládáno výstupní analogové napětí, druhy sítí vidíme na obr. 2.5. Referenční napětí určuje rozsah výstupního napětí převodníku. Rozsah převodníku je tak možné ovlivnit u jakékoliv konkrétní aplikace. Výsledné výstupní napětí z převodníku se získává z odporové sítě. Pro hodnocení vlastností převodníku si vyjmenujeme některé důležité parametry:

- Rozlišení – je dáno počtem bitů převodníku, tedy jeho šířkou.
- Přesnost – je to odchylka od ideální převodní charakteristiky.
- Linearita – pro lineárně vzrůstající hodnotu číselného vstupu by mělo výstupní napětí vzrůstat o stejnou hodnotu.
- Teplotní stabilita – závislost parametrů převodníku na teplotě.
- Rychlost převodu – čas ustálení výstupního analogového napětí, při změně číslicového vstupu.



Obr. 2.5 – 3-bitový D/A převodník a) s váhovou strukturou b) s příčkovou strukturou odporové sítě.

Rozlišovací schopnost neboli kvantizační krok D/A převodníku je dán konečným počtem hodnot výstupního napětí a lze ji určit dle vztahu (2.1).

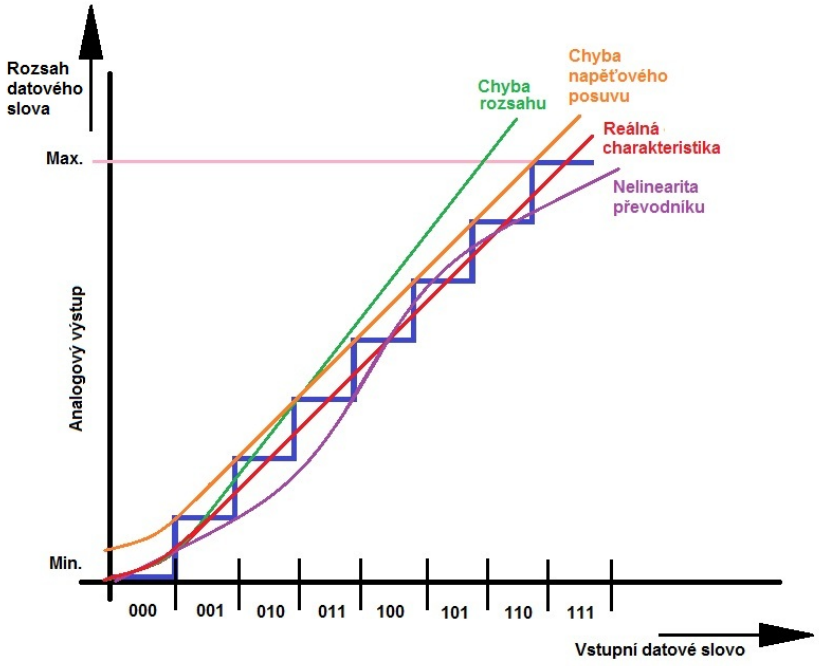
$$Q = \frac{1}{2^n - 1} \tag{2.1}$$

Kde:

Q – kvantizační krok

n – počet bitů vstupního stavového slova

Jedna z dalších vlastností, která nás zajímá u D/A převodníků je přesnost převodu vstupního datového signálu na výstupní signál. Ideální charakteristika nám udává závislost mezi vstupním datovým slovem a výstupní analogovou veličinou. Na obr. 2.6 je zobrazena přesnost a chyby v převodu tří bitového D/A převodníku. Celková přesnost převodníku je podstatně závislá na stabilitě zdroje referenčního napětí. Další vlastností, jež je dobré znát je maximální rychlost převodníku. Je určena počtem vstupních datových slov, která jsou D/A převodníkem převedena na výstupní analogovou veličinu za jednotku času. Dobu převodu spočteme jako převrácenou hodnotu rychlosti času.



Obr. 2.6 - Převod 3bitového D/A převodníku

D/A převodníky se používají všude tam, kde je třeba z digitálního signálu udělat analogový tedy ve všech přehrávačích hudby, nebo ve zvukových kartách počítačů. Používají se také v mobilních telefonech.

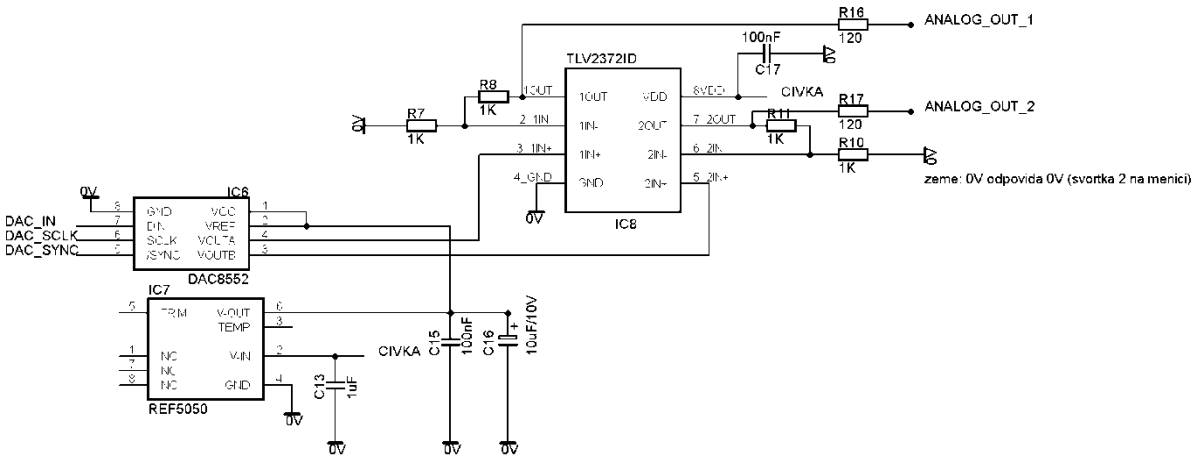
Při výběru vhodného převodníku je dobré si ověřit vlastnosti součástky v Datasheetu a dle parametrů součástky zvolit vhodnou pro naši aplikaci. Požadavky kladené na náš převodník jsou:

- Rozlišovací schopnost převodníku minimálně 8-bit (jelikož požadujeme krok řízení otáček po 1 %)
- Parametr rychlosti převodu není až tak důležitý s ohledem na dynamiku řízení ventilátoru, která se řádově pohybuje v sekundách
- Vhodné sériové rozhraní (SPI či I2C) pro ovládání převodníku z MCU

V našem případě byl zvolen D/A převodník s označením DAC8552. Jedná se o 16-bitový, dvoukanálový převodník s výstupním analogovým napětím 5 V, rozlišovací schopností 4LSB (LSB - least significant bit, nejméně významný bit), rychlostí převodu 10 ms a komunikační rozhraní SPI, jež nám vyhovuje s ohledem na k připojení na MCU. Tento převodník splňuje naše požadavky a byl s úspěchem na Katedře elektrotechniky, elektroniky a zabezpečovací techniky použit v podobných nenáročných aplikacích. Parametry tohoto D/A převodníku daleko převyšují stanovené požadavky. Volba na tento D/A převodník padla z důvodu dostupnosti ve formě vzorků zdarma a malého počtu vývodů (pouze 8). [14]

Nutností je připojení přesné externí napěťová reference k převodníku. Námi zvolená reference má označení REF5050 a jejím výrobcem je firma Texas Instruments [15]. Základní parametry této reference jsou:

- Výstupní referenční napětí 5 V
- Teplotní drift 8 ppm
- Počáteční přesnost 0,1 %
- Dlouhodobá stabilita 5 ppm / 1000 hodin



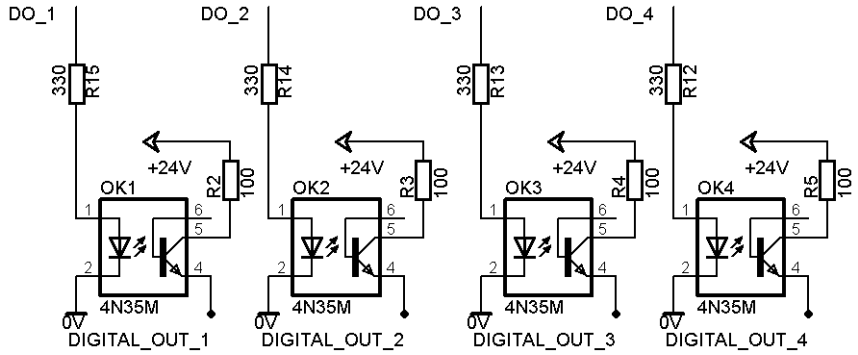
Obr. 2.7 - Analogový výstup

Maximální výstupní napětí převodníku je 5 V, což je pro určení maximální požadované frekvence frekvenčního měniče nedostačující hodnota. K určení nejvyšší frekvence je požadováno napětí 10 V. To znamená, že výstupní napětí z převodníku musíme ještě dvakrát zesílit. Z tohoto důvodu musí být na výstup převodníku připojen ještě operační zesilovač v neinverujícím zapojení, aby došlo k zesílení signálu z 5 na 10 V, jež je požadováno na analogovém výstupu modulu řízení otáček ventilátoru. Zároveň se tím impedančně oddělí výstup D/A převodníku. Byl vybrán operační zesilovač TLV2372. Důležitou vlastností toho typu operačního zesilovače je, že dokáže pracovat v blízkosti napájecích napětí, proto nese tento operační zesilovač označení RAIL-TO-RAIL. Vystačíme si tedy s napájecím napětím z D/A převodníku, které činí 5V. Další vlastnosti tohoto operačního zesilovače jsou šířka pásma 3MHz, rozsah napájecího napětí 2,7-16 V, napájecí proud 0,55 mA / kanál a rozsah pracovních teplot -40 až 125 °C [16]. Výstupní napětí z tohoto operačního zesilovače se již bude pohybovat v rozmezí 0-10 V a to v závislosti na vstupní číselné hodnotě vysílané z MCU do D/A převodníku. Tím pádem již bude vyhovovat našemu požadavku na spojité analogové řízení otáček ventilátoru.

2.2.3 Digitální výstup

Digitální výstupy modulu řízení otáček ventilátoru slouží k určení směru otáčení ventilátoru doleva, nebo doprava a k zapnutí či vypnutí ventilátoru. Digitální výstupy jsou galvanicky odděleny optickou vazbou. Více se o galvanickém oddělení dozvíme v kapitole 2.2.7, kde se problematice galvanického oddělení věnujeme více. Digitální výstupy nám simulují mechanické přepínače. Na výstupu je hodnota logické nuly

(0 V) nebo jedničky (v našem případě 24 V). Je využito optočlenu s označením 4N35 řízeného mikrokontrolérem [7]. Jelikož se jedná o levnou a jednoduchou součástku i její ovládání, tak byl modul řízení otáček ventilátoru osazen čtyřmi digitálními výstupy. Dva jsou využity k ovládání frekvenčního měniče a další dva jsou použity jako rezervní, popřípadě je lze využít k řízení dalšího zařízení.



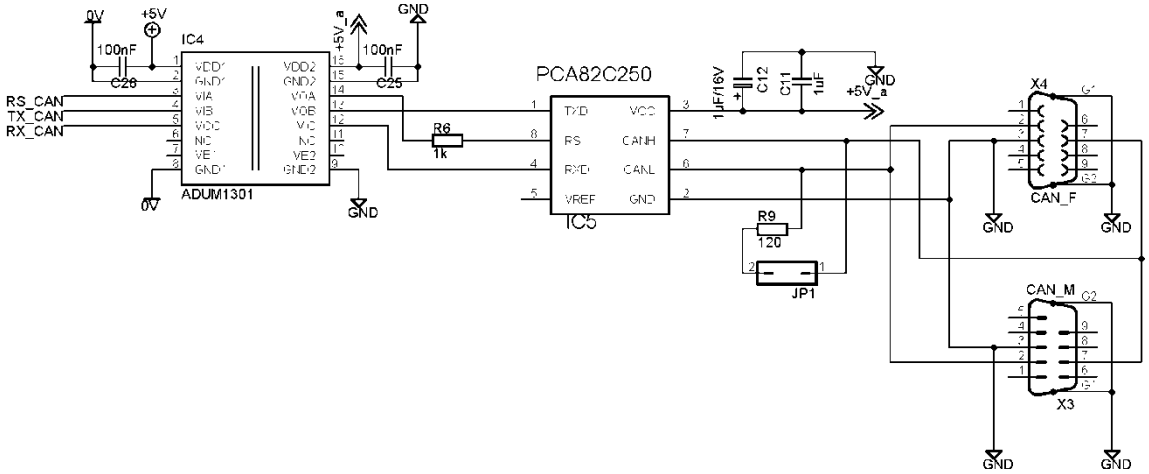
Obr. 2.8 – Digitální výstupy

2.2.4 CAN

Jelikož potřebujeme zjišťovat aktuální hodnotu rychlosti měřeného objektu na dynamometru, podle které bude MCU odvozovat vhodné otáčky ventilátoru k chlazení měřeného objektu, musíme modul řízení otáček ventilátoru přizpůsobit tak, aby byl schopen přijímat zprávu o aktuální rychlosti. Protože je v systému dynamometru již zakomponována sběrnice CAN, tak ji lze využít právě pro výčet měřené rychlosti.

Rozhraní CAN (Controller Area Network) je nejčastěji využito pro komunikační síť senzorů a funkčních jednotek v automobilu. Z automobilové oblasti se CAN rychle rozšířil do ostatních průmyslových aplikací, především do automatizační techniky. Jedná se o sériovou datovou sběrnici vyvinutou firmou BOSCH. Elektrické parametry fyzického přenosu jsou specifikované normou ISO 11898 a maximální teoretická rychlost přenosu sběrnice je 1 Mb/s. Tato sběrnice umožňuje výměnu dat mezi řídícími jednotkami a to pomocí datových rámců. Ty se skládají z datových polí, které jsou složené z různě definovaného počtu bitů (1 nebo 0). Pole s datovými informacemi obsahuje vlastní zprávu. Datový rámec obsahuje také pole pro řídící a kontrolní informace. Komunikace probíhá následujícím způsobem. Jedna z řídících jednotek vysílá do sítě data. Zbylé jednotky data přijímají a přijatá data vyhodnocují.

Pokud shledá řídící jednotka, že přijaté data jsou určena pro ni, tak je zpracuje, v jiném případě data ignoruje.



Obr. 2.9 – Zapojení CAN

Nejdůležitějším prvkem v tomto bloku je budič sběrnice CAN. Budič zajišťuje propojení fyzické sběrnice s řadičem MCU. Od MCU dostává data, jež se mají odeslat po sběrnici a zároveň mu přicházejí informace ze sběrnice, které posílá do MCU. V našem přípravku byl zvolen řadič od výrobce PHILIPS s označením PCA82C250. Podrobné informace o budiči jsou uvedeny v materiálech od výrobce [17]. Na obr. 2.9 je zobrazeno zapojení budiče CAN v modulu řízení otáček ventilátoru. Jelikož je linka CAN galvanicky oddělena od MCU, je na datových linkách použit převodním ADUM, který zajišťuje galvanické oddělení MCU od řadiče CAN. Jsou zde zobrazeny i dva konektory CANNON 9 jeden typu MALE a druhý FEMALE, to z důvodu zařazení a zřetězení celého zařízení do systému dynamometru. Důležitým členem je také rezistor R9, který slouží k impedančnímu přizpůsobení konce sběrnice CAN. Pokud to ovšem naše aplikace nevyžaduje tak nemusí být tento rezistor připojen. Pro připojení poslouží konektor JP1 (jumper), jež nalezneme na desce plošného spoje.

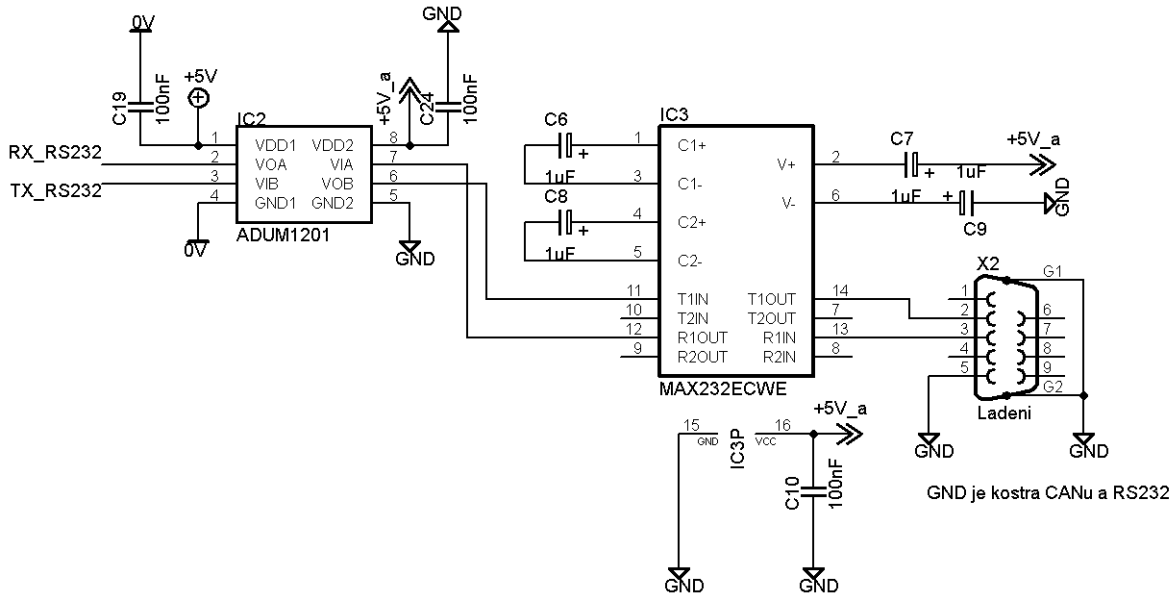
Vlastnosti přenosu jsou popsány následující tabulkou:

Parametr	Hodnota
Komunikační rychlost	1000 kb/s
Datový rámec	CAN 2.0A

Tab. 2.1 – Tabulka přenosu CAN

2.2.5 RS-232

Pro ověření funkce přípravku a ladění firmwaru modulu řízení otáček ventilátoru je využito sériové linky RS-232. Po této lince lze získávat informace, se kterými modul aktuálně pracuje jako například aktuální rychlost měřeného objektu. Na obr. 2.10 je zobrazen blok s budičem/přijímačem pro RS-232 a galvanické oddělení, kterému se budeme více věnovat v kapitole 2.2.7.



Obr. 2.10 – Zapojení RS-232

Napěťová úroveň RS-232 je 12 V, zatímco v mikrokontolérové technice se využívá logiky CMOS, kde se využívá 5 V úrovně. Z těchto důvodů musíme k MCU zařadit takový obvod, jež bude logické úrovně převádět. V našem výrobku je použit budič od výrobce MAXIM IC s označením MAX232CWE. Tento obvod obsahuje dvojici oddělovačů konvertujících napěťové úrovně [18]. Napětí je pro RS-232 získáváno pomocí nábojové pumpy a výstupní napětí proto velice závisí na kvalitě použitých kondenzátorů. V našem zapojení jsou použity tantalové kondenzátory, jež slouží pro nábojovou pumpu jako kapacity a jsou připojeny na vývody 1 až 6. Piny 13 a 14 slouží k připojení TTL signálu. Převedený signál dále putuje na piny 11 a 12 a je zpracován MCU. Tok informace může být i opačný.

Datový rámec přenosu po RS-232 je popsán následující tabulkou:

Parametr	Hodnota
Komunikační rychlost	9600 Bd/s
Datové bity	8
Parita	žádná
Stop bit	1
Mód komunikace	Asynchronní

Tab. 2.2 – Tabulka charakterizující datový rámec RS-232

2.2.6 Dálkové ovládání

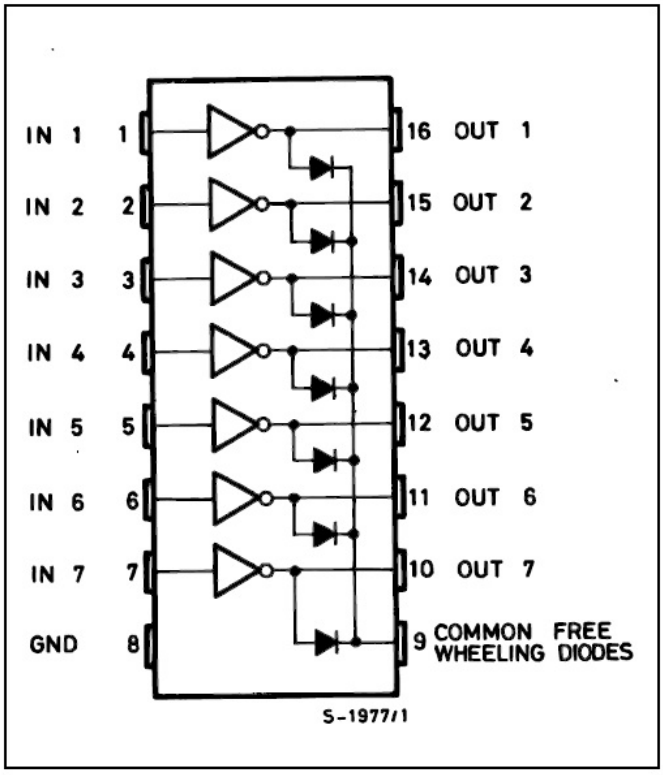
Na dnešním trhu, který je nabídkou dálkových ovladačů přesycen, nebylo těžké vybrat dálkové ovládání pro modul řízení otáček ventilátoru. Hlavní volbou, jež musela být vyřešena, byl způsob ovládání zařízení pomocí dálkového ovladače. Na trhu se objevují dva druhy dálkového ovládání, jeden pracuje na principu infračerveného paprsku, který přenáší signál z dálkového ovladače na zařízení. Druhý způsob je ovládání radiové. Radiový signál vysílaný dálkovým ovládáním přijímá přijímač, který následně řídí zařízení, ke kterému je přijímač připojen.

Pro modul řízení otáček ventilátoru bylo vybráno radiové dálkové ovládání, které se skládá z přijímače MRX6 s plovoucím kódem a vysílačem TX-6K. Přijímací modul MRX6 již obsahuje kompletní algoritmus kódování, zabezpečení přenosu, dekódování a vysokofrekvenční část. Na výstup toho přijímače lze už přímo připojit nějaký spínací člen, například relé. V našem případě bude přijímač připojen k MCU, jež bude vyhodnocovat stavy stisknutí tlačítek a následně ovládat modul řízení otáček ventilátoru. Typ výstupu přijímače dálkového ovládání je otevřený kolektor s ochrannou antiparalelní diodou. Zapojení výstupů je znázorněno na obr. 2.12. Maximální povolený proud na jeden výstup činí 100 mA. Více informací o zatížení výstupů nalezneme v materiálech výrobce k obvodu ULN2003A, jež je součástí přijímače dálkového ovládání. [22]

Přijímač vyžaduje napájecí napětí v rozsahu 7–24 V. Napájení vysílače zajišťuje 12 V baterie typu 23A. Dálkový ovladač i přijímač pracují na frekvenci 433,92 MHz. Oba komponenty taktéž pracují s ASK (Amplitude-Shift Keying) modulací, jež se také nazývá amplitudové klíčování. Na obr. 2.11 je vidět přijímač

dálkového ovládání s vysílačem. Modulu přijímače dálkového ovládání se musí synchronizovat s dálkovým ovladačem, více o synchronizaci obou komponent se dočteme v materiálech výrobce. [8]

Obr. 2.11 – Dálkový ovládač s přijímačem

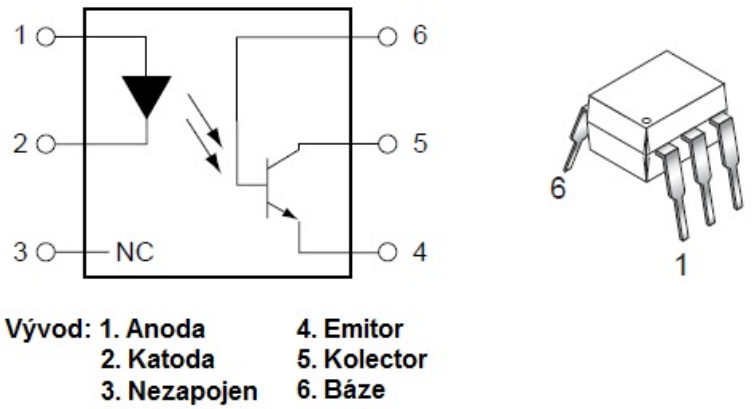


Obr. 2.12 – Zapojení výstupů přijímače dálkového ovládání

2.2.7 Galvanické oddělení

Je to způsob oddělení dvou částí obvodu tak, aby nedošlo k přímému spojení těchto rozdílných částí obvodu vodičem, ale aby docházelo k přenosu elektrické energie popřípadě k přenosu informace. Galvanické oddělení lze realizovat několika způsoby, nejznámějším způsobem je zřejmě transformátorová vazba, při které se využívá elektromagnetické indukce k přenosu energie. Nejčastější, se s transformátorovou vazbou setkáme u síťových transformátorů, jež jsou použity v každém zařízení, kde je potřeba pracovat se sníženým napětím. To znamená, že jsou použity ve veškeré domácí elektronice. Jejich výhodami jsou relativně jednoduchá konstrukce, celkem malé rozměry a účinnost pohybující se až k 90 %. Nevýhodami často bývá uvolnění transformátorových plechů a jejich rozkmitání, jež způsobuje nepříjemné bzučení. Dalšími nevýhodami je nutnost chlazení a poměrně nízký rozsah pracovních napětí a výkonů. Největší nevýhodou je ovšem to, že transformátor lze použít pouze pro střídavé nebo pulzační napětí, stejnosměrné napětí nelze transformovat.

Další možností galvanického oddělení je optická vazba. U optické vazby se využívá světelného zdroje, jako je například LED dioda, který nám pomocí světelného toku předává informaci na další součástku, jež může být fototranzistor, fotorezistor či fotodioda. Vysílač i přijímač jsou společně zakomponovány do jednoho pouzdra a představují společnou součástku. Na obr. 2.13 je zobrazeno blokové schéma součástky a jeho pouzdro.



Obr. 2.13 – Blokové schéma a pouzdro optočlenu

Výhody optočlenů jsou v malých rozměrech, vysoké účinnosti, bezhlučném provozu a v možnosti přenosu na velkou vzdálenost. Používají se tam, kde data opouští zařízení a přichází do styku s jiným zařízením, které je napájeno z vlastního zdroje elektrické energie. Jako typický příklad galvanického oddělení můžeme uvést dálkové ovládání a televizor. V naší práci je požadováno galvanické oddělení napájení, digitálních výstupů a budičů sběrnic CAN a RS-232.

Digitální výstupy:

U digitálního výstupu je použita optická vazba. Využívá se optočlenu s označením 4N35 více informací se dozvíme v dokumentech výrobce [7]. U digitálních výstupů není kladen požadavek na rychlost přenosu, jelikož pouze simulují jednoduché mechanické přepínače.

Sběrnice CAN a RS-232:

Pro budiče sběrnic je ovšem kladen požadavek na rychlost přenosu. Na sběrnice je potřeba rychlý galvanický oddělovač, kvůli maximální frekvenci přenosu dat. U sběrnice CAN je maximální teoretická rychlost přenosu 1 MHz odpovídá 1 Mb/s. Pro linku RS-232 je postačitelná rychlost 57 600 Hz, která odpovídá hodnotě 57 600 Bd, (Bd - Baud je jednotka modulační rychlosti udávající počet změn stavu přenosového média za jednu sekundu).

Z těchto důvodů je nutno vybrat dosti rychlý galvanický oddělovač. V našem modulu řízení otáček ventilátoru je zvolen oddělovač s názvem ADUM od firmy ANALOG DEVICE. Je to galvanický oddělovač s využitím vysokorychlostních CMOS obvodů a monolitickou vzduchovou technologii jádrového transformátoru. Vstupy jsou ošetřeny Schmittovým klopným obvodem. ADUM vykazuje lepší vlastnosti než klasické optočleny, výhodou je také absence dalších součástek pro správnou funkci. Vyráběny jsou varianty s různým počtem a konfigurací vstupů a výstupů [12] [13]. Pro oddělení budiče CAN je v práci využit ADUM 1301 a pro oddělení budiče RS-232 je využit ADUM 1201.

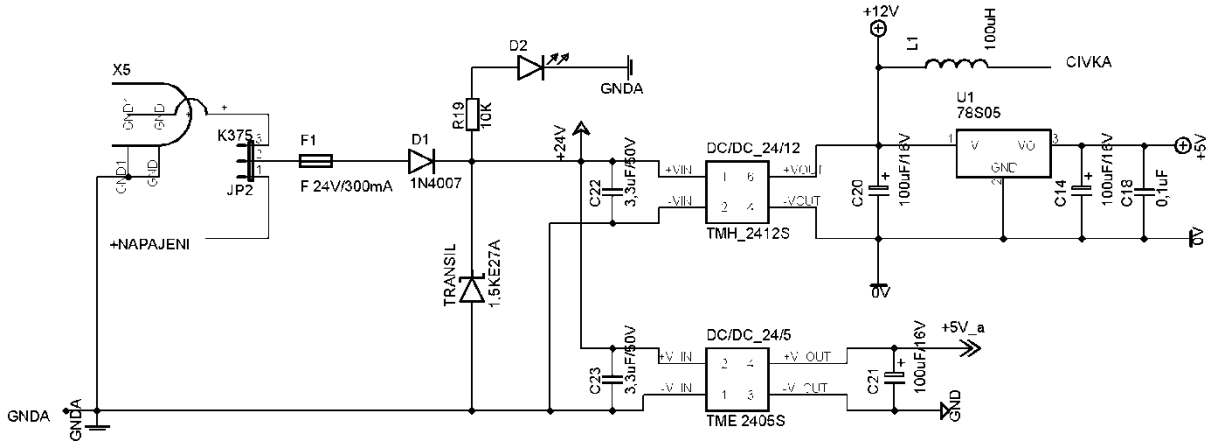
Obr. 2.14 – Funkční blokové schéma a) ADUM 1200 b) ADUM 1201

Napájení celého modulu řízení otáček ventilátoru musí být také galvanicky odděleno. Nejvhodnější bude pro galvanické oddělení napájení přípravku zvolit vhodné DC-DC měniče, které nám splní požadavek galvanického oddělení, mají malé konstrukční rozměry a zároveň nám dále přizpůsobí napětí na jinou, námi požadovanou hodnotu. Více se budeme zaobírat problematikou DC-DC měničů v kapitole 2.2.8 Napájení.

Pro rekapitulaci si uvedeme přehled galvanického oddělení:

- CAN - 2x vstup (TX, RS), 1x výstup (RX)
- RS-232 - 1x vstup (TX), 1x výstup (RX)
- Digitální výstup 4x
- Napájení

2.2.8 Napájení



Obr. 2.15 – Napájení modulu řízení otáček ventilátoru

Energii potřebnou pro modul řízení otáček ventilátoru lze získávat buď přímo z frekvenčního měniče napětí po aktivaci pomocného napájení (vice v příloze 1) nebo z externího zdroje elektrické energie, pro který je na desce plošných spojů umístěn vhodný konektor. Souosý konektor nese označení K375A. Abychom mohli napájet modul z frekvenčního měniče, je nutné nepřekročit odebíraný proud 300 mA. To je totiž maximální proud, který lze z pomocného napájení frekvenčního měniče odebírat. Na tuto hodnotu je i dimenzovaná pojistka. Dále je v cestě umístěna polovodičová dioda, jako ochrana před přepólováním. Následně je zde zapojen

transil, který nám chrání celé zařízení před napěťovými špičkami. Je zde první odbočka na napájení optočlenů s hodnotou 24 V.

Pro galvanické oddělení vstupního napětí je využito DC-DC měničů, abychom mohli zvolit vhodné parametry měničů, je nutné nejprve spočítat odběr jednotlivých větví zařízení. Následně vypočítat výkon a dle toho vybrat vhodný měnič.

První skupina součástek napájena z DC-DC měniče 24 V / 12 V:

Součástka	Odběr
Adum 1301 a 1201	5,5 mA (1,1 mA/kanál)
MCU AT90CAN128	30 mA
D/A převodník DAC8552	1,39 mA
5 V reference REF5050	1,2 mA
Stabilizátor napětí 78S05	6 mA
4 x optočlen 4N35	40 mA (4x 10 mA)
Přijímač dálkového ovládání	9 mA
Operační zesilovač TLV 2372	2 mA
Celkový odběr	**95,09 mA**

Tab. 2.3 – Tabulka odběru první větve součástek

Druhá skupina součástek napájena z DC-DC měniče 24 V / 5 V:

Součástka	Odběr
Adum 1301 a 1201	5,5 mA (1,1 mA/kanál)
Budič CAN PCA82C250	70 mA
Budič RS-232 MAX232CWE	20 mA
Celkový odběr	**99,5 mA**

Tab. 2.4 – Tabulka odběru druhé skupiny součástek

Po zjištění proudového odběru jednotlivé skupiny součástek je nutné spočítat výkon jednotlivé větve, jež bude požadován po DC-DC měniči. Výstupní napětí první větve činí 12 V a druhé větve 5 V. Výkon spočteme ze vztahu 2.2.

$$P = U \cdot I \ [W] \tag{2.2}$$

Kde:

P – Výkon

U – Napětí

I – Proud

Po dosazení do vztahu 2.2 nám výkon první větve vyšel 1,141 W a výkon druhé větve 0,498 W. Musíme počítat ještě s určitou proudovou rezervou jednotlivých větví, proto byl zvolen pro první větev DC-DC měnič s výkonem 2 W a s označením TMH 2412S. Druhá větev byla osazena měničem TME 2405S o výkonu 1 W. Oba měniče jsou značky TRACO POWER více informací se dozvíme z materiálů od výrobce. [19][20]

V první napájecí větvi, kde je napětí již pomocí DC-DC měniče změněno na hodnotu 12 V kvůli napájení přijímače dálkového ovládání, je dále umístěn stabilizátor napětí na 5 V, který napájí součástky uvedené v tab. 2.3. Tento stabilizátor má označení 78S05. [10]

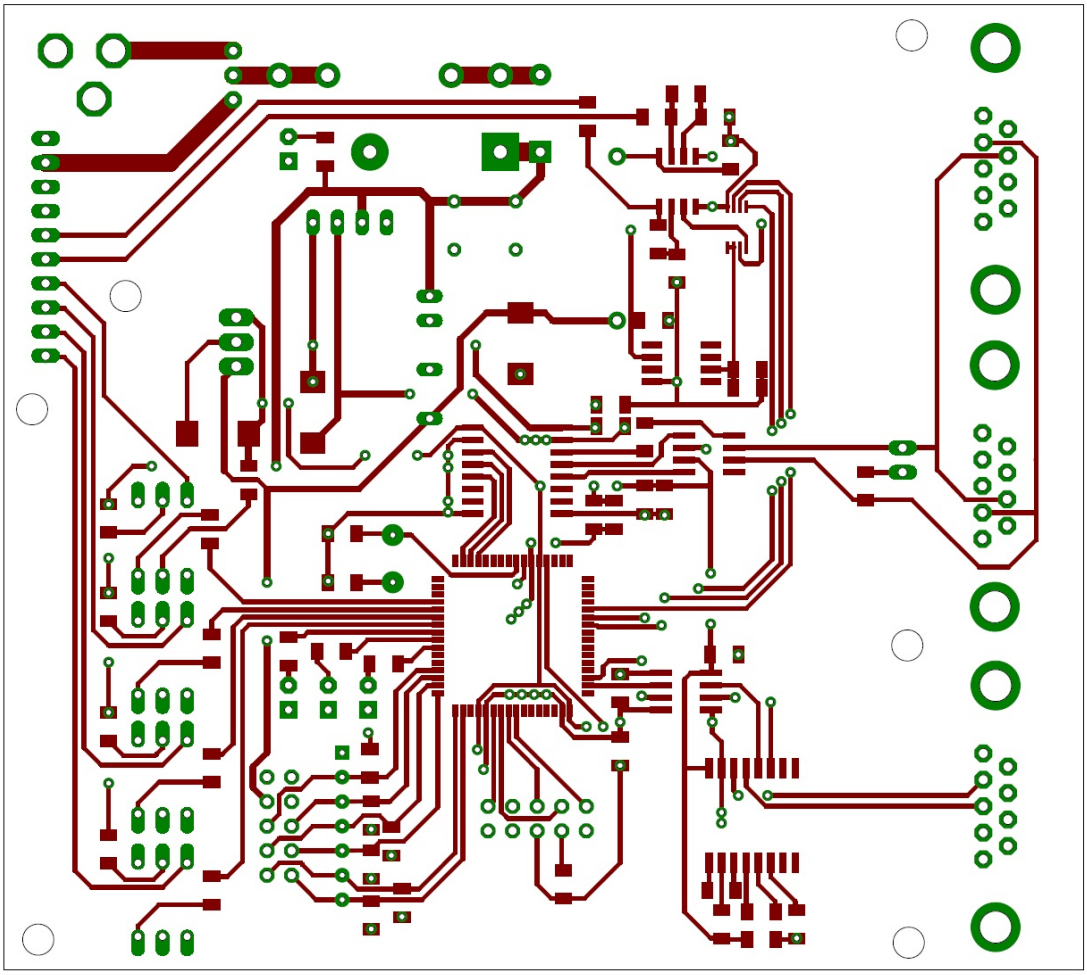
2.3 Stavba prototypu

2.3.1 Tvorba desky plošných spojů

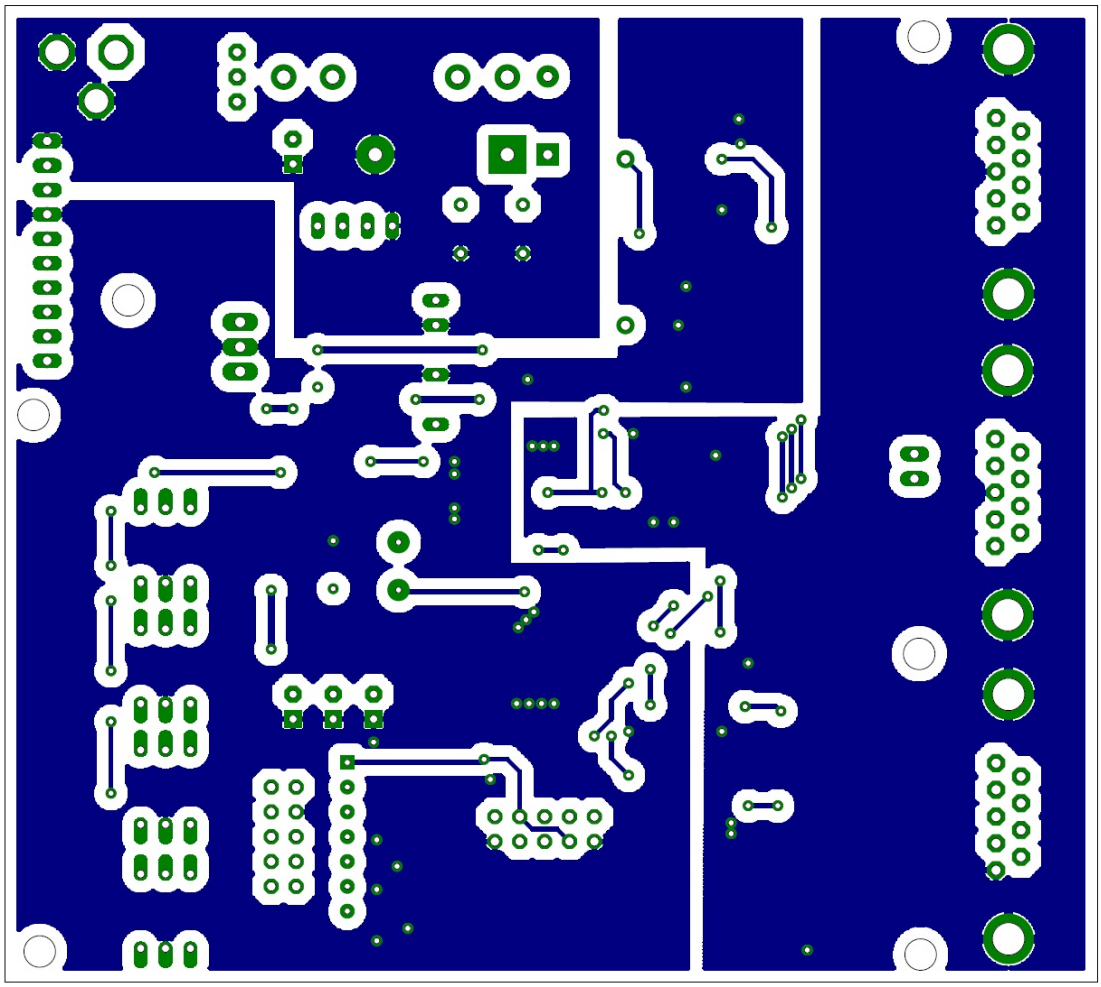
V první řadě bylo potřeba vytvořit návrh desky plošných spojů (DPS). Pro návrh DPS bylo využito vývojového softwaru EAGLE od společnosti CadSoft Computer, Inc.

Vzhledem k zjednodušení celé konstrukce byla navržena oboustranná DPS. V jedné vrstvě jsou tři zemní polygony a v druhé je většina zbývajících spojů. Součástky, jež je DPS osazena jsou ve větší míře typu SMD, díky tomu dochází k minimalizaci celého modulu. Návrh DPS byl komplikovaný především proto, že bylo nutno vytvořit tři zemní polygony. Jeden pro napájení další pro analogovou část a poslední pro digitální část. Návrh DPS byl odeslán do firmy PragoBoard s.r.o., kde byla deska profesionálně vyrobena, neboť v podmínkách DFJP by nebylo možné dosáhnout tak profesionální úrovně zpracování. Na následujících obrázcích jsou zobrazeny jednotlivé vrstvy DPS, osazovací plán a výsledné zkompletování DPS s osazenými součástkami.

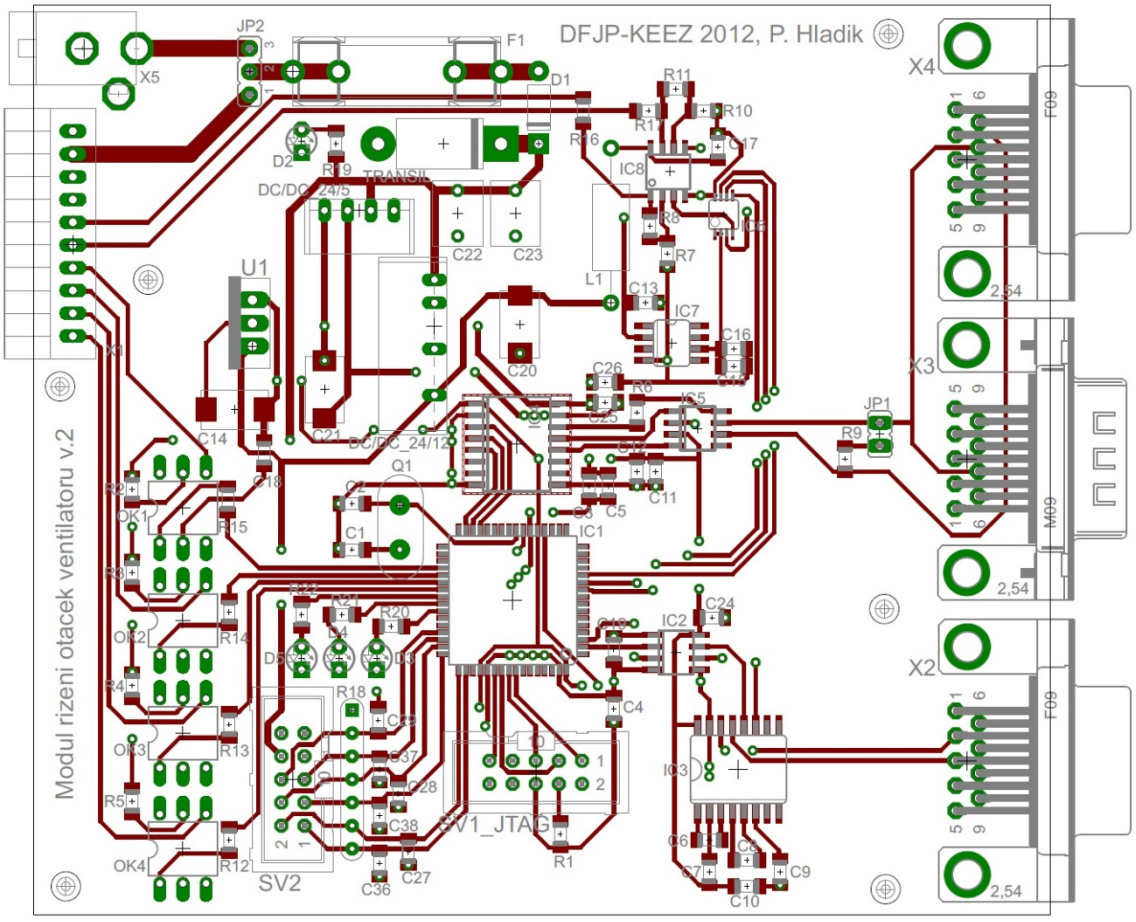
Po vytvoření DPS a jejím zkompletování, se osadí do vhodné krabičky. Výrobců, kteří se věnují výrobě plastových krabiček je na trhu dosti a tak nebyl problém sehnat vhodnou, právě pro náš modul. Naše krabička má označení KP 17 a je vyráběna firmou A&A, výroba, obchod a servis, s.r.o. [21]. Při návrhu DPS bylo nutné brát ohledy na rozměry zvolené krabičky, tak aby vyrobená DPS pasovala do vybrané krabičky.



Obr. 2.16 – DPS modulu řízení otáček ventilátoru – horní vrstva (top)



Obr. 2.17 - DPS modulu řízení otáček ventilátoru – spodní vrstva (bottom)

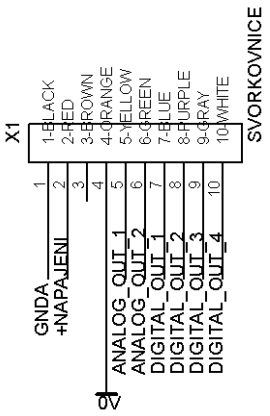


Obr. 2.18 - DPS modulu řízení otáček ventilátoru – osazovací plán

Obr. 2.19 – Modul řízení otáček ventilátoru po zkompletování

2.3.2 Spojení modulu a frekvenčního měniče

Modul řízení otáček ventilátoru byl osazen vhodným konektorem, aby bylo možné ho následně spojit s frekvenčním měničem. Ve schématu je tento konektor označen jako X1. Konektor je znázorněna na obr. 2.20.



Obr. 2.20 – Hlavní konektor

Vhodným spojením modulu řízení otáček ventilátoru a frekvenčního měniče docílíme správného ovládání měniče. Rozpis spojení jednotlivých pinů modulu a měniče je v následující tabulce:

Modul řízení otáček ventilátoru		**Frekvenční měnič**	
PIN modulu	Barva vodiče konektoru	PIN měniče	Význam pinu
1	BLACK (0V)	28	0V
2	RED (+24V)	33	ENC+ Supply
4	ORANGE (A0V)	2	0V (GND analogových vstupů)
5	YELLOW (AO1)	3	AI0 (žádaná frekvence [%])
6	GREEN (AO2)	10	AI1
7	BLUE (DO1)	5	DI0 (ON/OFF)
8	PURPLE (DO2)	6	DI1 (REVERZACE)
9	GREN (DO3)	7	DI2
10	WHITE (DO4)	8	DI3
Poznámka: Piny měniče číslo 4(AI0-), 11(AI1-), 2(0V) je nutné spojit, jedná se o GND přípravku.			

Tab. 2.5 – Spojení modulu řízení otáček ventilátoru a frekvenčního měniče

3 Firmware

Po-té co je hardware kompletně vyroben je třeba vytvořit vhodný firmware pro správnou funkci modulu řízení otáček ventilátoru. V této kapitole se zaměříme na tvorbu firmwaru, jež je pro modul použit. Při výběru MCU byl jeden z požadavků na vhodné programování ve vyšším programovacím jazyce. V našem případě byl vybrán programovací jazyk C. V současné době, je to jeden z nejpoužívanějších programovacích jazyků. Tento programovací jazyk byl také vybrán z důvodu částečné znalosti programovacího jazyka C a pro můj osobní rozvoj ve znalosti tohoto jazyka.

3.1 Volba programovacího prostředí

Pro programovací jazyk C existuje spousta vývojových prostředí. Jelikož je náš modul osazen MCU s označením AT90CAN128 a s architekturou AVR od výrobce ATMEL, tak bylo vybráno vývojové prostředí CodeVisionAVR, které je doporučováno výrobcem našeho MCU. Další výhodou je to, že Katedra elektrotechniky, elektroniky a zabezpečovací techniky (KEEZ) vlastní licenci k tomuto programu a tak bylo možné ho plně využít. Protože toto vývojové prostředí neobsahuje debugger, který se používá pro odladění programu a je tak možné ihned vidět místo, kde se objevila programátorská chyba, tak pro následnou kontrolu firmwaru byl využit program AVR studio, jehož licenci KEEZ opět vlastní. Tento program nám také umožnil nahrát vytvořený firmware do MCU pomocí rozhraní JTAG.

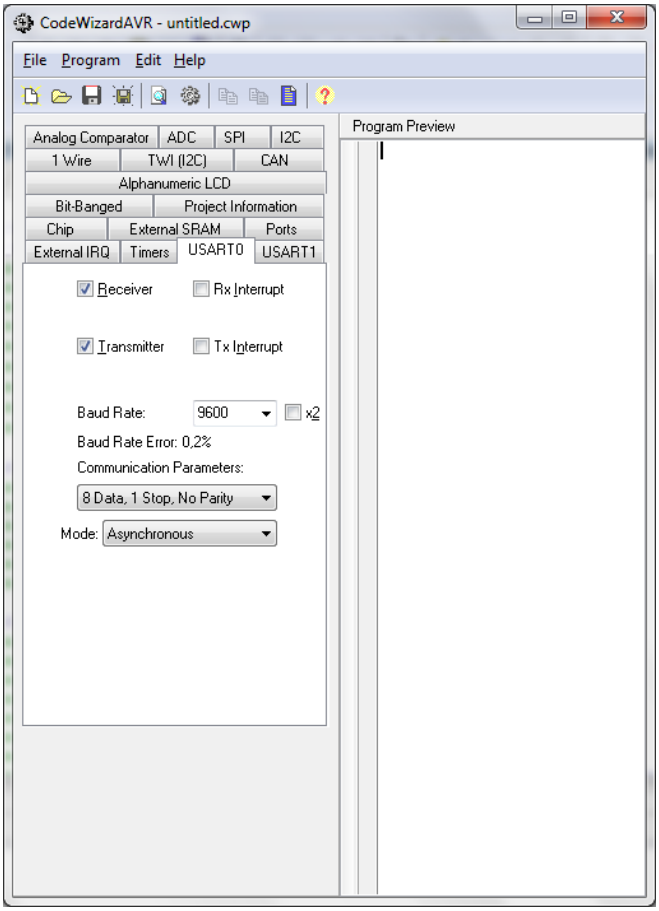
3.2 Nastavení MCU

Po spuštění CodeVisionAVR a založení nového projektu je nutné nastavit základní vlastnosti chování MCU. Prvním parametrem, který je nutné nastavit je typ našeho MCU, tedy AT90CAN128. Další z nastavovaných parametrů je zdroj taktovacího kmitočtu a jeho frekvence. V našem případě byl zvolen externí oscilátor s frekvencí 16 MHz.

Lze nastavit už jednotlivé piny portů MCU, zda by měly být vstupní či výstupní a jejich chování. Například je možné pro nastavení logické hodnoty vstupu zapnout vnitřní pull-up a tím vybrat stav logické jedničky na příslušném pinu portu. Dále je nutné vybrat vhodné nastavení pro sériové periferní rozhraní (SPI), po kterém

budeme komunikovat s okolními periferiemi jako je D/A převodník. Pro nastavení komunikace po sériové lince (USART) musíme provést také vhodné nastavení.

Tyto nastavení nám usnadní pomůcka CodeWizardAVR, která nám pomocí svého graficky příjemného uživatelského prostředí umožní jednodušší a přehlednější nastavení všech periferií. Všechna tato nastavení lze provést také přímým zápisem do zdrojového kódu programu.



Obr. 3.1 – Nastavení USART0

Po nastavení jaké je zobrazeno na obr. 3.1 nám program vygeneruje takovouto část zdrojového kódu, která je vhodná pro naši komunikaci po sériové lince:

```
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;
```

3.3 Důležité části firmwaru

Firmware je řazen do jednotlivých částí a logických funkcí, aby byl co nejvíce přehledný. Zdrojový kód je komentován tak, aby bylo na první pohled zřejmé, kde začínají jednotlivé funkce a aby alespoň částečně popsal význam zapsaného kódu. Některé uvedené důležité části si nyní blíže popíšeme.

3.3.1 Definice vstupů/výstupů

Pro usnadnění a zpřehlednění celého firmwaru si pro začátek nadefinujeme symbolické konstanty (makra bez parametrů). Na MCU portu A je připojen modul ovládání ventilátoru, na portu B je D/A převodník, k portu C jsou připojeny digitální výstupy a LED diody pro signalizaci, na port D je připojen budič sběrnice CAN a pomocí portu E komunikujeme po sériové lince. Pro všechny tyto porty byly zvoleny vhodné symbolické konstanty. Následný kód nám zobrazuje definování konstant pro port A, kde je připojen přijímací modul dálkového ovládání:

```
#define TLAC_PIN        PINA
#define TLAC_PORT       PORTA
#define TLAC_DDR        DDRA
#define TLAC_ZAP        1
#define TLAC_VYP        2
#define TLAC_SMER       3
#define TLAC_OVL        4
#define TLAC_OTAC_N     5
#define TLAC_OTAC_D     6
```

Jednotlivé příkazy nám definují, o jaké tlačítko se jedná a na jaký pin portu A je tlačítko připojeno. Takže například `TLAC_SMER` nám říká, že jde o tlačítko, kterým se volí směr otáčení ventilátoru a číslo 3 nám určuje, že je tlačítko připojeno k pinu 3.

Následně je vhodné definovat makra s parametry (dále pouze makra). Při řešení programů se často vyskytne případ, kdy mnohokrát používáme funkci, která je velmi krátká. Takovou funkci lze bez problémů napsat, lepší variantou je ale použití maker. V našem případě jsme využili maskování pro nastavení maker. Maskování je způsob nastavení vybraných bitů registru na hodnotu logické 1 (`|= (1<<DO1)`), nebo 0 (`&= ~(1<<DO1)`). Následující výpis kódu nám zobrazuje nastavení maker pro ovládání digitálních výstupů a signalizačních LED diod.

```
#define MENIC_ZAP        (DO_PORT |= (1<<DO1))
#define MENIC_VYP        (DO_PORT &= ~(1<<DO1))
#define MENIC_SMER_L     (DO_PORT |= (1<<DO2))
#define MENIC_SMER_R     (DO_PORT &= ~(1<<DO2))
#define LED_OVL_ZAP      (DO_PORT |= (1<<LED_OVL))
#define LED_OVL_VYP      (DO_PORT &= ~(1<<LED_OVL))
#define LED_CAN_RX_ZAP   (DO_PORT |= (1<<LED_CAN))
#define LED_CAN_RX_VYP   (DO_PORT &= ~(1<<LED_CAN))
#define LED_SIGNAL_ZAP   (DO_PORT |= (1<<LED_SIGNAL))
#define LED_SIGNAL_VYP   (DO_PORT &= ~(1<<LED_SIGNAL))
```

Inicializace jednotlivých vstupních a výstupních portů bylo možné nastavit pomocí CodeWizardAVR. V našem případě tak učiněno nebylo a proto je nutné inicializovat porty nyní. Následující část zdrojového kódu inicializuje jednotlivé porty. Opět bylo využito maskování.

```
// tlačítka
TLAC_DDR = 0x00;            // cely port vstupni
TLAC_PORT = 0xFF;           // aktivace pull-upu

// D/A prevodnik
DAC_DDR |= (1<<MOSI) | (1<<SS) | (1<<SCK);
DAC_PORT |= (1<<SS);

// digitalni vystupy + LED
DO_DDR = 0xFF;              // cely port vystupni
DO_PORT = 0x00;             // vystupy neaktivni

// CAN
CAN_DDR |= (1<<CAN_TX) | (1<<CAN_RS);
CAN_PORT = 0x00;

// RS-232
RS232_DDR |= (1<<RS232_TX);
RS232_PORT = 0x00;
```

Je potřebné také nastavit všechny globální a lokální proměnné, se kterými se bude v programu pracovat. Podrobné nastavení všech konstant, globálních proměnných, lokálních proměnných, maker a vstupně/výstupních portů nalezneme v příloze 4 této práce.

3.3.2 Pomocné funkce

V našem programu využíváme i několik námi vytvořených pomocných funkcí, které si nyní blíže popíšeme. Deklarace jednotlivých lokálních funkcí vypadá následně:

```
unsigned char CtiTlacitka (void);
unsigned int PrepocetProcentoCislo (unsigned char procento);
void WriteToDAC(unsigned char channel, unsigned int counts);
void ProcessCanInterrupt(void);
unsigned char PrevedRychlostNaOtacky(unsigned char rychlost);
```

Funkce `CtiTlacika()` nám zajišťuje zjištění stavu tlačítek na dálkovém ovladači. Ošetřuje nám zákmity, které mohou vznikat při stisku tlačítka a vrací hodnotu stisknutého tlačítka. Stisknutému tlačítku odpovídá hodnota logická 1. `PrepocetProcentoCislo()` čte požadovanou hodnotu otáček ventilátoru v procentech a přepočítává ji na číselnou hodnotu v rozmezí od 0 do 65 535. Tuto vypočtenou hodnotu následně pomocí funkce `WriteToDAC()` odesíláme do D/A převodníku. Funkce `ProcessCanInterrupt()` má na starosti obsluhu události od řadiče CAN. V našem případě nás zajímá pouze příjem zprávy, která nese informaci o aktuální rychlosti měřeného objektu. Poslední funkce `PrevedRychlostNaOtacky()` nám obstarává převod rychlosti, kterou přijmeme pomocí předchozí funkce `ProcessCanInterrupt()`, na požadované otáčky ventilátoru. Pro názornost je zde zobrazena část kódu s funkcí `PrepocetProcentoCislo()`.

```
unsigned int PrepocetProcentoCislo (unsigned char procento)
{
    unsigned int vypocet;
    vypocet =(unsigned int)(((unsigned long) procento * 65535)/100);
    return vypocet;
}
```

3.3.3 Hlavní část programu

Hlavní funkce program se jmenuje `main()`. Je to první funkce, která je volána po spuštění programu. Sestává-li program z více částí, `main()` smí být uveden pouze v jednom modulu a v tomto modulu také jenom jednou. V našem případě celý program sestává ze dvou modulů. *CanAvr.c* a *CanAVR.h* mají na starosti správnou funkci komunikace s řadičem CAN, zajišťují příjem a vysílání zprávy po CAN. Hlavní modul celého programu je pojmenován *Modul_rizeni_otacek_ventilatoru.c*.

Celé řízení ventilátoru je umístěno v nekonečné smyčce v hlavní funkci `main()`. Tělo této smyčky nám zajišťuje vhodnou reakci na stisk tlačítek dálkového ovládání, dále také výpis hodnot po sériové lince a ovládá chod celého programu.

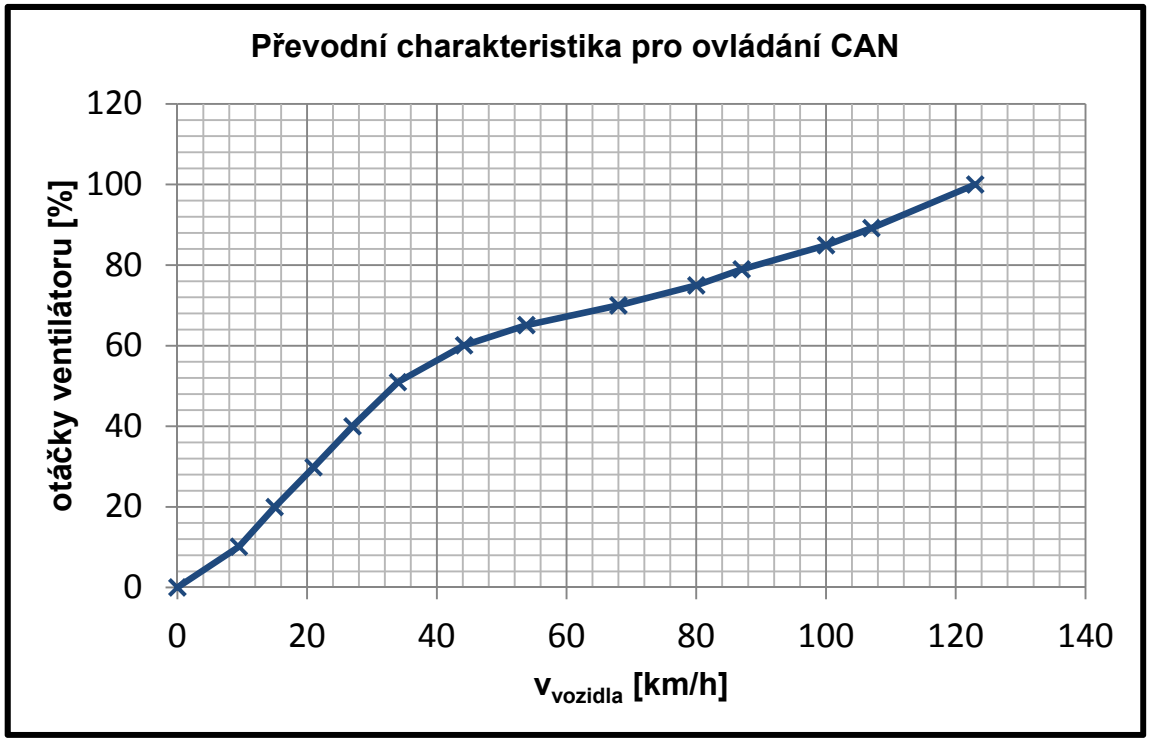
3.3.4 Komunikace po sběrnici CAN

Identifikátor zprávy, který nese informaci o rychlost měřeného objektu v km/h pro další zpracování MCU, byl nastaven na hodnotu 10 hexa. Jelikož celý systém dynamometru není kompletně dokončen a stále se na něm pracuje, identifikátor zprávy se může do budoucna změnit. V našem případě je využito pouze příjmu zprávy po sběrnici CAN, právě pro zjištění rychlosti měřeného objektu. Následující část zdrojového kódu provádí nastavení identifikátoru zprávy a nastavení příjmu zprávy po CAN.

```
#define CAN_MSG_RX1 0x10    // ID zpravy s rychlosti vozidla

canMsg = MakeCanMsg(RX_MODE,CAN_MSG_RX1,0x7FF,0,1,canData);
// prijem, ID = CAN_MSG_RX1, CAN2.0A, 1 bajt dat
mobNo = CanSetRx(canMsg);
printf("Nastaven MOB%d RX pro prijem zpravy \tID=0x%X\n", mobNo,
CAN_MSG_RX1);
delay_ms(1000)
```

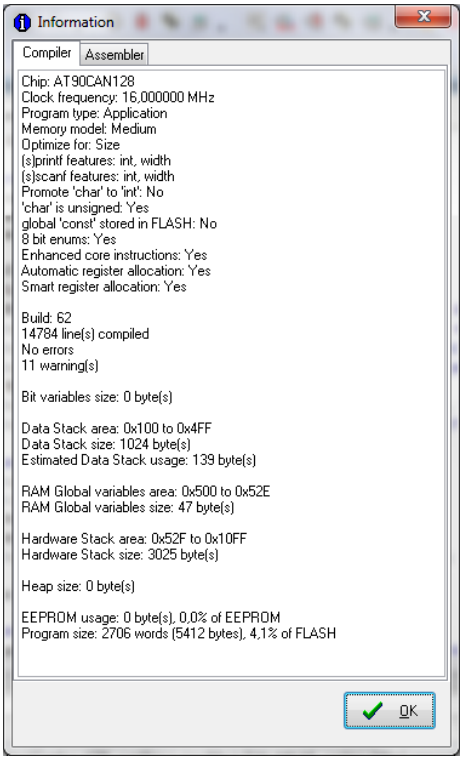
Ve firmwaru, v části pro ovládání ventilátoru pomocí sběrnice CAN, je implementována převodní charakteristika žádaných otáček ventilátoru v %, jež odpovídá hodnotě rychlosti vozidla. Při změně rychlosti měřeného objektu se budou otáčky ventilátoru pohybovat právě po této křivce. Převodní charakteristika je zobrazena na obr. 3.2.



Obr. 3.2 – Převodní charakteristika pro ovládání CAN

3.3.5 Kompilace programu

Po zapsání celého programu je čas na zkompilování a nahrání do MCU. Po kompilaci programu se nám zobrazí informační tabulka, která nám zobrazuje základní informace o výsledku kompilace a linkování programu. Zobrazí nám typ MCU pro který byl program napsán, hodinovou frekvenci, velikost programu, kolik z celkové paměti MCU nám náš program zabere a mnoho dalších informací. Informační tabulka je zobrazena na obr. 3.3. Společně s kompilací programu se nám vytvoří, kromě mnoha jiných souborů, soubor s příponou *hex*. Tento soubor musíme nahrát do našeho MCU pro jeho správnou funkci. K tomu nám poslouží program AVR studio. Hlavní část programu naleznete v příloze 4 této bakalářské práce. Kompletní zdrojový kód a soubory, jež jsou vytvořeny kompilátorem a nahrané v MCU naleznete na přiloženém CD-ROM nosiči.



Obr. 3.3 – Informační tabulka kompilátoru

4 Ověření funkce modulu

K ověřování funkce jednotlivých komponent a firmwaru docházelo v průběhu celého vývoje modulu řízení otáček ventilátoru. Je ovšem nezbytné seznámit se s ovládáním a provést několik měření při reálných podmínkách v provozu zařízení.

4.1 Ovládání

K ovládání modulu řízení otáček ventilátoru slouží dálkový ovladač, pomocí něhož nastavujeme potřebné chování modulu. Dálkové ovládání má šest tlačítek, jejichž rozložení, je zobrazeno na obr. 4.1. Nyní si popíšeme význam jednotlivých tlačítek:

- Zapnuto – Zapíná ventilátor
- Vypnuto – Vypíná ventilátor
- Směr otáčení – Volí směr otáčení ventilátoru doleva/doprava
- Zdroj ovládání – Volí zdroj ovládání otáček ventilátoru dálkové ovládání/CAN
- Zvýšení otáček – Zvyšuje otáčky ventilátoru
- Snížení otáček – Snižuje otáčky ventilátoru

Obr. 4.1 – Popis dálkového ovladače

Pokud je zvolen zdroj ovládání pomocí sběrnice CAN nelze zvyšovat a snižovat otáčky ventilátoru, řízení otáček je totiž plně automatizováno a je závislé na rychlosti měřeného objektu.

Modul řízení otáček ventilátoru je osazen také čtyřmi LED diodami, které mají za účel informovat uživatele o aktuálním nastavení. Jedna slouží pro zjištění napájení modulu otáček ventilátoru. Druhá má za úkol informovat uživatele o stisku

tlačítka dálkového ovladače, kdy se při každém stisknutí jakéhokoliv tlačítka rozsvítí. Třetí nám signalizuje příjem zpráv po sběrnici CAN. Při každém přijmu zprávy se, tato dioda rozsvítí. Poslední dioda nám určuje, zda je zvoleno ovládání pomocí sběrnice CAN, to nám dioda svítí, či je zvoleno ovládání z dálkového ovladače, to dioda nesvítí. V příloze č. 3 naleznete fotografie modulu řízení otáček ventilátoru.

4.2 Funkční zkoušky

Po zkompletování modulu, nahrání firmwaru a osazení do krabičky jsme mohli přistoupit k samotnému testování. Předpokladem je již vhodné naparametrování frekvenčního měniče, kterému jsme se věnovali v kapitole 1.3.2 a spojení modulu s měničem, aby ho bylo možné ovládat pomocí analogových a digitálních vstupů na měniči, viz kapitola 2.3.2. Ověření ovládání pomocí dálkového ovladače bylo v pořádku. Modul reagoval na všechna tlačítka dálkového ovladače adekvátně. Signalizace pomocí LED diod byla také v pořádku.

Obr. 4.2 – Modul řízení otáček ventilátoru

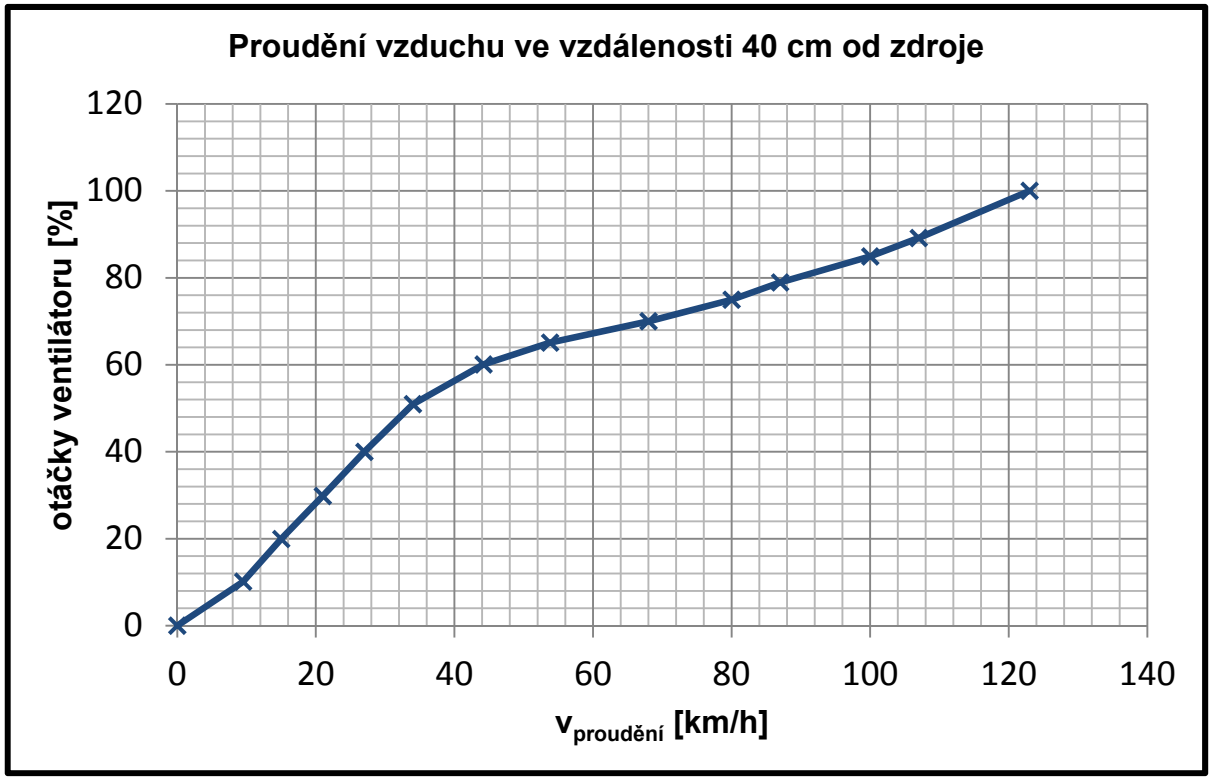
4.2.1 Měření rychlosti proudění vzduchu

Následně jsme provedli měření rychlosti proudění vzduchu z ventilátoru, ve vzdálenosti 40 cm od ventilátoru. K tomuto měření nám posloužil anemometr. Měření rychlosti proudění vzduchu probíhalo ve třech vzdálenostech od ventilátoru. V prvním případě byl anemometr umístěn přímo na ventilátoru, v druhém případě ve vzdálenosti 40 cm od ventilátoru a v posledním 80 cm od ventilátoru. Při měření rychlosti na ventilátoru a ve vzdálenosti 40 cm, byla rychlost proudění téměř totožná, ovšem ve vzdálenosti 80 cm od ventilátoru proudění ztrácelo na rychlosti dost výrazně a to tak, že rychlost byla o polovinu menší. Z toho plyne, že při provozu je nutné mít ventilátor co nejblíže chlazeným místům, tak aby se dosáhlo co nejefektivnějšího chlazení měřeného objektu.

V tabulce 4.1 jsou uvedeny naměřené hodnoty proudění vzduchu ve vzdálenosti 40 cm od ventilátoru. Hodnota $v_{proudění}$ nám udává rychlost proudění vzduchu z ventilátoru, $f_{řízení\ motoru}$ je frekvence řízení motoru a otáčky ventilátoru vyjadřují procentuální otáčky ventilátoru, kde 100 % otáček je jmenovitá hodnota motoru (v našem případě 50 Hz). Pro zpřehlednění naměřených dat je zde také uveden graf závislosti rychlosti proudění vzduchu na otáčkách ventilátoru obr. 4.3.

$v_{proudění}$ [km/h]	$f_{řízení\ motoru}$ [Hz]	otáčky ventilátoru [%]
0	0	0
10	5,06	10
15	9,97	20
21	14,91	30
27	19,99	40
34	25,47	51
44	30,04	60
54	32,55	65
68	35	70
80	37,48	75
87	39,47	79
100	42,46	85
107	44,58	89
123	50	100

Tab. 4.1 – Tabulka hodnot při měření 40 cm od ventilátoru



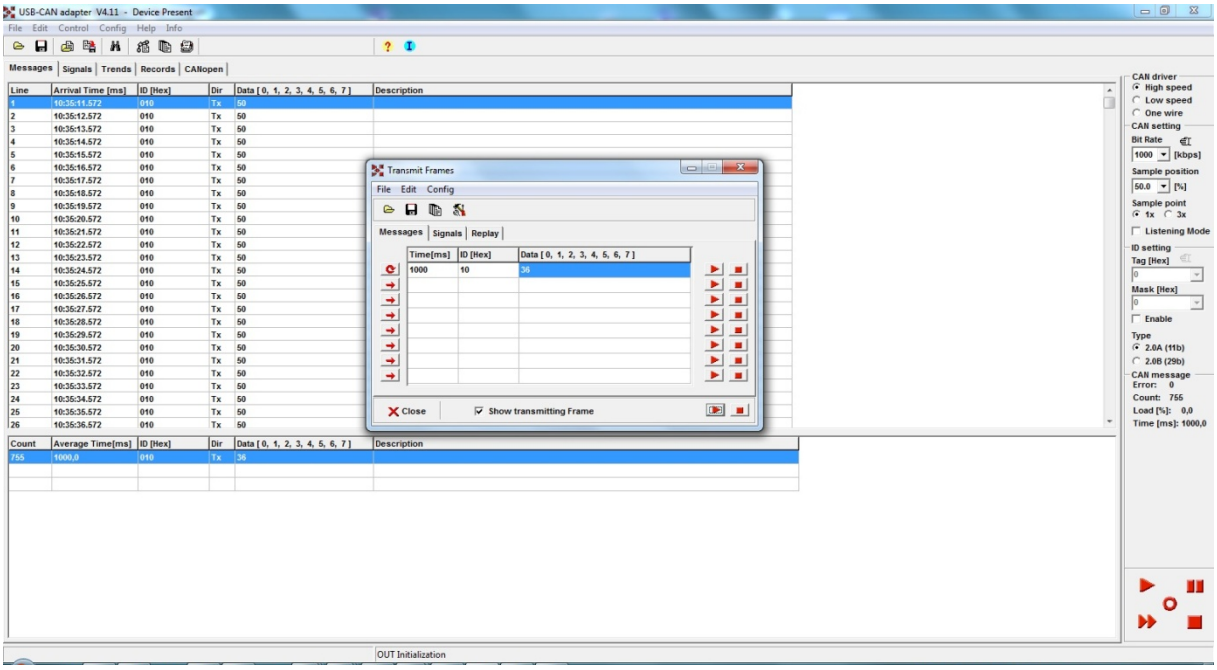
Obr. 4.3 – Graf závislosti rychlosti proudění vzduchu na otáčkách ventilátoru

4.2.2 Ověření komunikace CAN a RS-232

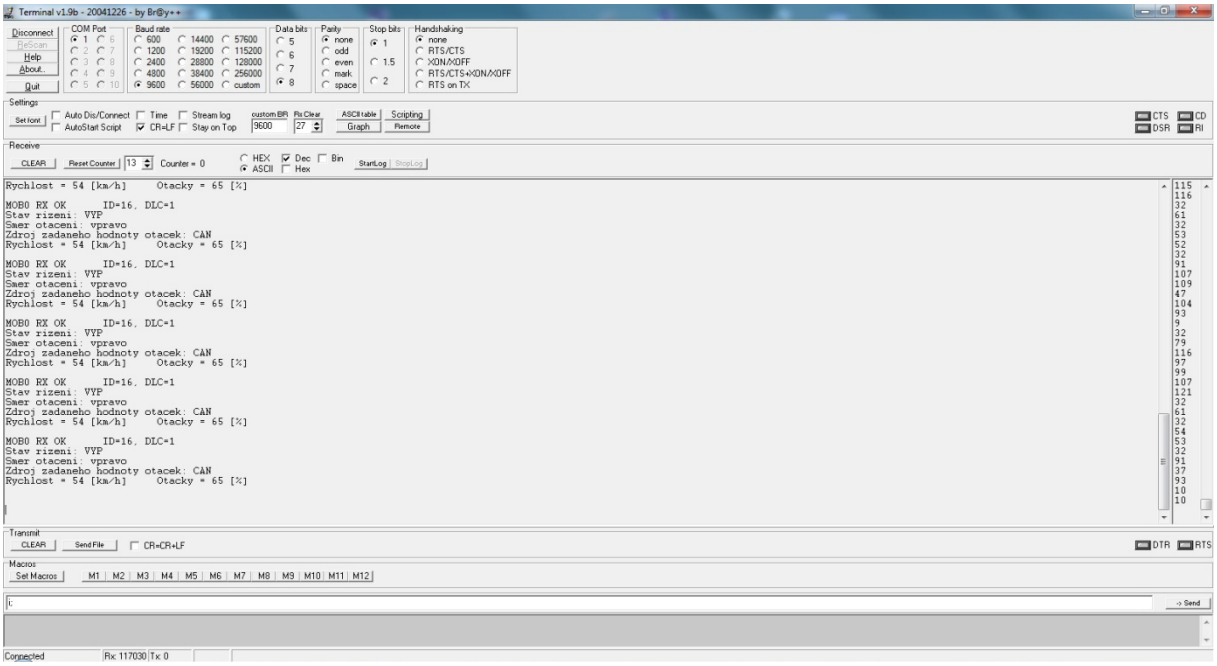
Správnost komunikace modulu řízení otáček ventilátoru s nadřazeným systémem a hlavně příjem zprávy o rychlosti měřeného objektu po sběrnici CAN jsme ověřili pomocí programu CAN analyzátor. Tento program dokáže pomocí převodníku USB / CAN zachytit aktuální datový tok po sběrnici CAN. Pomocí programu CAN analyzátor jsme odeslali informaci o rychlosti jak je vidět na obr. 4.4. Hodnotu rychlosti jsme nastavili na 54 km/h. Do programu CAN analyzátor, musíme zapsat tuto hodnotu v hexadecimálním tvaru. Hodnotě 54 dekadicky, tedy odpovídá hodnota 36 hexadecimálně.

K zobrazení zpracovávaných hodnot a aktuálním nastavení modulu, nám posloužil program TERMINAL, který komunikuje po sériové lince s modulem řízení otáček ventilátoru. Z obr. 4.5 vyplývá, že zpráva s rychlostí měřeného objektu byla modulem správně přijata a zpracována.

Díky zobrazení aktuálních hodnot pomocí programu TERMINAL obr. 4.5, můžeme říci, že komunikace modulu po sběrnici CAN a RS-232 probíhá v pořádku.



Obr. 4.4 – Okno programu CAN analyzátor



Obr. 4.5 – Okno programu TERMINAL

Závěr

Cílem této práce bylo vytvoření modulu řízení otáček ventilátoru k dynamometru, pro zatěžování spalovacích motorů. Modul má za úkol ovládat frekvenční měnič, ke kterému je připojen ventilátor chlazení měřeného objektu (motocykl či motor samotný). Modul musí být zapojen do systému řízení dynamometru pomocí sběrnice CAN.

Před návrhem modulu, bylo nutné nejprve nastudovat ovládání frekvenčního měniče a sběrnici CAN. Z informací týkajících se ovládání frekvenčního měniče vyplynulo, že jeho řízení bude realizováno pomocí analogových a digitálních vstupů. Na základě těchto informací mohly být zvoleny vhodné komponenty, z nichž je modul řízení otáček ventilátoru sestaven.

Požadavky na řízení modulu, jako je směr otáčení ventilátoru, jeho zapnutí či vypnutí, jsou zadávány a vysílány pomocí dálkového ovládání, jež je hlavním ovládacím prvkem modulu. Následně jsou tyto požadavky přijaty přijímačem dálkového ovládání a zpracovány mikrokontrolérem. Ten se stará o zpracování veškerých dat a o následné řízení frekvenčního měniče. Mikrokontorlér je hlavní částí celého modulu. Modul ovládá frekvenční měnič pomocí dvou digitálních a jednoho analogového výstupu. Měnič nám následně ovládá připojený ventilátor. Dva digitální výstupy řídí zapnutí a vypnutí ventilátoru a směr otáčení. Analogový výstup modulu slouží k určení žádané hodnoty otáček ventilátoru. Velikost otáček ventilátoru lze volit pomocí dálkového ovladače ručně, nebo je velikost otáček řízena automaticky pomocí sběrnice CAN, v závislosti na rychlosti měřeného objektu. Čím vyšší rychlost objektu, tím vyšší otáčky ventilátoru. Modul také obsahuje sériový port, který slouží pro komunikaci s osobním počítačem.

Firmware k modulu řízení otáček ventilátoru je napsán v programovacím jazyku C. Pro psaní firmwaru bylo využito vývojové prostředí CodeVisionAVR a pomocí prostředí AVR studio byl firmware nahrán do mikrokontoléru modulu řízení otáček ventilátoru.

Výsledkem mé bakalářské práce je navrhnutý, vytvořený, naprogramovaný a odzkoušený, plně funkční modul řízení otáček ventilátoru.

Seznam použité literatury

[1] NOVÁK, Jaroslav. *Elektromechanické systémy v dopravě a ve strojírenství*. 1. vyd. Praha: ČVUT, 2002, 86 s. ISBN 80-01-02457-1.

[2] Nízkonapěťové motory: Trojfázové asynchronní motory nakrátko 1LA7 osová výška 56 až 160 výkon 0,06 až 18,5 kW. SIEMENS. *Nízkonapěťové motory* [online]. [cit. 2012-01-07]. Dostupné z: http://www.elektromotory.net/upload/file/katalog_1la7.pdf

[3] Automatizace: Měniče frekvence a softstartéry – přehled trhu. [online]. [cit. 2012-01-10]. Dostupné z: http://www.automatizace.cz/article.php?a=1142

[4] LELEK, Tomáš. Měřící ústředna k dynamometru s výstupem na sběrnici CAN. Dopravní fakulta Jana Pernera, 2011. 87 s. Bakalářská práce. Univerzita Pardubice.

[5] HEROUT, Pavel. Učebnice jazykla C. 5. vyd. Praha: KOPP, 2008, 280 s. ISBN 978-80-7232-383-8

[6] JAVŮREK, Jiří. Regulace moderních elektrických pohonů. 1. vyd. Praha: Grada Publishing, 2003, 264 s. ISBN 80-247-0507-9

[7] Fairchildsemi: 4N35. [online]. [cit. 2012-01-11]. Dostupné z: http://www.gme.cz/dokumentace/523/523-006/dsh.523-006.1.pd

[8] Mikromodul přijímače 434MHz s plovoucím kódem MRX6. [online]. [cit. 2012-01-11]. Dostupné z: http://www.flajzar.cz/data/files/515-navod_MRX6.pdf

[9] SIEMENS: SINAMICS G120. [online]. [cit. 2012-04-15]. Dostupné z: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/27069930/CU240S__en-US.pdf?func=cslib.csFetch&nodeid=27069937&forcedownload=true

[10] ST: 78S05. [online]. [cit. 2012-02-02]. Dostupné z: http://www.gme.cz/dokumentace/330/330-018/dsh.330-018.1.pdf

[11] ATMEL: 8-bit Microcontroller, AT90CAN128. [online]. [cit. 2012-01-07]. Dostupné z: http://www.atmel.com/Images/doc7679.pd

[12] ANALOG DEVICES: ADuM1200_1201. [online]. [cit. 2012-02-17]. Dostupné z: http://www.analog.com/static/imported-files/data_sheets/ADuM1200_1201.pd

[13] ANALOG DEVICES: ADuM1300_1301. [online]. [cit. 2012-02-17]. Dostupné z: http://www.analog.com/static/imported-files/data_sheets/ADuM1300_1301.pdf

[14] Texas Instruments: DAC8552. [online]. [cit. 2012-02-13]. Dostupné z: http://www.ti.com/lit/ds/symlink/dac8552.pdf

[15] Texas Instruments: REF5050. [online]. [cit. 2012-02-13]. Dostupné z: http://www.ti.com/lit/ds/symlink/ref5050.pdf

[16] Texas Instruments: TVL2372. [online]. [cit. 2012-02-13]. Dostupné z: http://www.ti.com/lit/ds/symlink/tlv2372.pdf

[17] PHILIPS: PCA82C250. [online]. [cit. 2012-02-12]. Dostupné z: http://www.farnell.com/datasheets/101903.pdf

[18] MAXIM Integrated Products: RS-232 Drivers. [online]. [cit. 2012-03-11]. Dostupné z: http://www.vo.gme.cz/dokumentace/959/959-027/dsh.959-027.1.pdf

[19] TRACO POWER: DC/DC Converters TMH Series, 2 Watt. [online]. [cit. 2012-02-21]. Dostupné z: http://www.farnell.com/datasheets/319836.pdf

[20] TRACO POWER: DC/DC Converters TME Series, 1 Watt. [online]. [cit. 2012-02-21]. Dostupné z: http://www.farnell.com/datasheets/319835.pdf

[21] KRABICKY: KP 17. [online]. [cit. 2012-03-03]. Dostupné z: http://www.krabicky.cz/viewproduct.php?=SID&productid=18

[22] ULN2003A. [online]. [cit. 2012-05-30]. Dostupné z: http://www.gme.cz/dokumentace/380/380-005/dsh.380-005.1.pdf

Seznam obrázků

Seznam tabulek

Seznam příloh

Parametrizace měniče Siemens G120 s ovládacím panelem CU240S PN-F

Účel parametrizace: Nastavit ovládání měniče pomocí analogového signálu žádané hodnoty frekvence a digitálních vstupů ZAP/VYP a směr otáčení.

Způsob řízení ASM:	skalární řízení bez čidla otáček
Metoda ovládání:	ON/OFF1 and REV
	(viz Function_Manual_en-US.pdf str. 155)
Potřebné vstupy na měniči:	2x digitální vstup 24 V, 1x analogový vstup 0-10 V
	DI0 jako ON/OFF1
	DI1 jako REVERSE
	AI0 pro nastavování frekvence
Potřebné ovládací prvky:	2x spínač, 1x zdroj analogového napětí 0-10 V

Rychlá parametrizace (Quick commissioning)

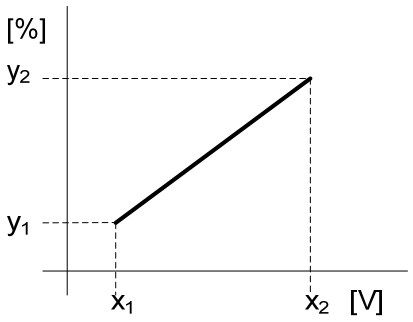
Parametrizace měniče pro připojený motor.

Viz CU240S_en_US.pdf

str. 59 až 60

Nastavení parametrů:

Číslo parametru	Hodnota parametru	Poznámka	Datasheet
P003	3	Nastaví Expert Access Leve – přístup ke všem parametrům	
P0700	2	Selection of command source = „Terminal“	CU240S_en_US.pdf str. 67
P0727	0	Volí metodu ovládání ON/OFF1 and REV	CU240S_en_US.pdf str. 68 a 69 Function_Manual_en-US.pdf str. 155
P0701	1	DI0 jako ON/OFF1	
P0702	12	DI1 jako REVERSE	
P1000	2	Selection of frequency setpoint = „Analog setpoint“	
P0756	0	Unipolar input 0 - 10 V	
P0757	0	x_1 [V]	Převodní charakteristika analogového vstupu, viz Obrázek 1.
P0758	0	y_1 [% vůči ref. hodnotě frekvence uložené v P2000]	
P0759	10	x_2 [V]	
P0760	100	y_2 [% vůči ref. hodnotě frekvence uložené v P2000]	
P3900	0	Způsob konce rychlé parametrizace	CU240S_en_US.pdf str. 60

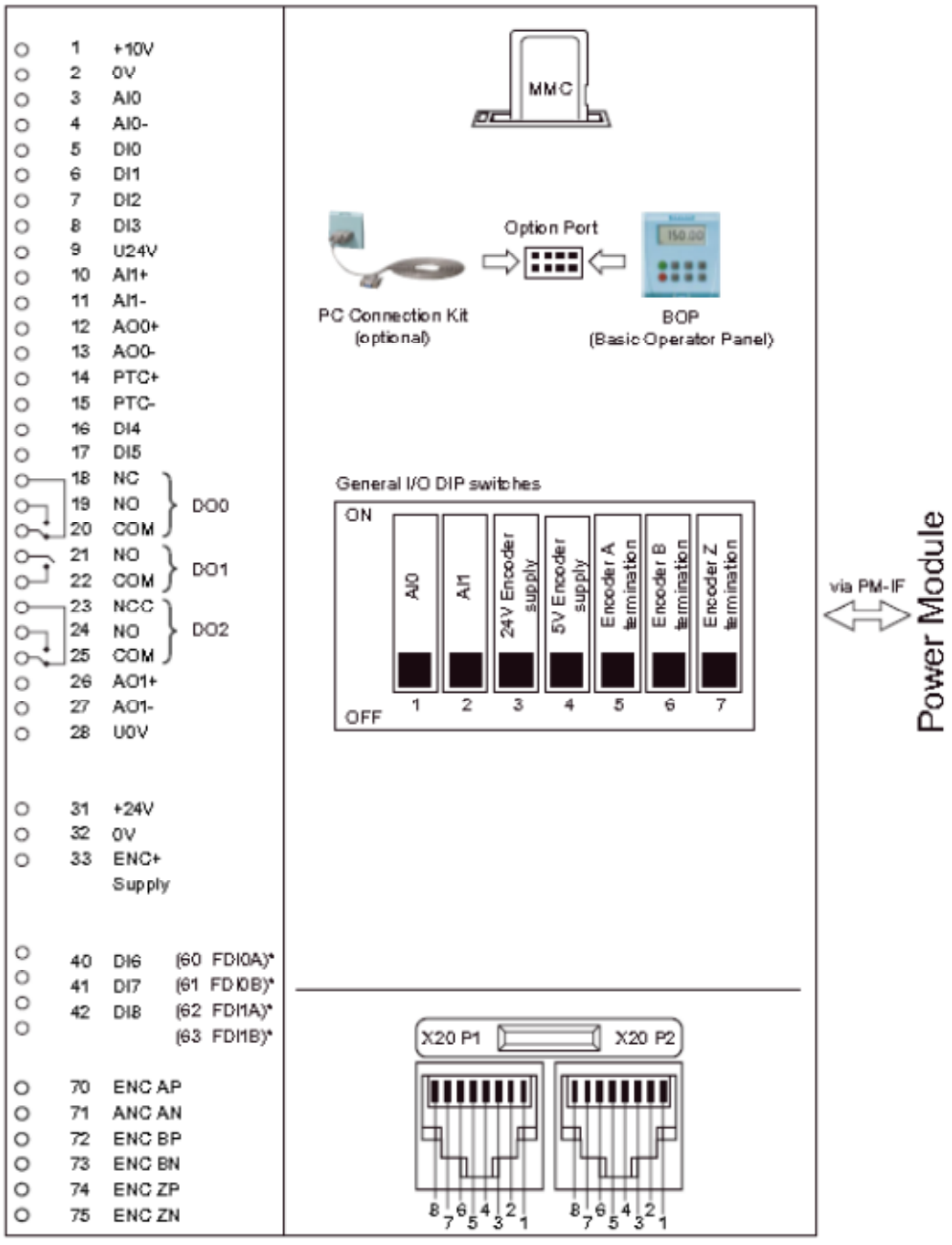


Obrázek 1 Převodní charakteristika analogového vstupu žádané hodnoty frekvence

Aktivace pomocného napájení otáčkového čidla

Po aktivaci bude na svorce 33 (ENC+ Supply) dostupné napětí +24 V / 300 mA. Toto napětí lze využít pro napájení externí řídící jednotky ovládání otáček ventilátoru.

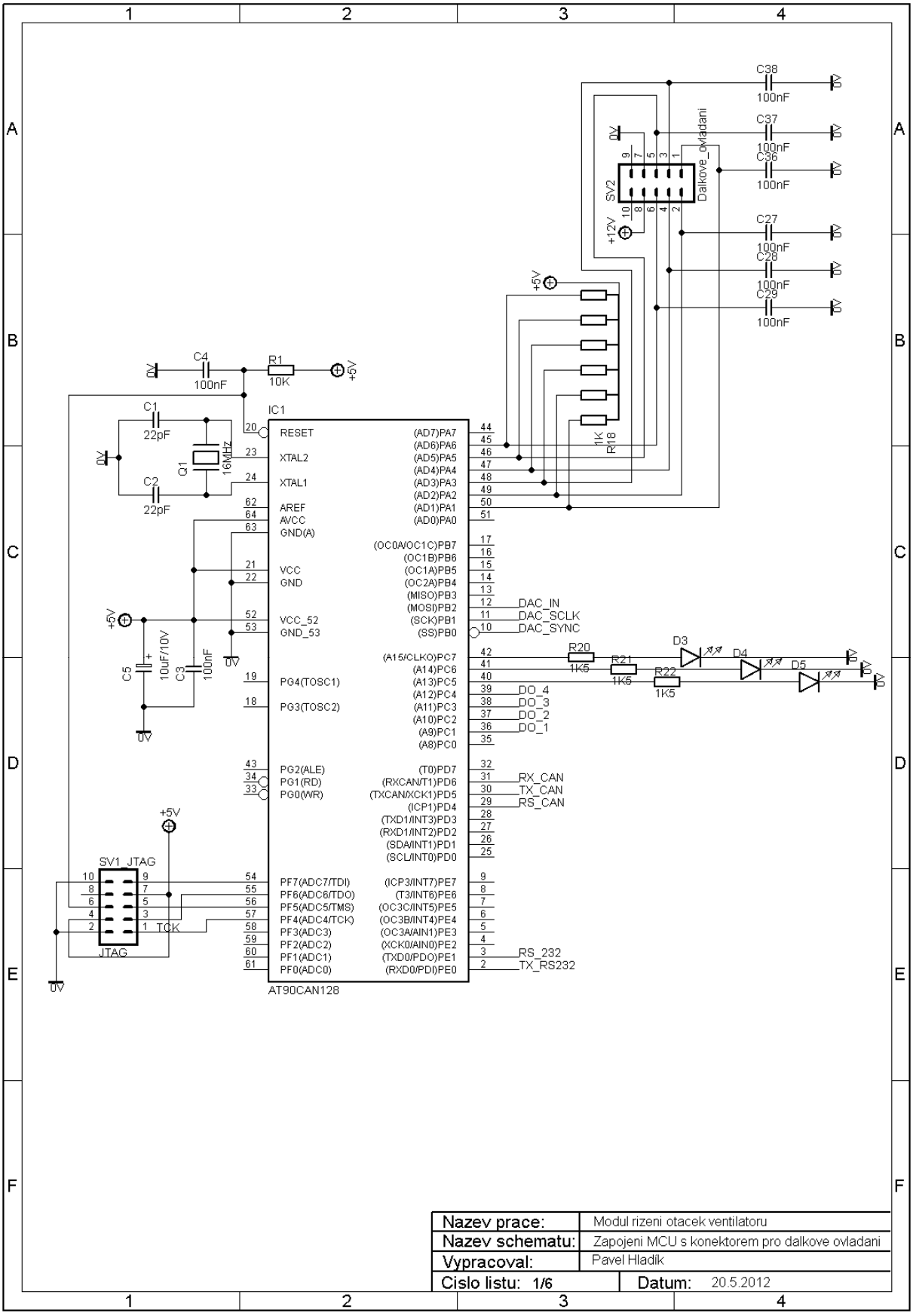
Postup: DIP switch č. 3 nastavit do polohy ON



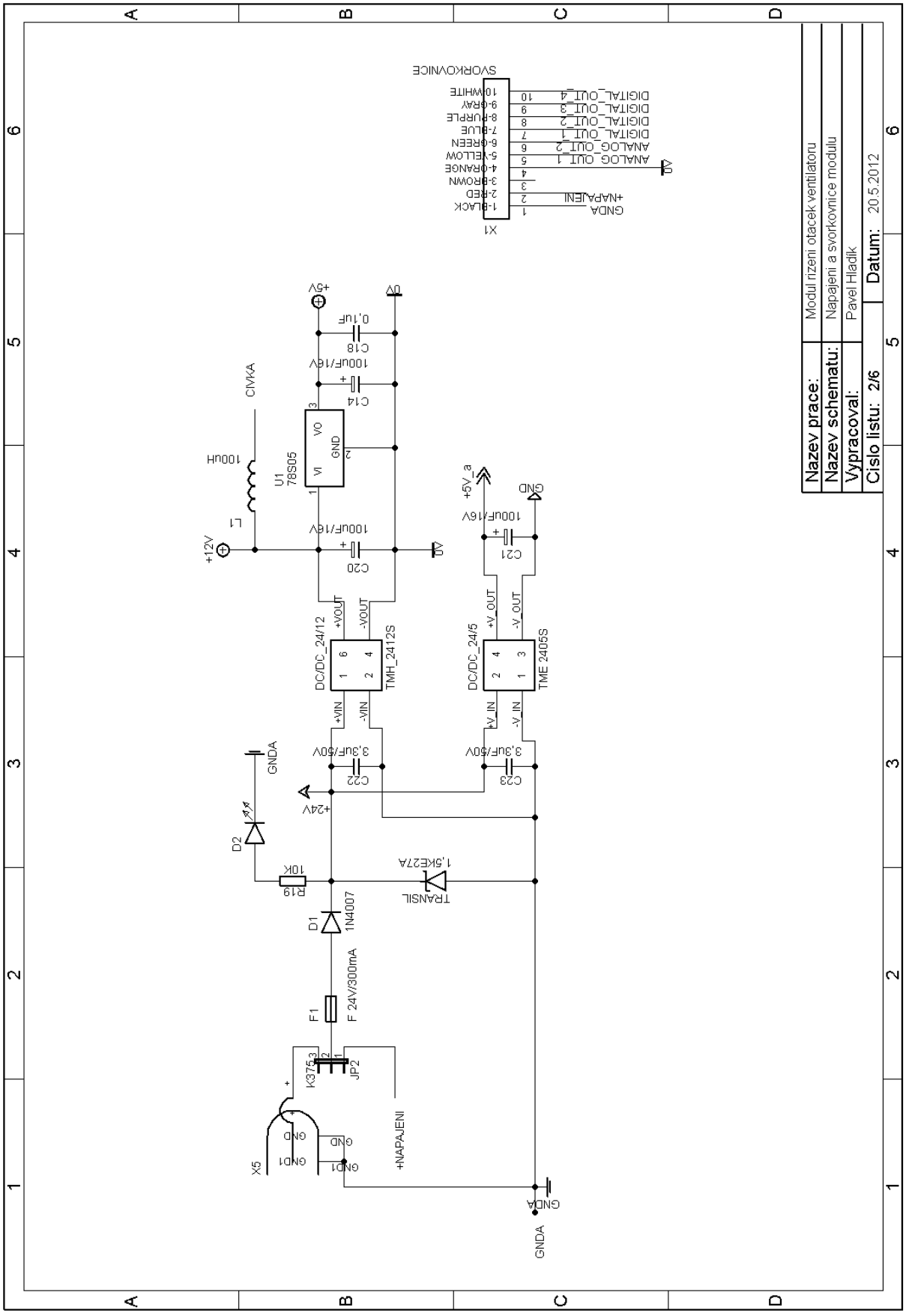
* Failsafe versions in brackets

Figure 3-4 Block diagram CU240S PN / CU240S PN-F

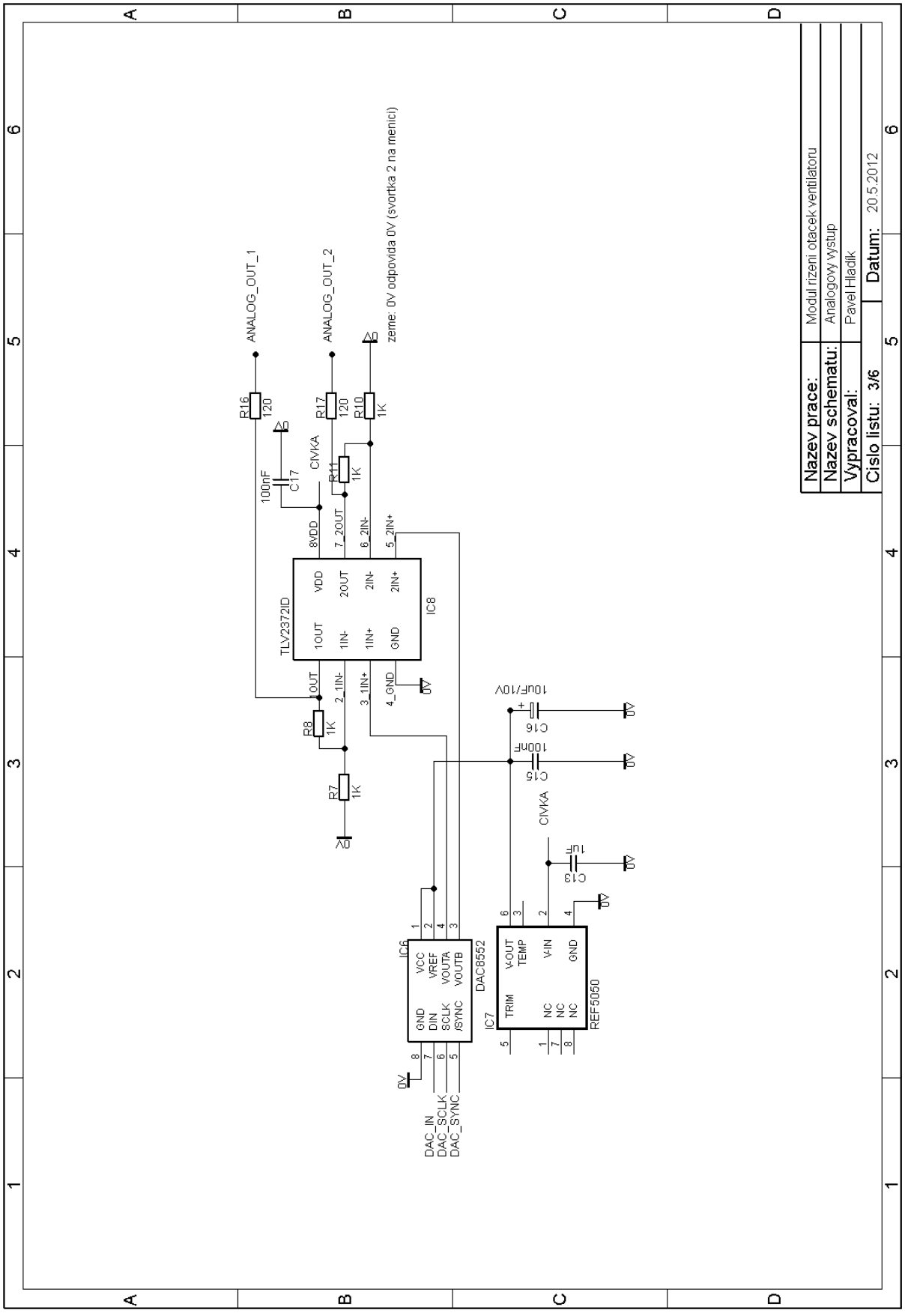
Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru



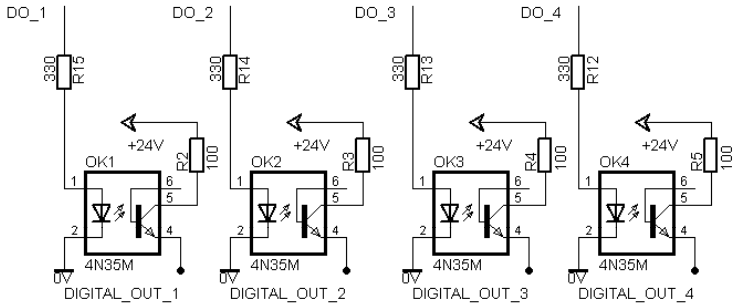
Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru



Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru

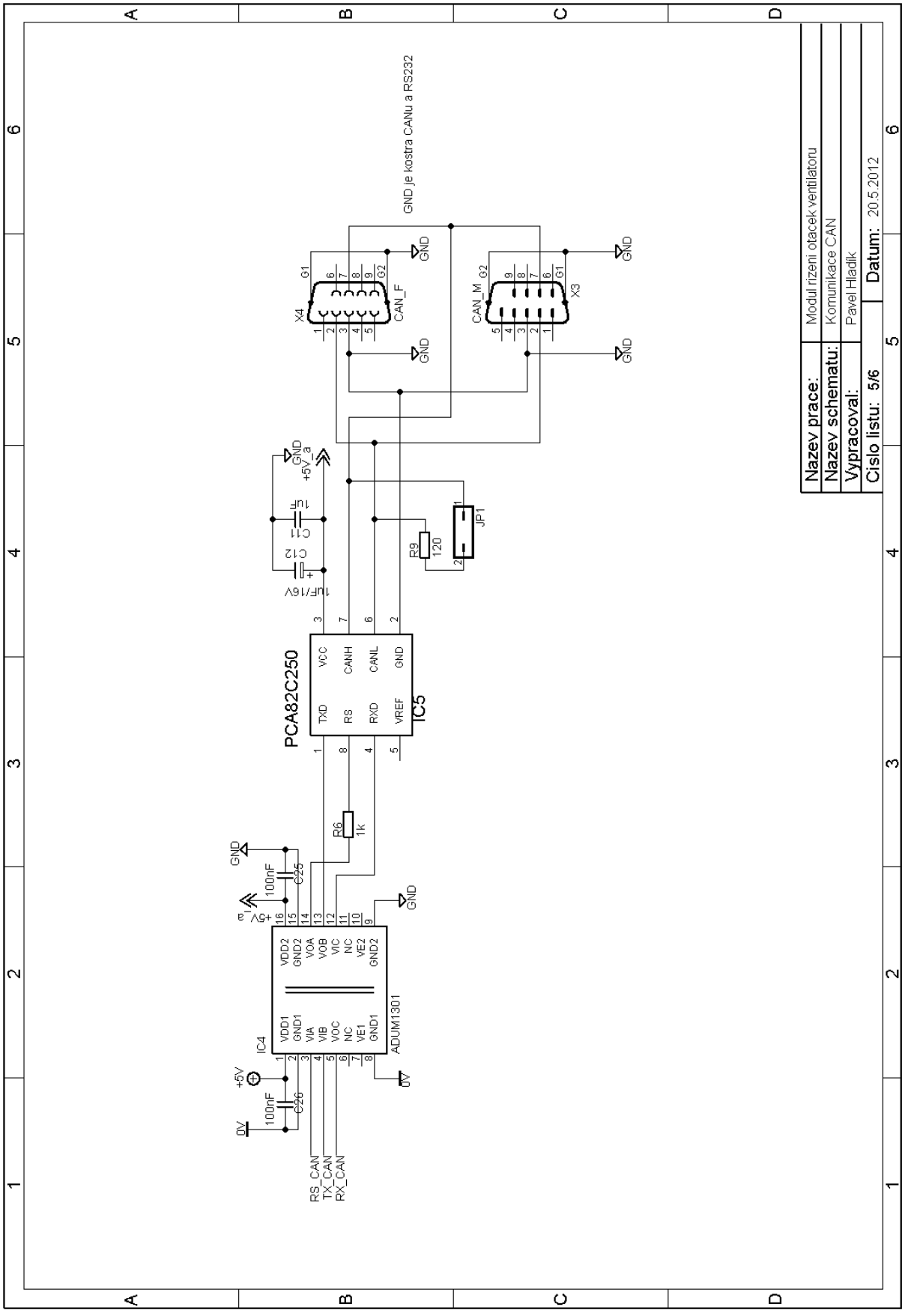


Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru

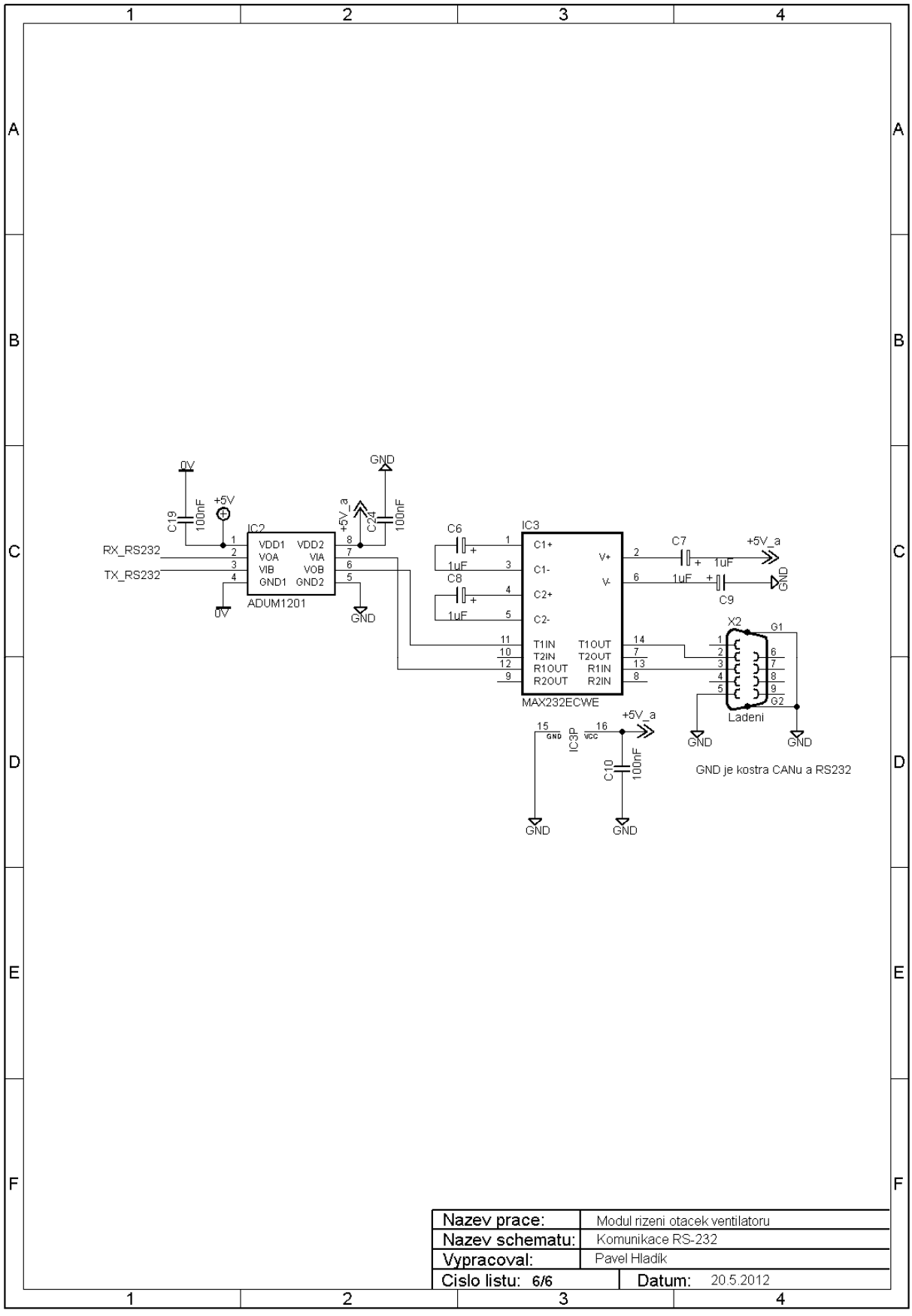


Nazev prace:	Modul rizeni otacek ventilatoru
Nazev schematu:	Digitalni vystup
Vypracoval:	Pavel Hladík
Cislo listu: 4/6	Datum: 20.5.2012

Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru

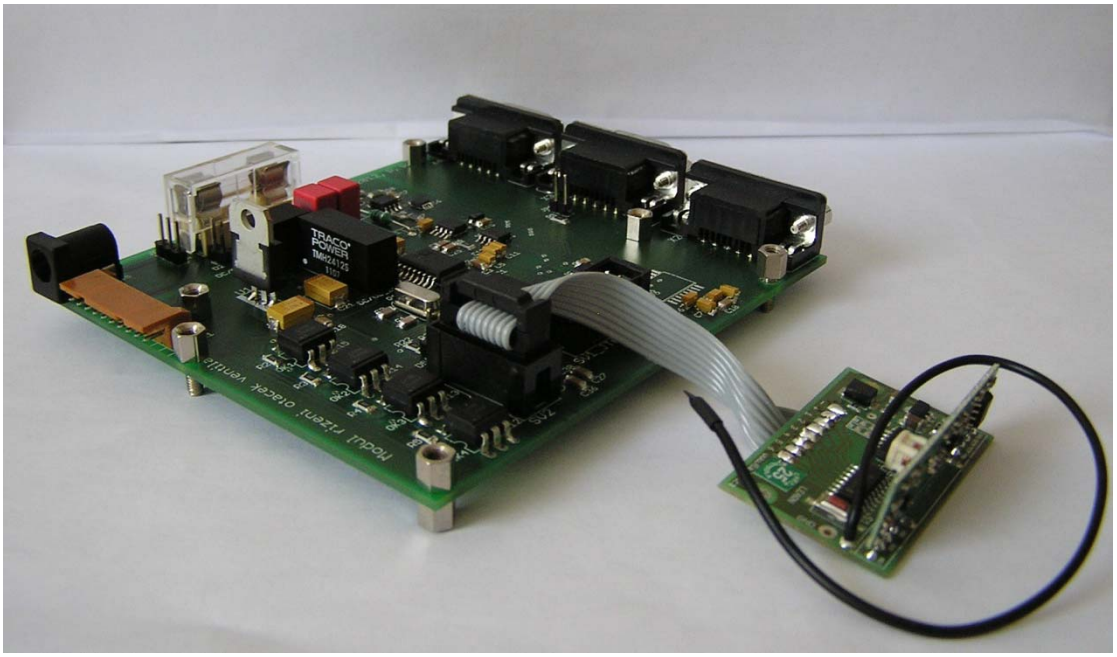


Příloha 2 – Schéma zapojení modulu řízení otáček ventilátoru



Příloha 3 – Fotografická dokumentace

DPS modulu s připojeným přijímačem dálkového ovládání



Osazení DPS do krabičky a připojení k frekvenčnímu měniči

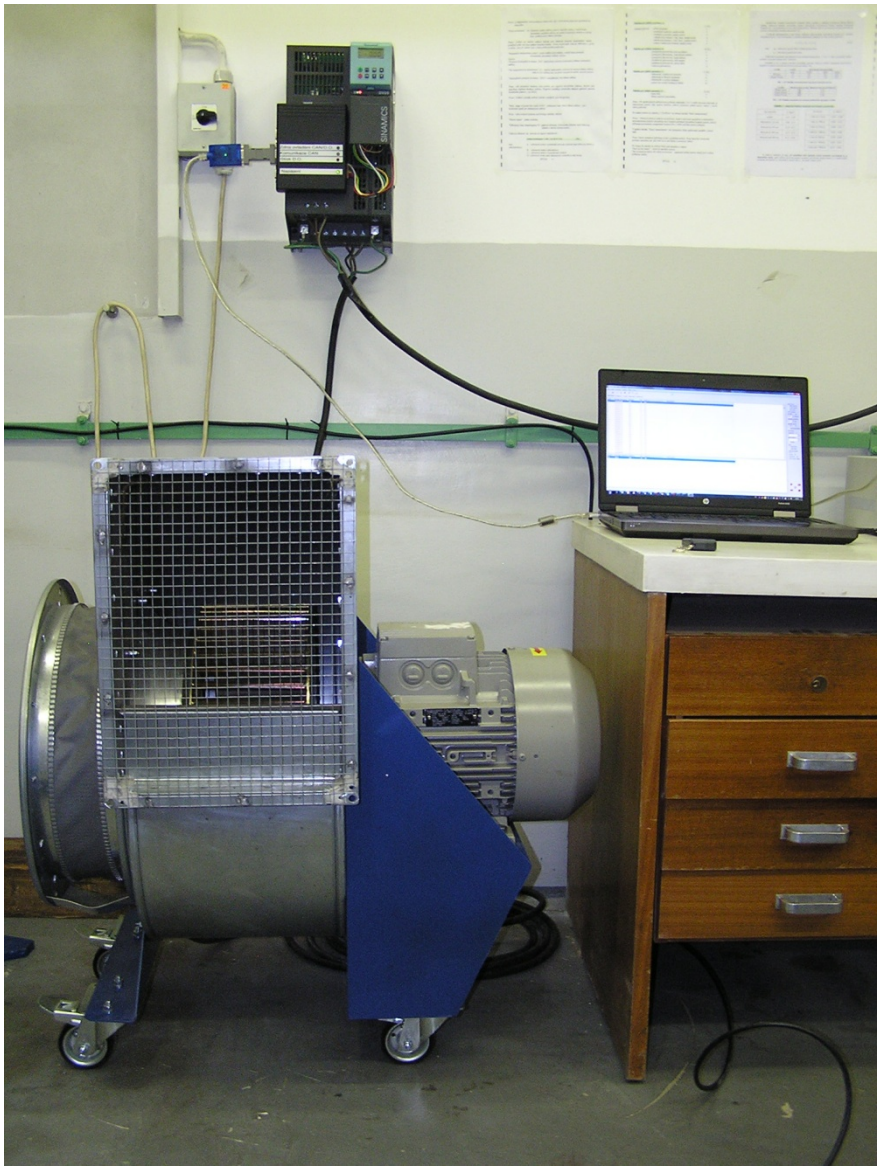
Příloha 3 – Fotografická dokumentace

Zkompletovaný modul řízení otáček ventilátoru

Umístění a připevnění modulu

Příloha 3 – Fotografická dokumentace

Měření na modulu řízení otáček ventilátoru



Dálkový ovladač modulu řízení otáček ventilátoru

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
Project : Modul_rizeni_otacek_ventilatoru
Version : 1.1
Date : 14.4.2012
Author : NeVaDa
Company : Hladík
Comments:
Chip type : AT90CAN128
Program type : Application
AVR Core Clock frequency: 16,000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 1024
*****************************************************/
#include <90can128.h>
#include <delay.h>
#include <stdio.h>
#include "CanAvr.h"
// SPI functions
#include <spi.h>
//definice vstupy/vystupy
//vstupy tlacitek
#define TLAC_PIN PINA
#define TLAC_PORT PORTA
#define TLAC_DDR DDRA
#define TLAC_ZAP 1 //2
#define TLAC_VYP 2 //4
#define TLAC_SMER 3 //8
#define TLAC_OVL 4 //16
#define TLAC_OTAC_N 5 //32
#define TLAC_OTAC_D 6 //64
// D/A prevodnik
#define DAC_PIN PINB
#define DAC_PORT PORTB
#define DAC_DDR DDRB
#define SS 0
#define SCK 1
#define MOSI 2
//digitalni vystupy + LED
#define DO_PIN PINC
#define DO_PORT PORTC
#define DO_DDR DDRC
#define DO1 1
#define DO2 2
#define DO3 3
#define DO4 4
#define LED_OVL 5 // indikuje zdroj zadane hodnoty (CAN nebo na ruku)
#define LED_CAN 6 // indikuje smer otaceni (levy neb pravy)
#define LED_SIGNAL 7 // indikuje stisk tlacitka na dalkovem ovladaci
//CAN
#define CAN_PIN PIND
#define CAN_PORT PORTD
#define CAN_DDR DDRD
#define CAN_RS 4 // Control data CAN
#define CAN_TX 5 // Write data CAN
#define CAN_RX 6 // Read data CAN
//RS232
#define RS232_PIN PINE
#define RS232_PORT PORTE
#define RS232_DDR DDRE
#define RS232_TX 0 // Read data RS232
#define RS232_RX 1 // Write data RS232
//definici kodu tlacitek
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
#define KOD_TLAC_ZAP (1<<TLAC_ZAP)
#define KOD_TLAC_VYP (1<<TLAC_VYP)
#define KOD_TLAC_SMER (1<<TLAC_SMER)
#define KOD_TLAC_OVL (1<<TLAC_OVL)
#define KOD_TLAC_OTAC_N (1<<TLAC_OTAC_N)
#define KOD_TLAC_OTAC_D (1<<TLAC_OTAC_D)
//makra pro ovladani digitalnich vystupu DO1 az DO4 a LED1 az LED3
#define MENIC_ZAP (DO_PORT |= (1<<DO1)) //strednik se doda pri volani makra
#define MENIC_VYP (DO_PORT &= ~(1<<DO1))
#define MENIC_SMER_L (DO_PORT |= (1<<DO2))
#define MENIC_SMER_R (DO_PORT &= ~(1<<DO2))
#define LED_OVL_ZAP (DO_PORT |= (1<<LED_OVL)) // Ovladani z CAN (LED sviti) nebo z
dalkoveho ovl. (LED nesviti)
#define LED_OVL_VYP (DO_PORT &= ~(1<<LED_OVL))
#define LED_CAN_RX_ZAP (DO_PORT |= (1<<LED_CAN)) // Signalizuje aktivitu na
sbernici
CAN
#define LED_CAN_RX_VYP (DO_PORT &= ~(1<<LED_CAN))
#define LED_SIGNAL_ZAP (DO_PORT |= (1<<LED_SIGNAL)) // Signalizuje stisk tlacitka
na
dalkovem ovl.
#define LED_SIGNAL_VYP (DO_PORT &= ~(1<<LED_SIGNAL))
// slave select pro D/A prevodnik
#define SS_LOW (DAC_PORT &= ~(1<<SS))
#define SS_HIGH (DAC_PORT |= (1<<SS))
#define OTACKY_MAX 100 // 100 % maximalni zadana hofdnota otacek
#define OTACKY_MIN 0 // 0 % minimalni zadana hodnota otacek
#define OKNO_ZAKMITY 10 //casove okno pro osetreni zakmitu tlacitek [ms]
#define RYCHLOST_ZMENY_OTACEK 5 //rychlost inkrementace/dekrementace zadane hodnoty
otacek
[%/s]
#define DELICKA_OTACKY (1000/RYCHLOST_ZMENY_OTACEK/OKNO_ZAKMITY) //delicka pro
inkementace/
dekrementace zadane hodnoty otacek
// definice CAN ID
#define CAN_MSG_RX1 0x10 // ID zpravy s rychlosti vozidla
// Deklarace globalnich promennych
typedef enum {LEVY, PRAVY} SMER_OTACENI;
typedef enum {CAN, RUCNI} ZDROJ_OVLADANI;
typedef enum {ZAPNUTO, VYPNUTO} STAV_RIZENI;
SMER_OTACENI smer = PRAVY; // smer otaceni
ZDROJ_OVLADANI ovl = RUCNI; // zdroj zadane hodnoty
STAV_RIZENI stav = VYPNUTO; // stav rizeni (vypinac, ktery spousti/zastavuje
ventilator)
unsigned char otacky = 0; // zadana hodnota otacek v procentech 0 - 100 %
unsigned char rychlost = 0; // rychlost vozidla [km/h]
unsigned char fRychlostPrijata = 0; // priznak uspesneho prijmu CAN zpravy s
rychlosti
#define VELIKOST_POLE 14
unsigned char TabOtacky [VELIKOST_POLE] = {0, 10, 20, 30, 40, 51, 60, 65, 70, 75,
79, 85, 89,
100}; // otacky ventilatoru v %, vztazna frekvence je 50Hz = 100% otacek
unsigned char TabRychlost [VELIKOST_POLE] = {0, 10, 15, 21, 27, 34, 44, 54, 68, 80,
87, 100,
107 ,123}; // Rychlost vozidla v km/h
// Deklarace lokalnich funkci
unsigned char CtiTlacitka (void);
unsigned int PrepocetProcentoCislo (unsigned char procento);
void WriteToDAC(unsigned char channel, unsigned int counts); // D/A prevodnik
void ProcessCanInterrupt(void); // CAN
unsigned char PrevedRychlostNaOtacky(unsigned char rychlost); // Prevede rychlost
vozidla na
pozadovane otacky ventilarotu
void main(void)
{
// Deklarace lokalnich promenych
unsigned char tlac = 0; // kod stisknuteho tlacitka
unsigned char tlac_minule = 0; // kod minuleho zmacknuti tlacitka
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
unsigned int cnts; // zadana hodnota pro D/A prevodnik
unsigned char k = 0; // pocitadlo hlavni smycky
tCanMsg canMsg; // CAN
unsigned char canData[8]; // CAN
unsigned char mobNo; // CAN
// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif
// Input/Output Ports initialization
// tlacitka
TLAC_DDR = 0x00; // cely port vstupni
TLAC_PORT = 0xFF; // aktivace pull-upu
// D/A prevodnik
DAC_DDR |= (1<<MOSI) | (1<<SS) | (1<<SCK);
DAC_PORT |= (1<<SS);
// digitalni vystupy + LED
DO_DDR = 0xFF; // cely port vystupni
DO_PORT = 0x00; // vystupy neaktivni
// CAN
CAN_DDR |= (1<<CAN_TX) | (1<<CAN_RS);
CAN_PORT = 0x00;
// RS-232
RS232_DDR |= (1<<RS232_TX);
RS232_PORT = 0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0A=0x00;
TCNT0=0x00;
OCR0A=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
TCCR2A=0x00;
TCNT2=0x00;
OCR2A=0x00;
// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer3 Stopped
// Mode: Normal top=0xFFFF
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x00;
// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=0x00;
// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=0x00;
// Timer/Counter 3 Interrupt(s) initialization
TIMSK3=0x00;
// USART0 initialization //nastaveni komunickace po RS232
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;
// USART1 initialization
// USART1 disabled
UCSR1B=0x00;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
ADCSRB=0x00;
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
DIDR1=0x00;
// ADC initialization
// ADC disabled
ADCSRA=0x00;
// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 4000,000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x54;
SPSR=0x00;
// TWI initialization
// TWI disabled
TWCR=0x00;
// CAN Controller initialization, 1000 kbps
CanInit(AVR_CLOCK_16_MHZ,CAN_SPEED_1000_KBPS);
// nastaveni prijmu zpravy po CANu
canMsg = MakeCanMsg(RX_MODE,CAN_MSG_RX1,0x7FF,0,1,canData); // prijem, ID =
CAN_MSG_RX1, CAN2.
0A, 1 bajt dat
mobNo = CanSetRx(canMsg);
printf("Nastaven MOB%d RX pro prijem zpravy \tID=0x%X\n", mobNo, CAN_MSG_RX1);
delay_ms(1000);
// ovladani LED
LED_SIGNAL_VYP;
if (ovl == CAN) LED_OVL_ZAP;
else LED_OVL_VYP;
LED_CAN_RX_VYP;
// Prijem zpravy z rychlosti z CANu
while (1)
{
k++;
tlac = CtiTlacitka(); // zpozdeni min. 10 ms
// ZMENA OTACEK POMOCI DALKOVEHO OVLADANI POUZE V MODU RUCNIM
if (ovl == RUCNI && (!(k%10))) // kadych 100 ms reakce na zmenu zadane hodnoty
tlacitek, urcuje rychlost pricitani/odecitani pristustku pro D/A prevodnik
{
switch (tlac)
{
// reakce na stistknute a drzene tlacitko - men zadanou hodnotu otacek
case KOD_TLAC_OTAC_N: //if (tlac == tlac_minule)
{
if (otacky < OTACKY_MAX) otacky++; //
inkrementace otacek po 1 %
}
break;
case KOD_TLAC_OTAC_D: //if (tlac == tlac_minule)
{
if (otacky > OTACKY_MIN) otacky--; //
dekrementace otacek po 1 %
}
break;
}
}
if (tlac != tlac_minule) //porovnani s minulym stavem tlacitka, reaguje pouze na
zmenu
{
switch (tlac)
{
// zapne menic
case KOD_TLAC_ZAP: stav = ZAPNUTO;
break;
// vypne menic
case KOD_TLAC_VYP: stav = VYPNUTO;
break;
// voli zdroj zadane hodnoty (CAN/dalkove ovl.)
case KOD_TLAC_OVL: if (ovl == CAN)
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
{
ovl = RUCNI;
LED_OVL_VYP;
}
else if (ovl == RUCNI)
{
ovl = CAN;
LED_OVL_ZAP;
}
break;
case KOD_TLAC_SMER:
if (smer == LEVY)
{
smer = PRAVY;
}
else if (smer == PRAVY)
{
smer = LEVY;
}
break;
}
}
tlac_minule = tlac; //ulozeni aktualniho stavu tlacitka
// signalizace stisknuteho tlacitka
if (tlac) LED_SIGNAL_ZAP;
else LED_SIGNAL_VYP;
// aktivuj optoclen dle navoleneho smeru otaceni
if (smer == LEVY) MENIC_SMER_L;
else MENIC_SMER_R;
if (stav == ZAPNUTO) MENIC_ZAP;
else MENIC_VYP;
// reakce na preruseni od radice CAN
if (fCanInt)
{
mCANIntOff; // vypnuti preruseni od radice CAN na dobu nezbytne nutnou pro
zpracovani udalosti
ProcessCanInterrupt(); // volani funkce pro zpracovani preruseni
mCANIntOn; // znovu povoleni preruseni od radice CAN
}
// ovladani z CANu
if (ovl == CAN)
{
otacky = PrevedRychlostNaOtacky(rychlost); // prevede rychlost na otacky [%]
}
cnts = PrepocetProcentoCislo(otacky);
WriteToDAC(0,cnts); // nastavi napeti D/A prevodniku kanal 0
// pokud byla prijata CAN zprava s rychlosti, tak za 100 ms zhasni LED_CAN_RX
if (fRychlostPrijata == 1 && (!(k%10)))
{
fRychlostPrijata = 0;
LED_CAN_RX_VYP; // zhasni LED
}
// kazdou sekundu vypis stavu
if (!(k%100))
{
printf("Stav rizeni: ");
if (stav == ZAPNUTO) printf("ZAP\n");
else printf("VYP\n");
printf("Smer otaceni: ");
if (smer == LEVY) printf("vlevo\n");
else printf("vpravo\n");
printf("Zdroj zadaneho hodnoty otacek: ");
if (ovl == CAN) printf("CAN\n");
else printf("dalkove ovladani\n");
printf("Rychlost = %d [km/h]\t Otacky = %d [%%]\n\n", rychlost, otacky);
}
}
}
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
// Cte stav tlacitek na dalkovem ovladaci
// Parametry: zadne
// Vraci: unsigned char - kod stisknuteho tlacitka, popripade kod odpovidajici vice
stisknutym
tlacitkum
// - stisknutemu tlacitku odpovida logicka 1
unsigned char CtiTlacitka (void)
{
//osetreni zakmitu
unsigned char pom;
do
{
pom = TLAC_PIN; //prvni cteni
delay_ms(OKNO_ZAKMITY); //pauza na preskoceni zakmitu
} while (pom != TLAC_PIN); //druhe cteni a porovnani s prvnim, pokud jsou rozdilna
tak
cti znovu stav tlacitek
return ~pom; //v pom je ulozen kod klavesy, poripade kod stisku vice klaves
}
//Cte pozadovanou hodnotu v % a prepocitava na pozadovanou hodnotu D/A prevodniku
//Parametry: unsigned char procento - pozadovana hodnota otacek v %
//Vraci: unsigned int - Prepoctene cislo 0 - 65535
unsigned int PrepocetProcentoCislo (unsigned char procento)
{
unsigned int vypocet;
vypocet = (unsigned int) (((unsigned long) procento * 65535) / 100);
return vypocet;
}
// Zapise novou hodnotu do DAC
// Vstupy: channel - cislo kanalu (0 nebo 1)
// counts - hodnota pro DAC v rozsahu 0 - 65535
void WriteToDAC(unsigned char channel, unsigned int counts)
{
unsigned char command = 0;
#asm("cli")
if (channel == 0)
{
command = (1<<4);
SS_LOW;
spi(command);
spi(counts>>8); // nejrpve MSB
spi(counts&0xFF); // pak LSB
SS_HIGH;
}
else if (channel == 1)
{
command = (1<<5) | (1<<2);
SS_LOW;
spi(command);
spi(counts>>8);
spi(counts&0xFF);
SS_HIGH;
}
#asm("sei")
}
// Funkce pro obsluhu udalosti od radice CAN
void ProcessCanInterrupt(void)
{
unsigned char i;
unsigned char canData[8]; // pomocny buffer pro datove bajty CAN zpravy (jak
prijem, tak
odeslani)
unsigned char rxLengthOk; // shoduje se skutecny pocet datovych bajtu prijate
zpravy s
ocekavanym poctem ? 1 - ano, 0 - ne
tCanMsg canMsg;
unsigned char mobNo;
fCanInt = 0; // nulovani priznaku preruseni od radice CAN
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
for(i = 0; i < 15; i++) // zjisteni, ktery MOB vygeneroval preruseni
{
switch (canMobInt[i]) // kdyz MOB vygeneroval preruseni, bude na prislusne pozici v
poli canMobInt nastaven priznak, jinak 0
{
// zjisteni jaka akce vygenerovala preruseni od daneho MOBu
case TXOK_INT: // zprava byla odeslana
break;
case RXOK_INT: // zprava byla prijata
rxLengthOk = CanRx(i,&canMsg); // precte prijatou zpravu z daneho MOBu
switch (canMsg.id) // reakce na prijatou zpravu
{
case CAN_MSG_RX1: // prijem zpravy s rychlosti vozidla
if (rxLengthOk) // ukladej rychlost pouze pokud prisel pozadovany
pocet datovych bajtu
{
printf("MOB%d RX OK\tID=%d, DLC=%d\n", i, canMsg.id, canMsg.dlc);
// info o prijmu zpravy na RS-232
rychlost = canMsg.data[0]; // ulozeni rychlosti ze zpravy do
globalni promenne
fRychlostPrijata = 1;
LED_CAN_RX_ZAP; // rozsvit LED
}
else printf("MOB%d RX OK\tID=%d, DLC=%d\tDLC se neshoduje!\n", i,
canMsg.id, canMsg.dlc);
// znovu nastavime MOB pro prijem
canMsg = MakeCanMsg(RX_MODE,CAN_MSG_RX1,0x7FF,0,1,canData);
mobNo = CanSetRx(canMsg);
break;
}
break;
// preruseni bylo vyvolanou chybou na urovni prislusneho MOBu
case BERR_INT: // chyba bitu (BIT ERROR)
break;
case SERR_INT: // chyba vkladani bitu (STUFF ERROR)
break;
case CERR_INT: // chyba CRC (CRC ERROR)
break;
case FERR_INT: // chyba ramce (FRAME ERROR)
break;
case AERR_INT: // chyba potvrzeni prijmu (ACK ERROR)
break;
default:
break;
}
}
// preruseni bylo vyvolanou chybou na "general" urovni
// nastava jen pri prijmu a to jen v pripade, kdyz se neshoduje ID prijate zpravy s
ocekavanym
// v ostatnich pripadech (TX, RX a ID se shoduji) error na urovni MOBu
switch (canGenInt) // CAN general interrupt
{
case SERG_INT:
break;
case CERG_INT:
break;
case FERG_INT:
break;
case AERG_INT:
break;
case BOFFI_INT: // prechod do BUS-OFF
break;
default:
break;
}
}
unsigned char PrevedRychlostNaOtacky(unsigned char rychlost)
{
```

Příloha 4 – Hlavní část zdrojového kódu modulu řízení otáček ventilátoru

```
unsigned char i,j; // body v hledanem intervalu i=hornni j=spodni
// Pokud je hodnota rychlosti vetsi jak maximum rychlosti tak vraci nejvetsi
hodnotu
otacek
if (rychlost > TabRychlost [VELIKOST_POLE - 1])
{
return TabOtacky[VELIKOST_POLE - 1];
}
for (i=0; i<VELIKOST_POLE; i++)
{
if (TabRychlost[i]>rychlost)
break;
}
j = i-1;
//otacky = (((float)(TabOtacky [i] - TabOtacky [j]) / (TabRychlost [i] -
TabRychlost [j]))
* (rychlost - TabRychlost [j]) + TabOtacky [j]);
otacky = (unsigned char)( ((unsigned int)(TabOtacky [i] - TabOtacky [j])*(rychlost
-
TabRychlost [j])) / (TabRychlost [i] - TabRychlost [j]) + TabOtacky [j] );
return otacky;
}
```