UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Řídící jednotka pneumatického systému

Jaroslav Firbas

Bakalářská práce

2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jaroslav Firbas**
Osobní číslo:	**I10436**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Řídicí jednotka pneumatického systému.**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cíl:
Realizovat elektronickou část pneumatického systému se dvěma ventilátory. Elektronická část bude zajišťovat ovládání příkonu ventilátoru, předzpracování signálů čidel a signálů ovládacího panelu s grafickým zobrazovačem pro připojení k mikrokontroleru ATmega128. Vytvořit základní programové vybavení v jazyce "C" pro vyhodnocení otáček obou ventilátorů, ovládání příkonu jednoho ventilátoru, zajištění obsluhy ovládacího panelu a řízení otáček ovládaného ventilátoru.
Teoretická část:
a) přehled měření a generování napěťového signálu na mikrořadičích
b) diskrétní PID regulátor a jeho SW realizace na mikrořadičích
Praktická část:
a) realizovat ovládání ventilátoru spínáním napájecího napětí signálem PWM
b) realizovat řídicí jednotku (ŘJ) s mikrořadičem Atmel AVR ATmega128
- ŘJ bude umožňovat připojení všech signálů ovládacího panelu pneumatického systému včetně signálů pro připojení LCD modulu, generování napětí bude pomocí D/A převodníku připojeného přes sběrnici SPI š
- ŘJ bude umožňovat další komunikaci přes sběrnici RS232
c) vytvořit základní programové vybavení (SW) se základními funkcemi
- vyhodnocení signálů z měření otáček ventilátorů, generování řídicího signálu PWM pro ovládání jednoho ventilátoru, měření a generování jednoho napětí v rozsazích 0-5 V
- zajištění zobrazování, přepínání a změnu vybraných informací na ovládacím panelu (LCD modul, LED)
- realizaci jednoduchého diskrétního PI regulátoru včetně zajištění přepínání režimů podle přepínačů na ovládacím panelu

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

BALÁTĚ, Jaroslav. Automatické řízení. 2., přeprac. vyd. Praha: BEN, 2004, 663 s. ISBN 80-730-0148-9.
VÍTEČKOVÁ, Miluše a Antonín VÍTEČEK. Základy automatické regulace. 1. vyd. Ostrava: VŠB - Technická univerzita, 2006, 198 s. ISBN 80-248-1068-9.
On line dokumentace dostupná na Atmel Corporation [online]. 2012 [cit. 2012-10-29]. Dostupné z: http://www.atmel.com/

Vedoucí bakalářské práce: **doc. Ing. František Dušek, CSc.**
Katedra řízení procesů

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice -8- Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji:

Tuto práci jsem 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 vyplívající 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 25. 3. 2013

Jaroslav Firbas

Poděkování

Rád bych tímto poděkoval své rodině za podporu při studiu na Univerzitě Pardubice. Dále bych chtěl poděkovat vedoucímu své práce panu doc. Ing. Františku Duškovi, CSc. a panu Ing. Liboru Havlíčkovi, Ph.D. za odborný dohled a konzultace v průběhu vypracování.

Anotace

Práce se věnuje návrhu a realizaci řídící jednotky pneumatického systému. Pneumatický systém je tvořen větrným tunelem s dvěma ventilátory, první ventilátor vhání vzduch do větrného tunelu, čímž roztáčí druhý ventilátor. Řídící jednotka je doplněna o měřící systém, který měří rychlost otáčení obou ventilátorů. Řídící jednotka dále zajišťuje ovládání příkonu jednoho ventilátoru pomocí PWM signálu, komunikaci s ovládacím panelem včetně LCD displeje, generování a měření signálů pro připojení externího regulátoru a umožňuje komunikaci prostřednictvím USB.

Klíčová slova

Pneumatický systém, ventilátor, měření, regulace

Title

Control unit of pneumatic system

Annotation

The work is dedicated to the design and implementation of the pneumatic system control unit. The pneumatic system consists of wind tunnel with two fans. The first fan blows air into the wind tunnel, and the air spins the second fan. The control unit is complemented by a measurement system that measures the rotation speed of both fans. The control unit also provides PWM signal to control the first fan power, communication with the control panel including LCD displays, generating a measurement signal for an external controller and enables communication via USB.

Key words

Pneumatic system, fan speed measurement, microcontroller ATmega128

Obsah

Seznam zkratek

A/D	Převodník z analogového na digitální signál
ADC	Analog-to-digital converter
D/A	Převodník z digitálního na analogový signál
DAC	Digital to Analog converter
EEPROM	Electrically Erasable Programmable Read-only Memory
SRAM	Static Random Access Memory
ROM	Read-only memory
JTAG	Joint Test Action Group
SPI	Serial Peripheral Interface
RISC	Reduced instruction set computing
USB	Universal Serial Bus
PWM	Pulse Width Modulation
LCD	Liquid Crystal Display
LED	Light Emited Diode
LSB	Least significant bit
MSB	Most significant bit
TTL	Tranzistor-Tranzistor Logic
GND	Společná zem

Seznam obrázků

Seznam tabulek

1 Úvod

K výuce odborných předmětů v oblasti automatizace a řízení procesů neodmyslitelně patří i laboratorní experimenty. Takovéto laboratorní experimenty probíhají při laboratorních cvičeních, kdy má student za úkol pracovat s laboratorním modelem. Laboratorní model bývá doplněn (či nahrazen) matematickým modelem představující charakteristické vlastnosti reálného modelu. Zařízení, která by představovala reálný model, v laboratořích moc nebývá. Je to především proto, že takovéto zařízení představující reálný model bývá cenově nákladné. Další možností jak získat reálný model, je takový model navrhnout a vyrobit. A právě návrhem a konstrukcí takového modelu se zabývá tato práce.

Cílem této bakalářské práce je navrhnout a realizovat řídící jednotku pneumatického systému, tvořeného větrným tunelem, dvěma ventilátory a řídící jednotkou. Do větrného tunelu bude vhánět primární (větší) ventilátor vzduch a tím bude roztáčet sekundární (menší) ventilátor. Primární ventilátor bude možné ovládat pomocí spínání napájecího napětí PWM signálem, to znamená, že bude možné regulovat jeho otáčky. Otáčky obou ventilátorů budou měřeny za pomocí optických čidel, které tvoří LED dioda a fototranzistor. Signál z těchto čidel pak bude upraven na TTL logiku a přiveden k mikrokontroléru ATmega128, na kterém dojde k vyhodnocení signálu. K zobrazení změřených dat bude využit 2×16 znakový LCD displej, který bude umístěn na ovládacím panelu zařízení. Na ovládacím panelu se ještě budou nacházet tři tlačítka a tři přepínače, které budou sloužit k nastavování různých funkcí modelu. Kromě tlačítek a přepínačů budou na ovládacím panelu ještě tři kontrolní LED diody. LCD displej bude s mikrokontrolérem ATmega128 komunikovat přes SPI sběrnici. SPI sběrnice bude využita i pro komunikaci s D/A převodníkem, který bude vytvářet analogové napětí odpovídající naměřeným otáčkám ventilátoru, tudíž bude vytvářet unifikovaný napěťový signál pro externí regulátor. K modelu bude možné připojit externí regulátor využívající standardní unifikované napěťové signály. Řídící jednotka bude moci digitálně komunikovat přes rozhraní USB.

2 Teoretická část

V této části bude popsána vybraná problematika spojená s návrhem a realizací řídící jednotky pneumatického systému.

2.1 Mikrokontrolér ATmega128

Tento mikrokontroler je od firmy Atmel a patří do řady mikrořadičů AVR, která je založena na pokročilé architektuře RISC. Tato architektura obsahuje 133 instrukcí, většina těchto instrukcí se vykoná během jednoho cyklu. ATmega128 je 8bitový mikrokontroler, který obsahuje 32 obecných registrů, které umožňují paralelní přístup. Díky relativně nízké pracovní frekvenci zůstává odběr proudu velmi malý. Modely AVR používají programovou paměť typu flash a datovou paměť typu EEPROM a SRAM. Atmega128 obsahuje mimo jiné i JTAG rozhraní, které se používá pro programování pamětí typu flash a umožňuje ladění programu přímo za chodu mikrokontroléru.

Obr. 1 – ATmega128 [1]

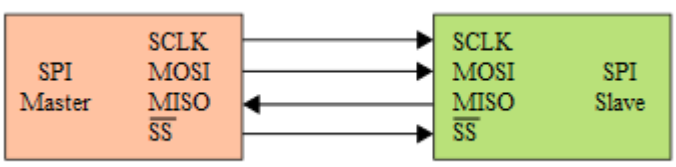
Základní vlastnosti ATmega128

- 8-mi bitový mikrokontrolér
- Pokročilá RISC architektura
- 32 8-mi bitových univerzálních pracovních registrů

- Až 16 MIPS při 16 MHz (až 16 miliónů instrukcí za sekundu)
- **Nonvolatilní programovou a datovou paměť**
  - 128 kB flash paměť programu
  - 4 kB EEPROM paměti
  - 4 kB interní SRAM paměti
- SPI rozhraní
- JTAG rozhraní
- **Periferie**
  - Dva 8-mi bitové čítače/časovače
  - Dva 16-ti bitové čítače/časovače
  - Dva 8-mi bitové PWM kanály
  - 6 PWM kanálů s programovatelným rozlišením 2 až 16 bitů
  - 8-mi kanálový, 10-ti bitový analogově-digitální převodník
  - Duální sériové rozhraní UART
  - Master / Slave sériové SPI rozhraní
  - On-chip analogový komparátor
- **Speciální funkce mikrokontroléru**
  - Interní RC oscilátor
  - Vnější a vnitřní zdroje přerušení
  - 6 úsporných režimů
- 53 programovatelných vstupních/výstupních pinů

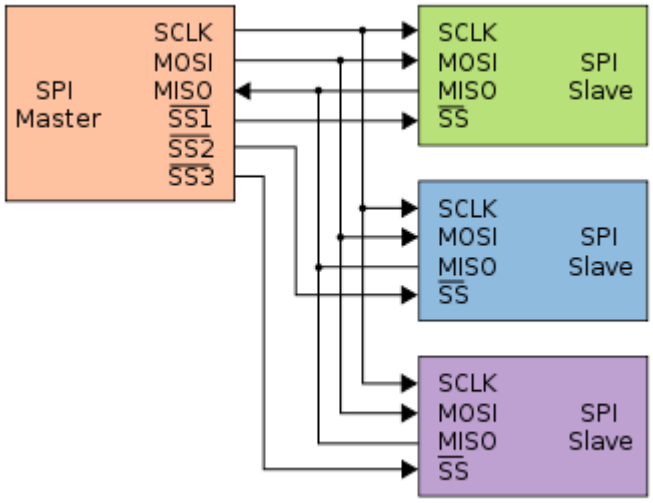
2.2 SPI (Serial Peripheral Interface)

SPI je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícím mikroprocesorem a ostatními integrovanými obvody (EEPROM, A/D převodníky, displeje…). Komunikace je realizována pomocí společné sběrnice. Adresace se provádí pomocí zvláštních vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení (piny SS nebo CS).



Obr. 2 – Sběrnice SPI [2]

Zařízení na SPI sběrnici jsou rozdělena na Master a Slave, kde zařízení Master je řídící a zařízení Slave je podřízené. Master řídí komunikaci pomocí hodinového signálu (SCLK) a určuje, se kterým zařízení se bude komunikovat pomocí signálu SS. Slave vysílá podle hodinového signálu, pokud je aktivován signálem SS.



Obr. 3 – Sběrnice SPI (pro více zařízení) [2]

Komunikace probíhá tak, že řídící zařízení (Master) nastaví logickou „0“ na SS zařízení, se kterým chce komunikovat (Slave). Poté se začne generovat hodinový signál na SCLK a v té chvíli vyšlou obě zařízení svoje data, přičemž MOSI (Master Out, Slave In) je vždy Master výstup a Slave vstup, MISO (Master In, Slave Out) je Master vstup, Slave výstup. Pro ukončení komunikace přestane Master generovat hodinový signál a nastaví SS na logickou “1”. Při použití sběrnice SPI pro více zařízení, je nutné, aby Master měl pro každé zařízení vlastní SS signál, jak je vidět na obrázku 2.2b. Ostatní signály (MOSI, MISO, SCLK) jsou pro všechna zařízení společná [2].

2.3 Komunikace s PC

V této kapitole bude teoreticky popsáno základní komunikační rozhraní pro sériovou komunikaci s PC. Konkrétně půjde o standard RS-232 a USB.

2.3.1 RS-232

Standard RS-232 se nejčastěji používá jako komunikační rozhraní osobních počítačů a další elektroniky. RS-232 umožňuje propojení a vzájemnou sériovou komunikaci dvou zařízení. To znamená, že jednotlivé bity přenášených dat jsou vysílány postupně za sebou (v sérii) po jednom páru vodičů v každém směru.

Standard definuje asynchronní sériovou komunikaci pro přenos dat. Pořadí přenosu datových bitů je od nejméně významného bitu (LSB) po bit nejvýznamnější (MSB). Počet datových bitů je volitelný, obvykle se používá 8 bitů, lze se také setkat se 7 nebo 9 bity. Logický stav „0“/„1“ přenášených dat je reprezentován pomocí dvou možných úrovní napětí, které jsou bipolární v rozsahu ±(3 - 25) V. Obvykle se, podle zařízení, používají úrovně ±5 V, ±10 V, ±12 V nebo ±15 V. Nejčastěji se používá varianta při které logické hodnotě „1“ odpovídá napětí -12 V a logické hodnotě „0“ pak +12 V. Základní tři vodiče rozhraní (příjem RxD, vysílání TxD a společná zem GND) jsou doplněny ještě dalšími vodiči sloužícími k řízení přenosu (vstupy DCD, DSR, CTS, RI, výstupy DTR, RTS). Ty mohou a nemusí být používány (zapojeny), nebo mohou být použity pro napájení elektronických obvodů v zařízení, jako je například počítačová myš [3].

Tab. 1 – Napěťové úrovně RS-232

Úroveň	Vysílač	Přijímač
log „0“	+5 V až +15 V	+3 V až +25 V
log „1“	- 5V až -15 V	-3 V až -25 V
Nedefinovaný	-3V až +3 V	

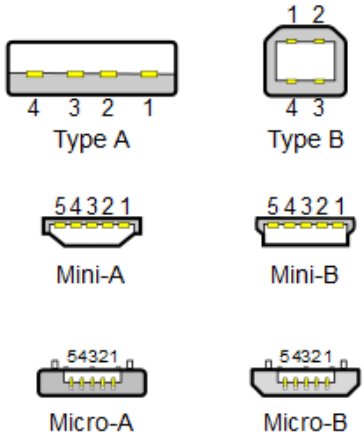
Při použití sériové linky RS-232 v zařízení s TTL nebo CMOS obvody, je nutné signály napěťově upravit, protože napěťové úrovně RS-232 nejsou přímo slučitelné s úrovněmi běžných logických obvodů. Pro úpravu napěťové logiky se nejčastěji používá obvod MAX232 od firmy MAXIM. Pro funkci tohoto obvodu stačí napájecí napětí +5 V a potřebná napětí si samostatně vyrobí pomocí 4 externích kondenzátorů. Obvod samostatně konvertuje logickou „0“ a logickou „1“ na potřebné napěťové úrovně.

2.3.2 USB

USB (Universal Serial Bus) je univerzální sériová sběrnice, představující moderní způsob připojení periferií k počítači. Nahrazuje dříve používané způsoby připojení (sériový a paralelní port) pro běžné druhy periférií (tiskárna, myš, klávesnice, …). Výhodou je možnost připojování Plug & Play bez nutnosti restartování počítače nebo ručního instalování ovladačů. Zařízení lze připojit za chodu k počítači a během několika sekund je přístupné.

Základní vlastnosti:

- Komunikační vzdálenost do 5 m
- Kabel obsahuje 4 vodiče. Dva jsou pro napájení (+5 V a zem). Druhý pár je kroucený a slouží pro přenos dat.
- Možnost připojení více zařízení
- Komunikační rychlost od 1,5 Mbit/s do 480 Mbit/s, pro USB 3.0 až do 5 Gbit/s



Obr. 4 – Konektory USB [4]

Verze USB:

USB 1.1:

Ve verzi USB 1.1 existují pomalá (Low-Speed) zařízení s přenosovou rychlostí 1,5 Mbit/s a rychlá zařízení (Full-Speed) s rychlostí 12 Mbit/s. USB 1.1 však nebylo schopno konkurovat vysokorychlostním rozhraním.

USB 2.0:

V roce 1999 se začalo uvažovat o druhé generaci USB, která by byla použitelná i pro náročnější zařízení (např. digitální kamery). Tato nová verze, označovaná jako USB 2.0,

přišla v roce 2000 a nabídla maximální rychlost 480 Mbit/s v režimu Hi-Speed, avšak zachovala zpětnou kompatibilitu s USB 1.1.

USB 3.0:

Třetí verze se začíná rozvíjet roku 2010. USB 3.0 disponuje více než 10× větší rychlostí, přenosová rychlost je 5 Gbit/s. Nová technologie má 9 vodičů namísto původních 4 (datové vodiče jsou již 4), přesto zpětně podporuje USB 2.0 a slibuje možnou nižší spotřebu energie. Díky tomu je možné používat libovolnou kombinaci zařízení a portů USB 2.0 a USB 3.0 [4].

Přenos dat:

Pro veškerou komunikaci mezi počítačem a funkční jednotkou jsou k dispozici tři typy paketů. Každá výměna dat začíná tím, že počítač vyšle tzv. token packet obsahující popis typu a směru výměny dat, adresu USB zařízení a číslo koncové jednotky (endpoint number). Pak zařízení, které má vysílat data, vyšle datový paket nebo indikuje, že žádná data nejsou k dispozici. Přijímací strana nakonec vyšle tzv. handshake packet, kterým informuje, zda přenos proběhl úspěšně.

Existují dva typy přenosového modelu:

Tok dat (stream) využívající izochronní přenos dat v reálném čase. Nemá přesně definovanou strukturu.

Zpráva (message) využívající asynchronní přenos. Má přesnou strukturu:

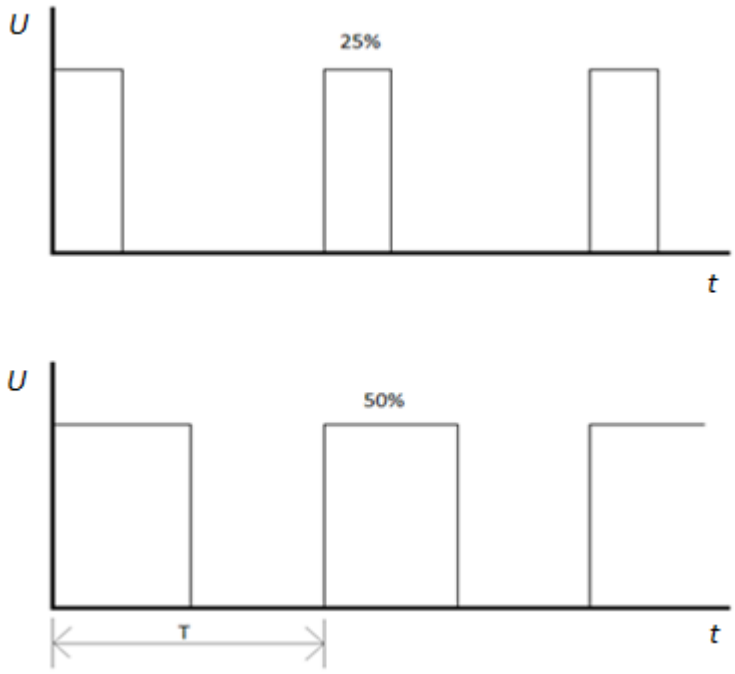
- Řídicí zpráva určená pro konfigurování poprvé aktivovaného zařízení.
- Zpráva obsahující větší objem dat (např. pro tiskárnu nebo plotter), jež je většinou segmentována do více částí.
- Zpráva s přerušením (obvykle několik bajtů), kterou spontánně vysílá zařízení, aby předalo zprávu o svém stavu (např. změna polohy myši).

Pro kódování je ve všech případech použit kód NRZI (not-return-to-zero recording).

Pro připojení USB rozhraní k aplikaci existuje relativně jednoduchá cesta. Pomocí integrovaných obvodů firmy FTDI lze skrz USB snadno připojit do širokého spektra aplikací. Základní použití obvodů sběrnici USB před konstruktérem skrývá. Z hlediska PC se pak jeví jako standardní (ovšem velmi rychlý) sériový port, z hlediska zařízení se jeví buď jako sériový port (typ '232) nebo jako osmibitová obousměrná sběrnice (typ '245) [5].

2.4 Pulzně šířková modulace (PWM)

Pulzně šířková modulace, neboli PWM (anglicky Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového signálu. Jako dvouhodnotová veličina může být použito například napětí, proud, nebo světelný tok. Signál je přenášen pomocí střídy. Pro demodulaci takového signálu pak stačí dolnofrekvenční propust. Vzhledem ke svým vlastnostem je pulzně šířková modulace často využívána ve výkonové elektronice, pro řízení velikosti napětí nebo proudu. Kombinace PWM modulátoru a dolnofrekvenční propusti bývá rovněž využívána jako levná náhrada D/A převodníku.



Obr. 5 – PWM signál se střídou 25 % a 50 %

Přenosový signál, který nese informaci o přenášené hodnotě, může nabývat hodnot zapnuto/vypnuto, to znamená logickou 1 nebo logickou 0. Hodnota přenášeného signálu je v přenosu "zakódována" jako poměr mezi stavy zapnuto/vypnuto. Tomuto poměru se říká střída. Cyklu, kdy dojde k přenosu jedné střídy, se říká perioda *T*. Omezením pro PWM je to, že přenos informace je vždy omezen na relativní vyjádření a to 0 - 100 %, to znamená, že musí být znám poměr mezi skutečnou hodnotou a procentuálním vyjádřením. Časové hodnoty periody se pohybují v sekundách, v milisekundách pro přesnější řízení. Perioda je vždy součtem doby zapnuto a vypnuto [6].

2.5 Převodník A/D a D/A

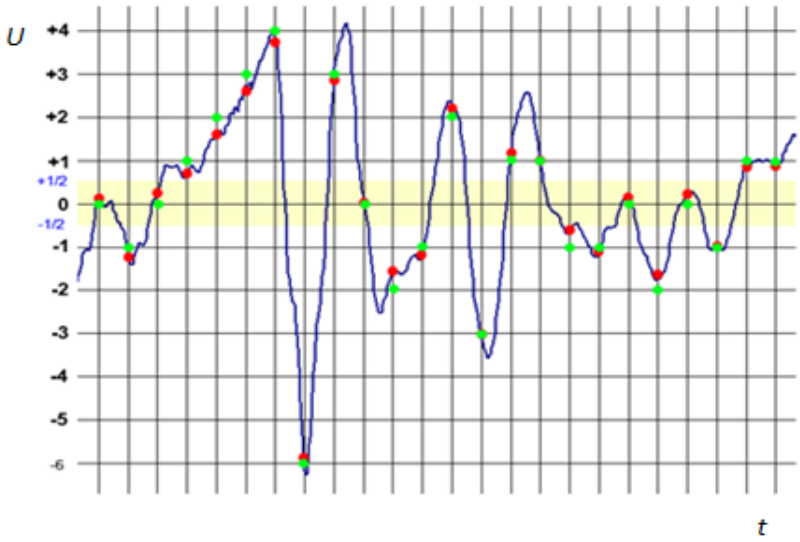
V této kapitole budou popsány základní vlastnosti a principy převodu A/D převodníků a D/A převodníků.

2.5.1 A/D převodníky

Analogově digitální převodník (zkratky A/D, v angličtině i ADC) je elektronická součástka určená pro převod spojitého (neboli analogového) signálu na signál diskrétní (digitální). Důvodem tohoto převodu je umožnění zpracování původně analogového signálu na číslicových počítačích. V digitální podobě se také dají signály daleko kvalitněji zaznamenávat a přenášet. Opačný převod z digitálního signálu na analogový zajišťuje D/A převodník.

Základní princip převodu:

Převod spojitého signálu na diskrétní se skládá ze dvou fází. Nejprve se provede vzorkování signálu, a potom následuje kvantování.



Obr. 6 – Vzorkování a kvantování signálu [7]

Vzorkování se provede tím způsobem, že rozdělíme vodorovnou osu signálu (v našem příkladu je na této ose čas) na rovnoměrné úseky a z každého úseku odebereme jeden vzorek (na obrázku jsou tyto vzorky znázorněny červenými kolečky). Je přitom zřejmé, že tak z původního signálu ztratíme mnoho detailů, protože namísto spojité čáry, kterou lze donekonečna zvětšovat dostáváme pouze množinu diskrétních bodů s intervalem odpovídajícím použité vzorkovací frekvenci. Chyba vzorkování může být ovšem ještě daleko horší. Pokud se totiž v původním spojitém signálu vyskytuje frekvence vyšší než je polovina vzorkovací frekvence (nazývaná též Nyquistova frekvence), dojde, jak praví Shannonův

teorém, k překrytí frekvenčních spekter vzorkovaného signálu a tedy ke ztrátě informace. Při rekonstrukci signálu pak může dojít k vzniku neexistující informace, která má za následek úplné a nenávratné zkreslení rekonstruovaného signálu díky jevu nazývajícímu se aliasing. Aliasingu se dá zabránit jedině takzvaným antialiasing filtrem, což je dolní propust zařazená před převodníkem. Ta nedovolí frekvencím vyšším než je Nyquistova frekvence vstoupit do převodníku.

Vzhledem k tomu, že počítače a další zařízení dále zpracovávající digitální signál umí vyjádřit čísla pouze s omezenou přesností, je potřeba navzorkované hodnoty upravit i na svislé ose. Protože se hodnota vzorku dá vyjádřit pouze po určitých kvantech, nazýváme tuto fázi A/D převodu kvantování. Na obrázku může veličina na svislé ose například nabývat pouze celočíselných hodnot. Aby bylo možné určit, které hodnoty má po kvantování nabývat určitý vzorek, je třeba rozdělit prostor kolem jednotlivých hodnot na toleranční pásy (jeden takový pás je naznačen kolem hodnoty 0). Kterémukoliv vzorku, který padne do daného tolerančního pásu, je při kvantování přiřazena daná hodnota. Kvantované hodnoty jsou na obrázku naznačeny zelenými kolečky. Jak je vidět, že kvantované hodnoty se ve většině případů liší od skutečných navzorkovaných hodnot. Velikost kvantizační chyby je vzdálenost mezi kvantovanými a původními navzorkovanými body, na obrázku ji vyjadřují délky pomyslných úseček mezi červenými a zelenými kolečky. Velikost této chyby se pohybuje v intervalu +1/2 až -1/2 kvantizační úrovně.

Protože se digitální signál zpravidla zpracovává na zařízeních pracujících ve dvojkové číselné soustavě, bývají počty kvantizačních úrovní A/D převodníků zpravidla rovny N-té mocnině čísla 2, přičemž nakvantovaný signál pak lze vyjádřit v N bitech [7].

Typy A/D převodníků:

- Komparační (paralelní, s postupnou komparací)
- Kompenzační (čítací, sledovací, s postupnou aproximací)
- Integrační (s dvojitou integrací)

2.5.2 D/A převodníky

Digitálně analogový převodník (zkratky D/A, v angličtině i DAC) je elektronická součástka určená pro převod digitálního (diskrétního, tj. nespojitého) signálu na signál analogový (spojitý).

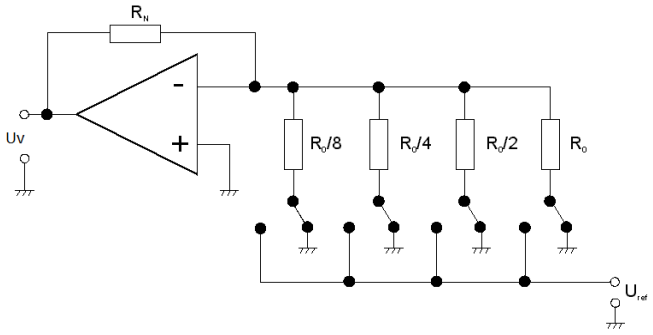
Základní princip převodu:

Tyto převodníky slouží k převedení číselné hodnoty *D* obvykle ve dvojkové soustavě na odpovídající hodnotu výstupního napětí. Jednotlivým bitům vstupního čísla odpovídají stavy spínačů převodníku. Číslo ve dvojkové soustavě můžeme napsat

$$D = z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \ldots + z_1 \cdot 2^1 + z_0 \cdot 2^0 = \sum_{i=0}^{n} z_i \cdot 2^i \qquad (1)$$

Pak bude *i*-tý bit odpovídat *i*-témů spínači převodníku. Pro $z_i = 1$ je tento spínač sepnut, pro $z_i = 0$ rozepnut. Čili pomocí řady rezistorů o hodnotách odporů odstupňovaných ve dvojkové soustavě odvodíme z referenčního zdroje U_R proudy příslušné velikosti, které se přivedou do sčítacího bodu invertorového zesilovače. Výstupní napětí U_V je úměrné součtu těchto proudů. Které proudy se sčítají, to určuje stav spínačů. Napětí na spínačích se mění z hodnoty U_R při rozepnutí na hodnotu 0 při sepnutí. To zmenšuje jejich rychlost – spínač vykazuje parazitní kapacitu a navíc se mění zatížení podle hodnoty vstupního čísla.

Všechny tyto nevýhody odstraňuje jiné provedení převodníku. Místo spínačů je zde použito přepínačů. Hodnotě $z_i = 0$ odpovídá *i*-tý přepínač přepnutý na zem a hodnotě $z_i = 1$ odpovídá přepínač přepnutý na invertující vstup zesilovače. Zdroj referenčního napětí je v tomto případě zatěžován konstantním odporem, který je tvořený paralelní kombinací všech náhradních odporů převodníku. Protože je konstantní, nemusí mít zdroj U_R malý vnitřní odpor [8].



Obr 7 – D/A převodník [8]

Typy D/A převodníků:

- Převodník s R-2R sítí
- Převodník s váhovou odporovou sítí

2.6 LCD displej

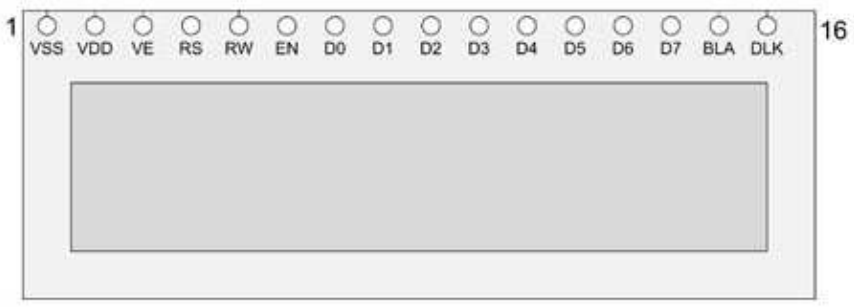
LCD displeje se u nás objevily koncem 80 let. Tehdy se považovaly za zázrak techniky. Dnes jsou LCD zobrazovací displeje všude okolo nás. Setkáváme se s nimi při koupi jízdenky na autobus nebo vlak, v telefonních automatech, na parkovištích ve faxech atd. Každý displej má v sobě speciální integrovaný obvod - řadič, který ovládá celý displej a komunikuje s okolím. Standardem se stal řadič HD44780 od firmy HITACHI. Všichni výrobci používají HD44780 nebo jeho ekvivalent. Proto je jedno, od jakého výrobce displej je. Pro dosažení plné kompatibility displeje, je zapojení přívodního konektoru stejné. To zjednodušuje použití displejů v praxi. Displej se vyrábí v několika variantách, které se liší počtem sloupců a řádků. Moduly se vyrábí s podsvícením LED nebo výbojkou.

Komunikace s LCD displejem:

Pro komunikaci s displejem je potřeba minimálně 6 vodičů až maximálně 11 vodičů a napájení +5 V. Před zapojením s procesorem si musíme zvolit datovou komunikaci. Komunikace s modulem muže být po 8-bitové sběrnici (DB0-DB7) nebo 4-bitové sběrnici (DB4-DB7). Po zapnutí procesoru a displeje je nutné provést základní inicializaci displeje. Nastaví se počet bitů datové komunikace (8 bitů nebo 4 bity), směr psaní znaků, posun řádku, pozice kurzoru a blikání kurzoru. Po této inicializaci je displej připraven na komunikaci.

- **8-bitová sběrnice**. Pošle se 8 bitů a zapíšou se signálem Enable
- **4-bitová sběrnice**. Tuto komunikaci je vhodné použít při nedostatku vývodů na procesoru. Komunikace bude 2× pomalejší, protože se data posílají nadvakrát. Nejdříve se pošlou vyšší 4 bity a zapíšou se signálem Enable a pak nižší 4 bity a zapíšou se signálem Enable. Nižší 4 bity (DB0-DB3) na displeji je potřeba spojit se zemí.

Zapojení vývodů LCD displeje:



Obr. 8 – LCD displej [9]

V následující tabulce jsou uvedeny názvy signálů a jejich popis.

Tab. 2 - Vývody LCD displeje

Vývod	Název	Popis
1	Vss	GND
2	Vdd	+5 V
3	Vo	Kontrast (0 V až 5 V)
4	RS	Registr Select (0 = instrukce, 1 = data)
5	R/W	Read / Write (0 = zápis, 1 = čtení)
6	E	Enable
7	DB0	Datová sběrnice 0
8	DB1	Datová sběrnice 1
9	DB2	Datová sběrnice 2
10	DB3	Datová sběrnice 3
11	DB4	Datová sběrnice 4
12	DB5	Datová sběrnice 5
13	DB6	Datová sběrnice 6
14	DB7	Datová sběrnice 7
15	A	Podsvícení – anoda
16	K	Podsvícení – katoda

Zobrazení znaků:

Každý znak je zobrazován jako matice 5 × 8 bodů. Definice jednotlivých znaků je uložena napevno ve vnitřní paměti ROM. Dále je možno zobrazit kurzor, nastavit blikání kurzoru, definovat posouvání zobrazených znaků, smazat displej, zobrazovat na konkrétní pozici apod. [9].

3 Praktická část

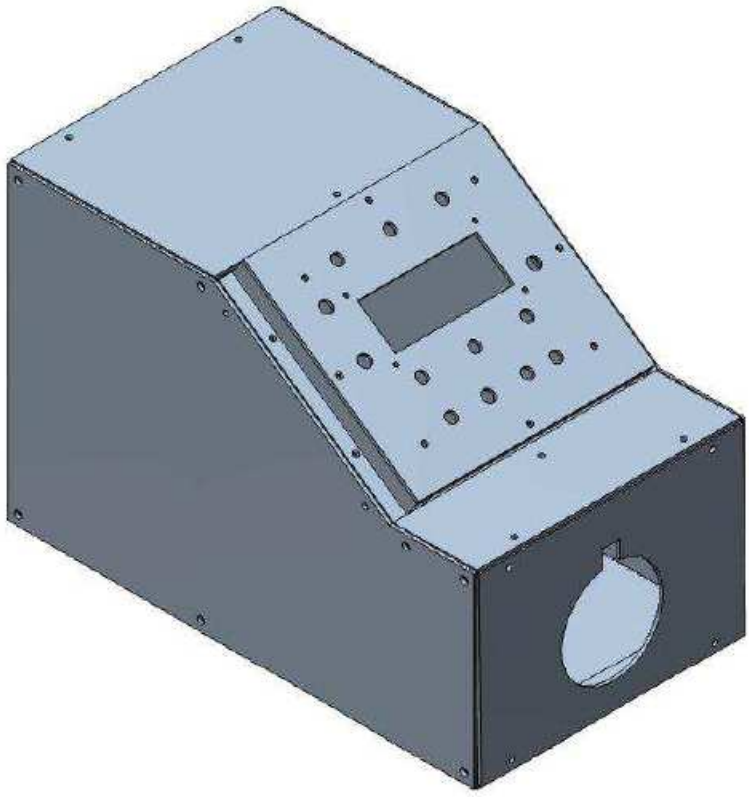
V této části bude popsán konstrukční návrh mechanické části větrného tunelu a návrh elektronické části řídící jednotky pneumatického systému.

3.1 Mechanická část

V této jsou popsány konstrukční řešení větrného tunelu a desky plošných spojů.

3.1.1 Větrný tunel

Větrný tunel byl navržen tak, aby obsahoval dostatečný prostor pro větrné proudění a bylo možné do něj umístit desku plošných spojů. Větrný tunel je vyroben z ocelových plechů a hliníkových profilů. Na následujícím obrázku je vidět, jak větrný tunel vypadá.



Obr. 9 – Větrný tunel [10]

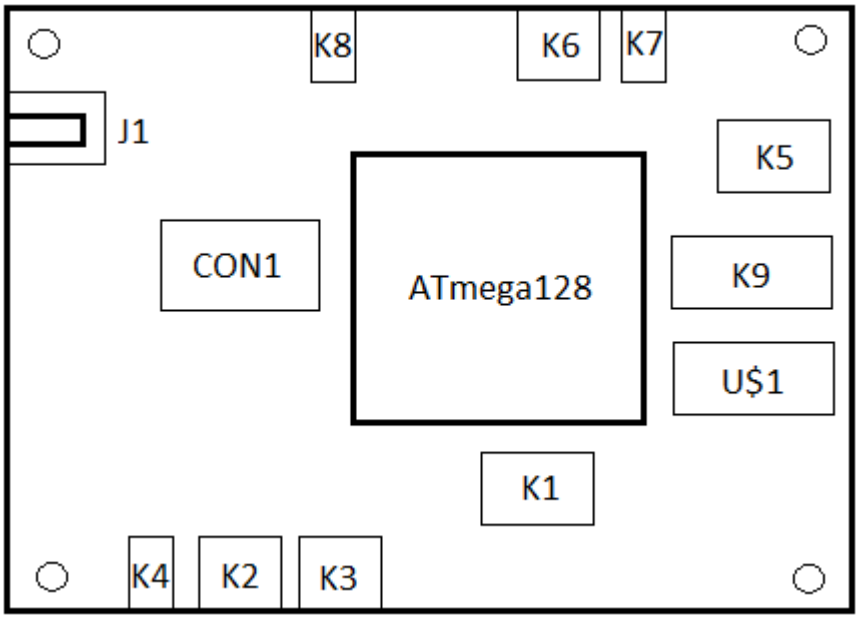
Jak je vidět na obrázku, tak dominantním prvkem celého větrného tunelu je ovládací panel, který byl umístěn na zkosenou hranu větrného tunelu, pro pohodlnější práci s ním. V přední části větrného tunelu se pak nachází kulatý otvor pro sekundární (menší) ventilátor o

rozměrech 40 × 40 mm s optickým čidlem. V zadní části se pak nachází větší kulatý otvor pro primární (velký) ventilátor o rozměrech 120 × 120 mm s optickým čidlem. V zadní části větrného tunelu se ještě nachází otvor pro konektor, který slouží k připojení napájení a konektor pro připojení USB.

Pro mou bakalářskou práci byl tento větrný tunel pouze využit. Návrhem a konstrukcí větrného tunelu se zabýval pan Drápalík a veškerou dokumentaci k tomuto větrnému tunelu naleznete v jeho diplomové práci v přiložených přílohách [10].

3.1.2 Návrh desky plošných spojů

Aby bylo možné umístit desku plošných spojů do větrného tunelu, je nutné, aby její rozměry byly maximálně 106 × 120 mm. Dále bylo žádoucí, aby se na desce plošných spojů nacházel jak řídicí systém, tak i měřící systém. Deska plošných spojů je spojena s ostatními částmi řídící jednotky pomocí konektorů. Na následujícím obrázku je znázorněno, na kterém místě desky plošných spojů se nacházejí jednotlivé konektory.



Obr. 10 – Rozvržení konektorů na DPS

Navržená deska plošných spojů má rozměry 106 × 120 mm, tedy maximální možnou velikost. Dále se v rozích desky plošných spojů nacházejí díry o průměru 3,2 mm, kvůli upevnění do větrného tunelu. Na desce plošných spojů jsou použity různé typy konektorů, typy konektorů a signály na jejich pinech jsou zaznamenány v následující tabulce.

Tab. 3 – Použité konektory

Konektor	Typ konektoru	Pin	Funkce	Poznámka
K1	PSH02-04PG	1	LED 3	Spojení s LED diodami na ovládacím panelu.
		2	LED 2	
		3	LED 1	
		4	+5 V	
K2	PSH02-03WG	1	LED (GND)	Optické číslo od sekundárního ventilátoru.
		2	Fototranzistor	
		3	+5 V	
K3	PSH02-03WG	1	LED (GND)	Optické číslo od primárního ventilátoru.
		2	Fototranzistor	
		3	+5 V	
K4	PSH02-02WG	1	PWM (+12 V)	Řízení otáček primárního ventilátoru.
		2	GND	
K5	PSH02-04PG	1	GND	Spojení s tlačítky na ovládacím panelu.
		2	Tlačítko 1 (GND/+5 V)	
		3	Tlačítko 2 (GND/+5 V)	
		4	Tlačítko 3 (GND/+5 V)	
K6	PSH02-03WG	1	Žádaná (0 - 5 V)	Spojení se zdířkou na ovládacím panelu.
		2	Měřená (0 - 5 V)	
		3	GND	
K7	PSH02-02WG	1	Signál z ventilátoru	Doplňující měření na malém ventilátoru
		2	GND	
K8	PSH02-02WG	1	+5 V	Spojení s potenciometrem.
		2	GND	
K9	PSH02-05PG	1	GND	Spojení s přepínači na ovládacím panelu.
		2	Přepínač 3 (GND/+5 V)	
		3	Přepínač 2 (GND/+5 V)	
		4	Přepínač 1 (GND/+5 V)	
		5	+5 V	
U$1	PSH02-05PG	2	Data -	Digitální komunikace přes USB.
		3	Data +	
		4	GND	
		5	GND	

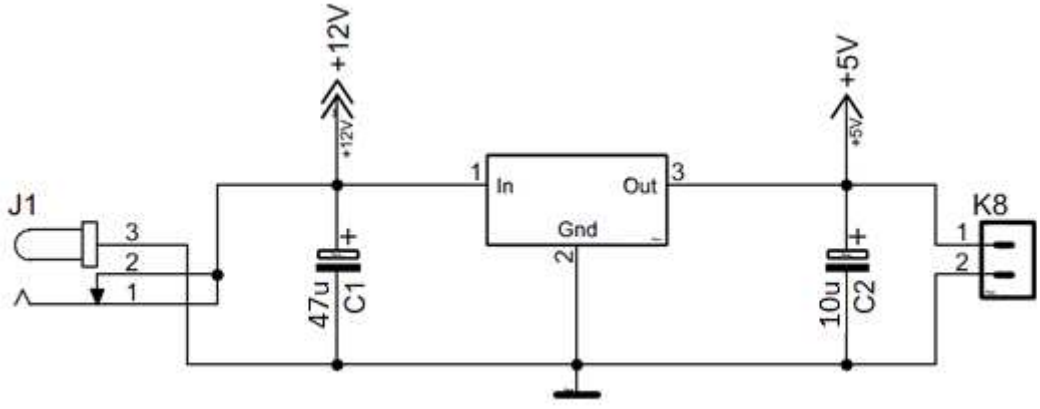
CON1	MLW10G	1	+5 V	Komunikace přes SPI s LCD displejem.
		2	SS	
		6	MOSI	
		8	SCK	
		9	+5 V	
		10	GND	
J1	K36724A	1	+12 V	Napájecí konektor.
		2	GND	

V sloupci funkce jsou uvedeny v závorkách napěťové úrovně.

3.2 Napájení

Pro napájení celého zařízení jsou potřebné dvě napěťové úrovně a to 12 V a 5 V. Napěťová úroveň 12 V je potřebná pro napájení hlavního ventilátoru a pro napájení vývodové desky s ATmega128. Pro ostatní komponenty je potřebné napětí 5 V.

Proto byl vybrán pro napájení standardní stabilizovaný síťový adaptér 12 V/1,5 A. Napěťové úrovně 5 V je dosaženo pomocí stabilizátoru napětí L7805, zapojeného podle následujícího obrázku.

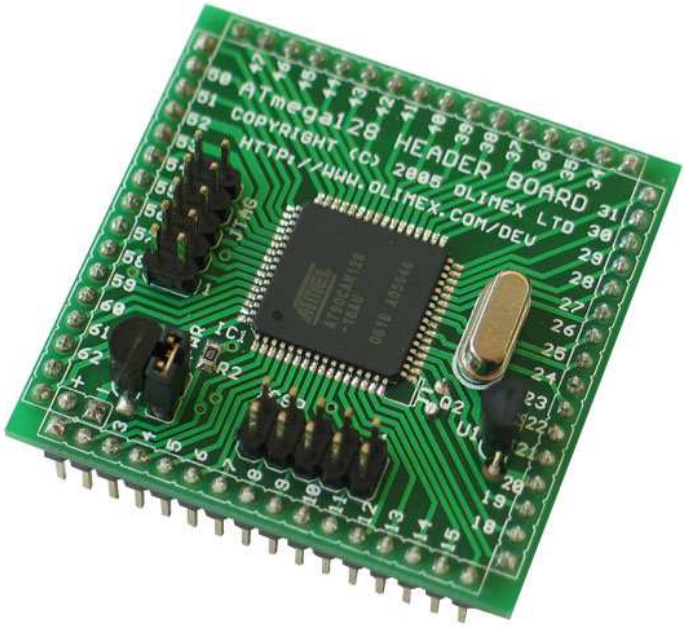


Obr. 11 – Schéma zapojení stabilizátoru napětí L7805

Ke stabilizátoru napětí L7805 jsou ještě připojeny paralelně dva elektrolytické kondenzátory pro stabilizaci napětí.

3.3 Vývodová deska ATmega128

Jelikož by bylo obtížné na desku plošného spoje napájet mikrokontrolér ATmega128 v SMD provedení, tak byla použita vývodová deska AVR-H128-C od firmy OLIMEX.



Obr. 12 – Vývodová deska ATmega128 [11]

Tato vývodová deska obsahuje kromě samotného mikrokontroléru ATmega128 ještě vlastní stabilizátor napětí LM78L05, který vytváří napětí 5 V pro napájení mikrokontroléru. Deska obsahuje také obvod resetu ZM33064, obvod oscilátoru s 16 MHz krystalem a obvod oscilátoru s 32768 Hz krystalem. Dále se na desce nachází dvojice konektorů JTAG a ICSP pro programování mikrokontroléru. Každému pinu mikrokontroléru ATmega128 je přiřazen jeden vývod na desce. Deska má ještě dva vývody navíc, ty se používají pro připojení napájení k vstupu stabilizátoru LM78L05. Vývodová deska má rozměry 47 mm × 47 mm a s deskou plošného spoje bude spojena pomocí dutinových lišt.

Použité piny vývodové desky a mikrokontroléru ATmega128

Jak již bylo řešeno, tak každému vývodu vývodové desky je přiřazen jeden pin mikrokontroléru ATmega128. V následující tabulce jsou popsány použité vývody vývodové desky a tudíž i piny mikrokontroléru ATmega128.

Tab. 4 – Použité piny vývodové desky a mikrokontroléru ATmega128

Pin	ATmega128	Funkce
+	----------	Vstupní napájení vývodové desky (+12 V)
-	----------	Vstupní napájení vývodové desky (GND)
6	PE4 (INT4)	Tlačítko +
7	PE5 (OC3C)	Generování PWM signálu
10	PB0	Signál SS pro SPI (LCD displej)
11	PB1 (SCK)	Hodinový signál SPI
12	PB2 (MOSI)	Signál MOSI pro SPI
17	PB7	Signál SS pro SPI (D/A převodník)
19	PG4	Kontrolní LED dioda
25	PD0 (INT0)	Tlačítko
26	PD1 (INT1)	Tlačítko -
27	PD2 (RXD1)	Digitální komunikace (RxD)
28	PD3 (TXD1)	Digitální komunikace (TxD)
29	PD4 (IC1)	Měření otáček ventilátorů
30	PD5	Ovládací signál pro přepnutí mezi ventilátory
31	PD6	LED dioda na ovládacím panelu (LED 1)
32	PD7	LED dioda na ovládacím panelu (LED 2)
33	PG0	Přepínač na ovládacím panelu (Přepínač 1)
34	PG1	Přepínač na ovládacím panelu (Přepínač 2)
35	PC0	Přepínač na ovládacím panelu (Přepínač 3)
36	PC1	LED dioda na ovládacím panelu (LED 3)
37	PC2	Signál pro řízení digitální komunikace (CTS)
38	PC3	Signál pro řízení digitální komunikace (RTS)
58	PF3 (ADC3)	Analogové napětí na ovládacím panelu
59	PF2 (ADC2)	Doplňující měření napětí na malém ventilátoru
62	AREF	Referenční napětí pro A/D převodník

3.4 Měření otáček ventilátorů

Měření otáček ventilátorů zajišťují dvě čidla s LED diodou a fototranzistorem. LED dioda i fototranzistor pracují v oblasti infračerveného záření, z důvodu odstranění vlivu okolního osvětlení. Tyto čidla jsou připevněna k ventilátorům.

Princip měření:

Při otáčení ventilátoru dochází k tomu, že lopatky ventilátoru střídavě stíní světlo dopadající na bázi fototranzistoru. Dopadající světlo vyvolá změnu napětí na kolektoru tranzistoru a právě změna tohoto napětí indikuje zatmění, či přímé osvětlení fototranzistoru.

Z takového signálu lze snadno spočítat rychlost otáčení ventilátoru, například pomocí následujícího vztahu

$$rychlost\ otaceni = \frac{pocet\ pulzu\ za\ 1\,s}{pocet\ lopatek} \cdot 60 \qquad [\text{ot/min}] \qquad (2)$$

Tento způsob měření je vhodný pro měření pouze vyšších otáček. Pro dosažení vyšší přesnosti při nižších otáčkách je potřeba delší interval měření signálu čidla a tedy nízkou frekvenci aktualizace informace o otáčkách. Proto byl využit jiný způsob měření otáček – měření doby zaclonění signálu jednou lopatkou ventilátoru. V tomto případě lze určit rychlost otáčení již po pootočení ventilátoru o jednu lopatku. Přesnost určení rychlosti je dána přesností měření časového intervalu mezi dvěma po sobě jdoucími náběžnými hranami signálu čidla. Při měření doby zaclonění s využitím časovače mikrokontroléru je nutno zvolit kompromis mezi přesností měření a minimálními měřenými otáčkami (přetečení časovače).

Pro výpočet rychlosti otáčení ventilátoru, za pomocí změřeného času lze použít následující vztah

$$rychlost\ otaceni = \frac{1}{t \cdot pocet\ lopatek} \cdot 60 \qquad [\text{ot/min}] \qquad (3)$$

kde t - čas pootočení ventilátoru o jednu lopatku v sekundách

Pro zjištění doby t pootočení ventilátoru o jednu lopatku se bude používat 16-ti bitový časovač mikrokontroléru, to znamená, že bude moci čítat do 65535 a poté přeteče. Jelikož bude použít 16 MHz oscilátor, bude nutné použít předděličku kmitočtu, aby nedocházelo k častému přetečení časovače. Použití předděličky sníží přesnost měření, neboť se zvýší

perioda inkrementace časovače. Pro výpočet času T inkrementace čítače lze použít následující vztah

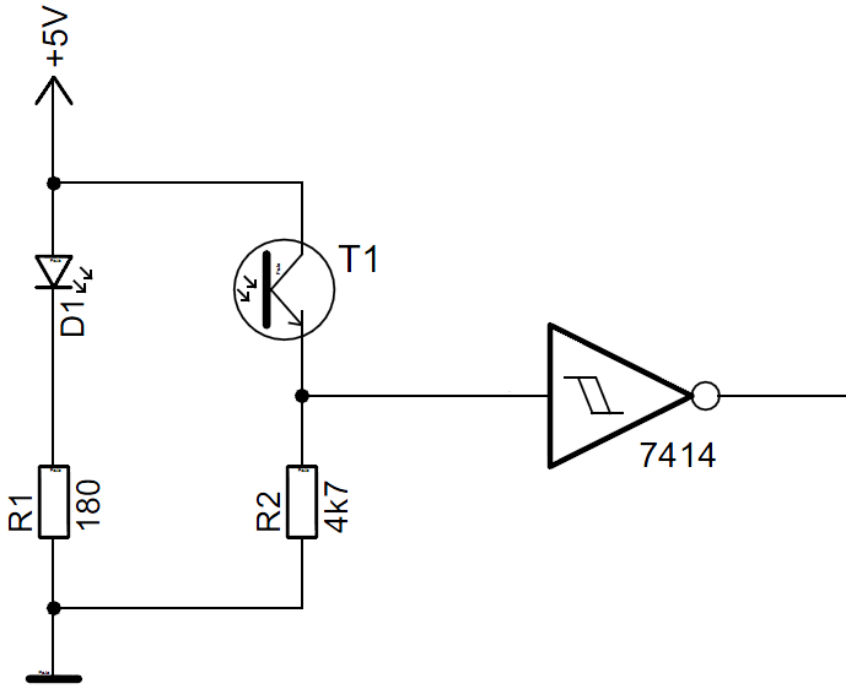
$$T = \frac{1}{\frac{f_{OSC}}{preddelicka}} \quad [s] \qquad (4)$$

kde f_{OSC} - frekvence oscilátoru, který využívá mikrokontrolér

Z výše uvedeného vztahu je patrné, že čím bude předdělička kmitočtu větší, tím bude větší i doba inkrementace časovače a bude snížena přesnost měření. Pro zjištění doby t pootočení ventilátoru o jednu lopatku lze použít následující vztah

$$t = pocet\ \ pulzu \cdot T \quad [s] \qquad (5)$$

Signál z čidla je ještě upraven na TTL logiku pomocí Schmittova klopného obvodu. Zapojení čidla je na následujícím obrázku.

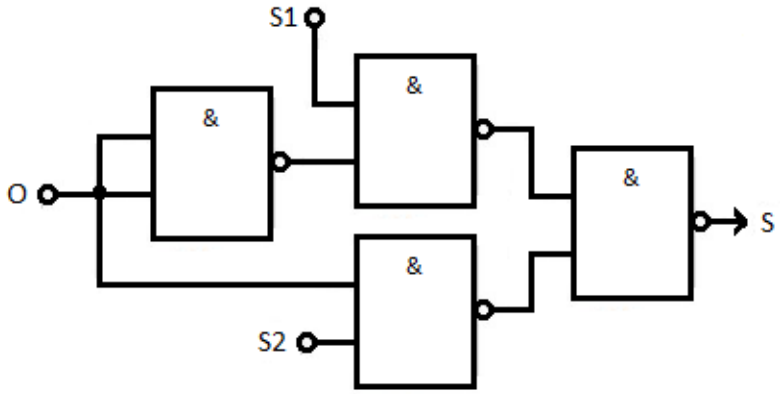


Obr. 13 – Schéma zapojení snímání otáček

Pro oba ventilátory je zapojení stejné.

Signál z čidel je poté přiveden na pin 29 (ICP1) mikrokontroléru ATmega128. Tento pin je vstup Input Capture Unit čítače/časovače1. Jelikož se měří signály od obou čidel na tomto pinu, je nutné mezi těmito signály přepínat. Přepínání signálů od optických čidel je prováděno

logickým obvodem SN74HCT00. Tento obvod obsahuje čtveřici hradel NAND, které jsou zapojeny podle následujícího obrázku.



Obr. 14 – Zapojení hradel NAND

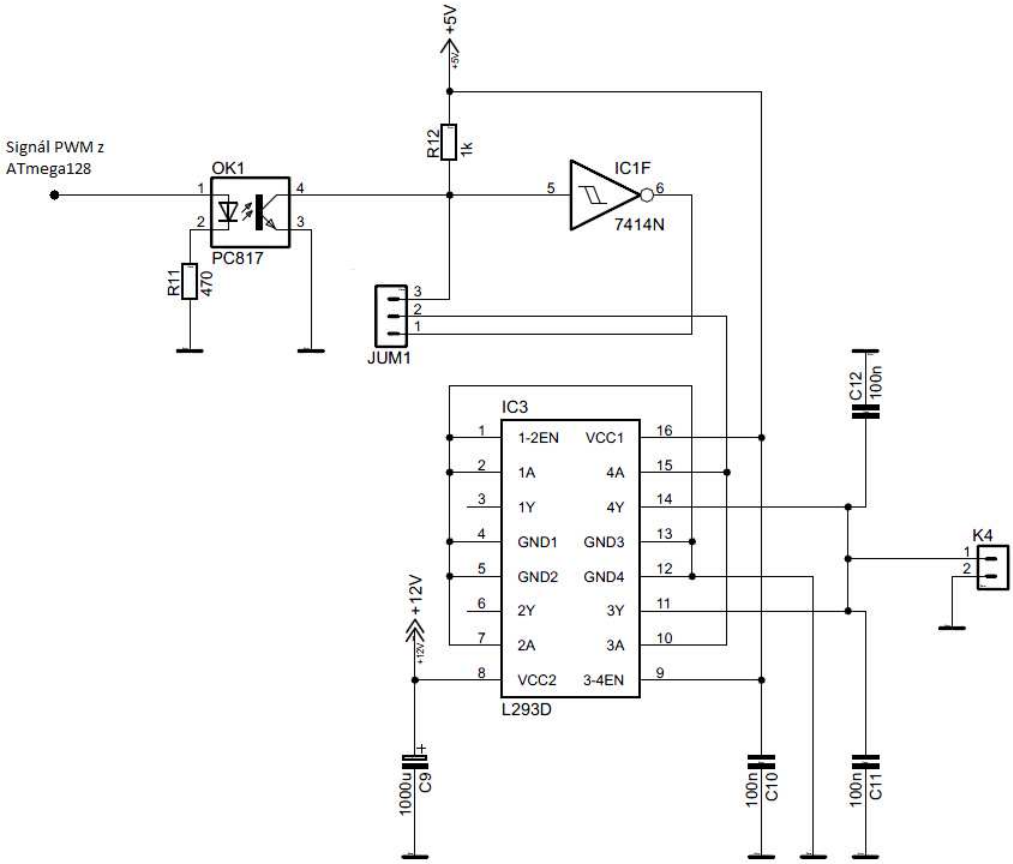
Signál označený (O) je ovládací signál, který je generován na pinu 30 (PD5) mikrokontroléru ATmega128. Logická úroveň signálu (O) určuje, ze kterého čidla (S1, S2) bude přiveden signál na výstup (S), tudíž na vstup ICP1. V případě, že na ovládacím signálu (O) bude logická 1, bude na výstup (S) přiveden signál z čidla (S2). V opačném případě, tedy na ovládacím signálu (O) bude logická 0, bude na výstup (S) přiveden signál z čidla (S1).

3.5 PWM signál

Pomocí PWM signálu bude ovládán primární ventilátor. Rychlost otáčení tohoto ventilátoru bude dána hodnotou příkonu na základě signálu generovaného na mikrokontroléru ATmega128. Mikrokontrolér ATmega128 bude tento PWM signál generovat na pinu 7 pomocí čítače/časovače. Hodnota příkonu je dána požadavkem na rychlost otáčení primárního nebo sekundárního ventilátoru. Příkon je ovládán sepnutím, nebo rozepnutím napájecího napětí na základě PWM signálu. Změna střídy PWM signálu umožňuje měnit příkon primárního ventilátoru v rozsahu 0 - 100 %. Pro střídu 100 % bude pak rychlost otáčení primárního ventilátoru největší, naopak tomu bude při střídě 0 %.

Pro napájení primárního ventilátoru se používá napětí 12 V, a jelikož PWM signál generovaný mikrokontrolérem ATmega128 má napěťové úrovně TTL, tedy 0 V nebo 5 V, je nutné zapojení ještě upravit, aby bylo možné spínat napětí 12 V. Pro toto spínání byl použit

obvod L293D, v zapojení s optočlenem PC817 oddělujícím řídicí PWM signál podle následujícího obrázku.

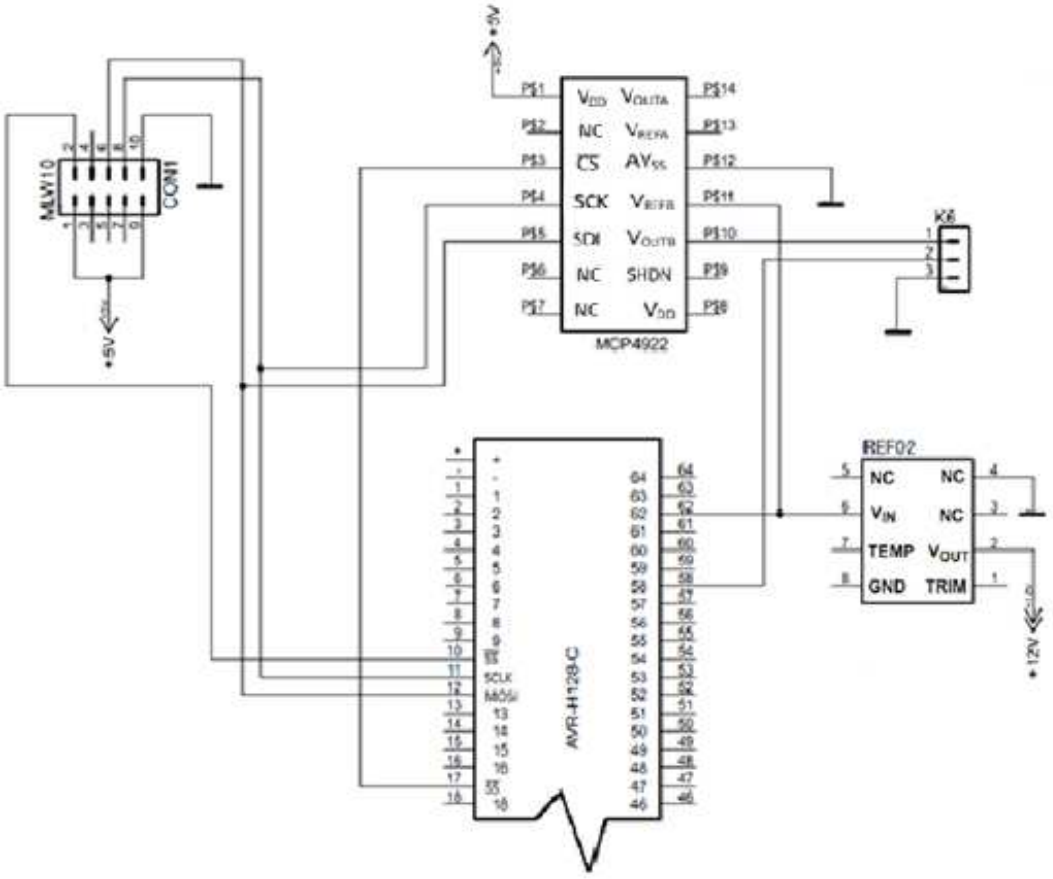


Obr. 15 – Ovládání primárního ventilátoru

PWM signál generovaný na pinu 7 mikrokontroléru ATmega128 je nejprve přiveden na optočlen PC817, který galvanicky odděluje ovládací napětí od spínaného napětí. Dále je pak tento signál upraven na TTL logiku pomocí Schmittova klopného obvodu. Samotný obvod L293D je napájen 5 V s využitím odrušovacích kondenzátorů. Spínané napětí 12 V je přivedeno na svorku VCC2 obvodu L293D. Obvod L293D pak bude spínat toto napětí podle přivedeného PWM signálu a přivádět ho na svorku K4, která je výstupem pro primární ventilátor. Zapojení je ještě doplněno o přepínač s označením JUM1, který slouží k změně směru otáčení primárního ventilátoru.

3.6 Sběrnice SPI

Pomocí sběrnice SPI komunikuje mikrokontrolér ATmega128 s LCD displejem a s D/A převodníkem MCP4922. Schéma zapojení je na obrázku níže.



Obr. 16 – Zapojení LCD displeje a D/A převodníku

3.6.1 LCD displej

Jelikož je LCD displej uzpůsobený pro komunikace pomocí paralelního přenosu, je nutné, aby LCD displej mohl komunikovat s mikrokontrolérem ATmega128 přes rozhraní SPI (což je sériová sběrnice) provést konverzi ze sériového přenosu na paralelní. K této konverzi byl využit adaptér od společnosti Mikroelektronika s expandérem MCP23S17. Adaptér má vyvedeny piny se signály potřebnými pro 4 bitové ovládání LCD displeje. Tudíž lze tento adaptér připojit přímo k LCD displeji. Pro komunikace s mikrokontrolérem ATmega128 přes SPI má tento adaptér vyveden plochý kabel zakončený konektorem IDC10. Na tomto konektoru jsou vyvedeny nezbytné signály pro komunikace přes sběrnici SPI.

Obr. 17 – Adaptér s MCP23S17 [12]

Na desce plošného spoje bude tedy jen 10–ti pinový konektor, který zajišťuje komunikaci mezi mikrokontrolérem ATmega128 a expandérem MCP23S17, tudíž i LCD displejem. Schéma zapojení konektoru je na obrázku 16, konektor je pojmenovaný jako CON1. Ke konektoru jsou přivedeny signály nezbytné pro komunikaci přes sběrnici SPI a to MOSI (pin 12), SCLK (pin 11) a signál SS (pin 10). Dále je ke konektoru ještě zapojeno napájení 5 V a zem.

3.6.2 D/A převodník MCP4922

Tento digitálně-analogový převodník má 12-ti bitové rozlišení, SPI s podporou až 20 MHz frekvencí hodinového signálu, rychlé nastavení výstupu 4,5 µs a široký teplotní rozsah.

Výstupní analogové napětí lze vypočítat pomocí následujícího vztahu

$$V_{\mathrm{OUT}} = \frac{V_{\mathrm{REF}} \cdot G \cdot D_N}{2^n} \qquad [\mathrm{V}] \qquad (6)$$

kde G - volitelný zisk (1× nebo 2×)

D_N - vstupní digitální hodnota

n - rozlišení převodníku, u tohoto převodníku n = 12

Komunikaci mezi převodníkem a mikrokontrolérem zajišťuje SPI sběrnice. Mikrokontrolér posílá D/A převodníku přes SPI sběrnici 12-ti bitové číslo, které následně D/A převodník převede na analogové napětí.

Obr. 18 – D/A převodník MCP4922 [13]

Zapojení D/A převodníku MCP4922 je na obrázku 16. Jelikož se jedná o jednostrannou komunikaci, tak jsou ke komunikaci nezbytné signály MOSI (pin 12), SCLK (pin 11) a signál SS (pin 17). Dále je k převodníku ještě zapojeno napájecí napětí 5 V, zem a referenční napětí z obvodu REF02, tento obvod zajišťuje přesné referenční napětí 5 V (s tolerancí ±0,3 %). Toto referenční napětí je přivedeno i k mikrokontroléru ATmega128 na pin 62 (AREF), tedy vstup pro referenční napětí integrovaného A/D převodníku mikrokontroléru ATmega128.

3.7 Ovládací panel

Ovládací panel zajišťuje komunikaci mezi uživatelem a zařízením. Díky ovládacímu panelu je možné nastavit dostupné funkce systému, na LCD displeji sledovat změřené, nebo zadané hodnoty, popřípadě připojit externí regulátor.

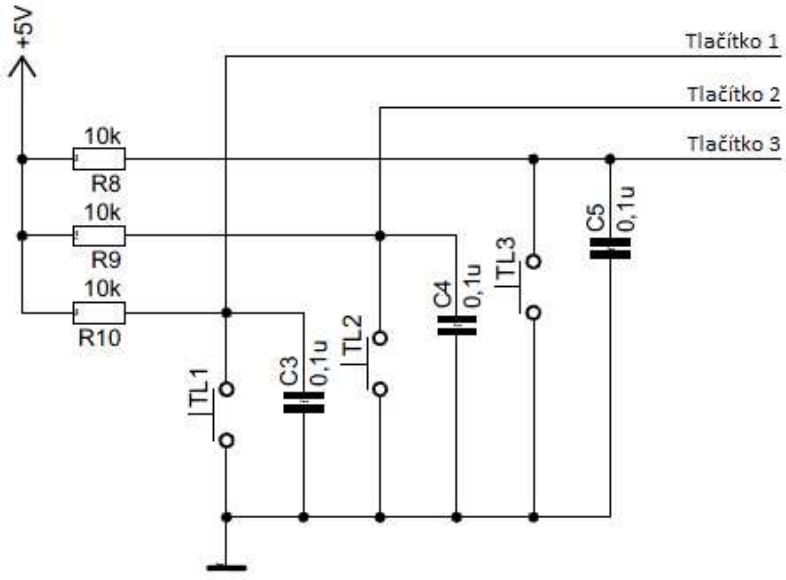
Na ovládacím panelu nalezneme trojici tlačítek, které slouží ke změně na LCD displeji zobrazované hodnoty a jejímu zvyšování či snižování. Dále se na panelu nachází trojice přepínačů, tyto přepínače slouží k přepínání stavu programu Digital / Analog, Remote / Local a Automat/ Manuál. Na ovládacím panelu se ještě nachází trojice LED diod, které signalizují aktuálně používaný stav. Ve spodní řadě ovládacího panelu se ještě nacházejí dva konektory typu CINCH, potenciometr a přepínač. Na levý konektor je vyvedeno napětí generované D/A převodníkem, které je úměrné aktuálním otáčkám ventilátoru. Napětí přivedené na pravý

konektor nebo odváděné z jezdce potenciometru (podle polohy přepínače) je měřeno A/D převodníkem a slouží jako externí řídicí signál s významem podle zvoleného stavu programu.

Obr. 19 – Ovládací panel [14]

3.7.1 Zapojení tlačítek

Tlačítka jsou připevněna k ovládacímu panelu a s deskou plošného spoje jsou spojena konektorem. Zapojení tlačítek zajišťující definovaný stav signálu jak při sepnutí tak i při rozepnutí je na následujícím obrázku.

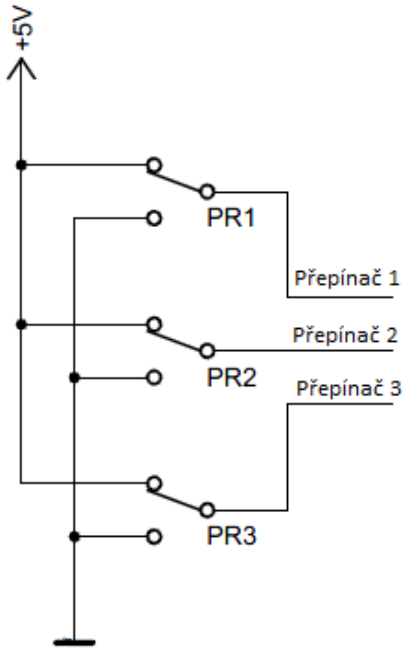


Obr. 20 – Zapojení tlačítek

Tlačítka jsou přes odpory R_8, R_9, R_{10} spojena s napájecím napětím 5 V. Při stisku tlačítka dojde ke změně napětí na výstupu a právě touto změnou dostane mikrokontrolér informaci o změně stavu tlačítka. Tlačítko 1 je spojeno s pinem 6 mikrokontroléru ATmega128, tlačítko 2 je spojeno s pinem 25 a tlačítko 3 je spojeno s pinem 26 mikrokontroléru. Všechny tyto piny mikrokontroléru se dají použít jako obecný vstup nebo výstup, ale mohou se také nastavit jako zdroj vnějšího přerušení.

3.7.2 Zapojení přepínačů

Přepínače jsou také pevně spojeny s ovládacím panelem a s deskou plošného spoje jsou spojeny konektorem. Zapojení je na následujícím obrázku.

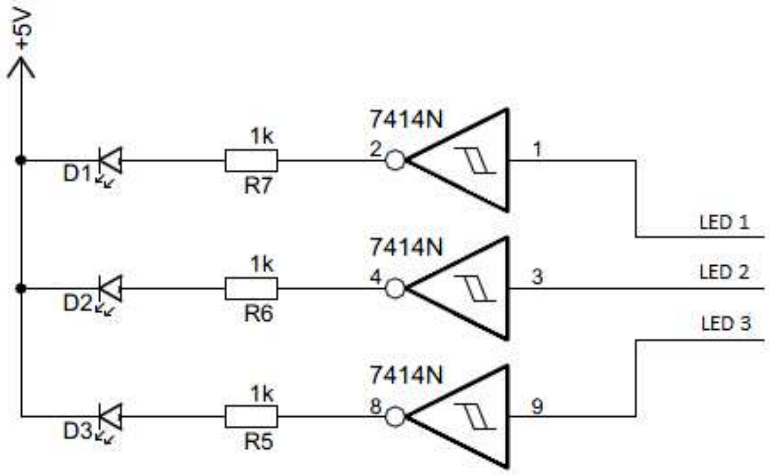


Obr. 21 – Zapojení přepínačů

Jak je vidět na obrázku výše, tak zapojení přepínačů je velice jednoduché. Přepínač je přepnut buď na 5 V, nebo na zem. K zjištění hodnoty, na kterou je přepínač přepnut pak postačí obecný vstupně / výstupní port mikrokontroléru ATmega128, který se nastaví jako vstup. Přepínač 1 je spojen s pinem 33, přepínač 2 je spojen s pinem 34 a přepínač 3 je spojen s pinem 35 mikrokontroléru ATmega128.

3.7.3 Zapojení LED diod

Stejně jako předchozí komponenty, i LED diody jsou připevněny přímo k ovládacímu panelu a s deskou plošného spoje jsou spojeny pomocí konektoru. Zapojení LED diod je na následujícím obrázku.



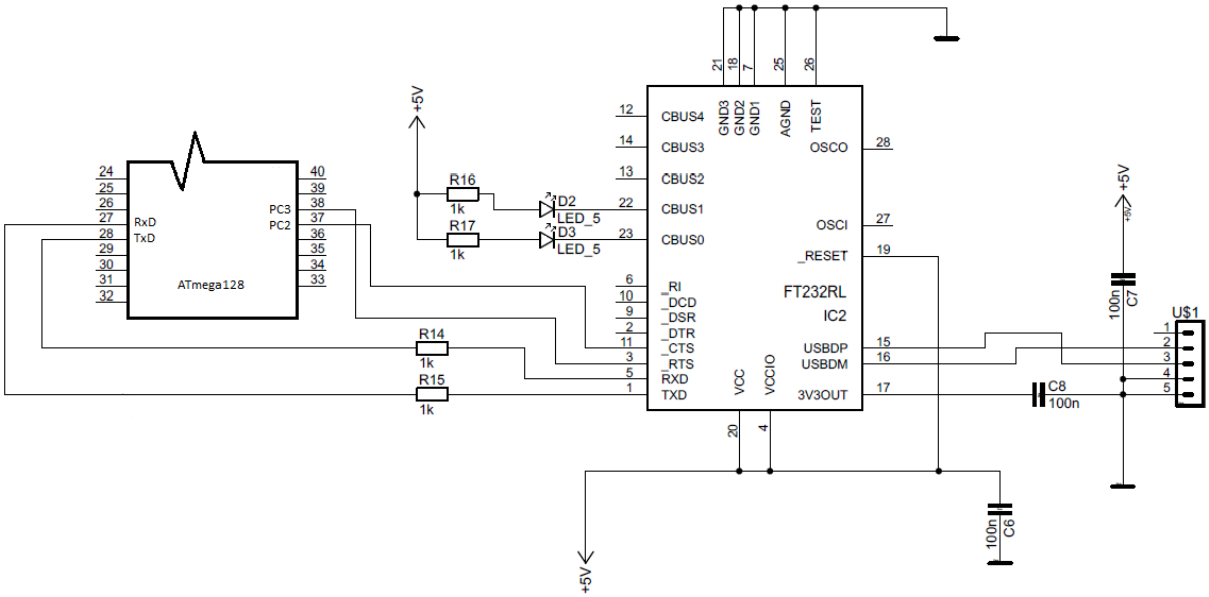
Obr. 22 – Zapojení LED diod

K LED diodám jsou do série zapojeny odpory R_5, R_6, R_7 o velikost 1 kΩ pro omezení protékajícího proudu. LED diody jsou napájeny napětím 5 V. Jelikož v pouzdře Schmittova klopného obvodu 7414N zůstali tři klopné obvody nevyužity, tak byly použity pro spínání LED diod místo klasických tranzistorů. Zároveň toto řešení ušetřilo místo na desce plošného spoje. Diody jsou ovládány obecnými vstupně / výstupními porty mikrokontroléru ATmega128, které se nastaví jako výstupní. Pro rozsvícení LED diody pak stačí na konkrétní port přivést logickou 0. LED 1 je přivedena na pin 31, LED 2 na pin 36 a LED 3 je přivedena na pin 32 mikrokontroléru ATmega128.

3.8 USB

Rozhranní USB bude využito pro komunikaci s PC. Je využito zapojení, které nahrazuje komunikaci přes rozhranní RS232. USB rozhraní je dnes součástí každého osobního počítače a většina osobních počítačů disponuje několika konektory pro připojení přes kabel USB. Na rozdíl od sériového portu RS-232, který požívá konektor D-Sub, tento konektor na dnešních osobních počítačích většinou už nenalezneme.

Jelikož mikrokontrolér ATmega128 nedokáže sám o sobě komunikovat pomocí rozhraní USB, je zapojení doplněno o převodník FT232RL, který převádí sériové USART rozhraní, kterým disponuje mikrokontrolér ATmega128 na USB rozhraní. Zapojení převodníku FT232RL a mikrokontroléru ATmega128 je na následujícím obrázku.



Obr. 23 – Zapojení převodníku FT232RL

Převodník FT232RL využívá dvou datových linek RxD a TxD, kterými je spojen přes odpory 1 kΩ s mikrokontrolérem ATmega128. Datová linka RxD z převodníku je přivedena na pin 28 (TxD) mikrokontroléru ATmega128. Druhá datová linka TxD z převodníku je připojena k pinu 27 (RxD) mikrokontroléru, oba tyto piny mikrokontroléru ATmega128 se dají použít pro komunikaci pomocí USART. Dále jsou z převodníku FT232RL připojeny k mikrokontroléru ještě dva signály (CTS a RTS), které slouží k řízení datového přenosu. Oba tyto signály jsou připojeny k obecným vstupně/výstupním portům mikrokontroléru ATmega128. Signál RTS je připojen k pinu 38 (PC3) a signál CTS k pinu 37 (PC2) mikrokontroléru ATmega128. Dále je převodník FT232RL připojen k napájecímu napětí 5 V a zemi. K převodníku FT232RL jsou ještě připojeny dvě LED diody, které budou blikáním signalizovat, že převodník přijímá, nebo odesílá data. Výstupní signály převodníku FT232RL jsou přivedeny na konektor U$1, k tomuto konektoru bude zapojen kabel, který bude zakončen USB zásuvkou typu A.

Po připojení k osobnímu počítači se převodník FT232RL připojí a zobrazí se jako virtuální COM port ve správci zařízení, což usnadňuje přijímání a odesílání dat z mikrokontroléru ATmega128.

K převodníku FT232RL není potřeba externí krystal, obvod interně generuje hodinový signál o frekvencích 6 MHz, 12 MHz, 24 MHz a 48 MHz, který lze vyvést z obvodu ven a použít ho pro mikrokontrolér, nebo jinou externí logiku. Převodník má integrovanou paměť EEPROM pro ID číslo zařízení a pro popis zařízení. Tento převodník je dodáván s Royalty-Free podporou ovladačů pro Windows, Linux a Mac OS [14].

4 Programová část

V této části bude popsáno programové řešení jednotlivých částí zařízení. Pro ověření funkčnosti zařízení byly vytvořeny jednoduché programy, které byly nahrány do mikrokontroléru ATmega128 a tím byla ověřena funkčnost jednotlivých částí zařízení.

4.1 SPI sběrnice

Jak již bylo řečeno, SPI sběrnice se používá pro komunikaci s LCD displejem a pro komunikaci s D/A převodníkem MCP4922. Aby bylo možné komunikovat s LCD displejem a D/A převodníkem, je nutné nejprve nastavit SPI sběrnici, to znamená nastavit mikrokontrolér ATmega128 jako master, signály MISO, CLK a signály CS pro obě zařízení jako výstupy a předdělička 128. Toto nastavení bude vypadat následovně

```
DDRB |= (1 << PB0) | (1 << PB1) | (1 << PB2) | (1 << PB7);
SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPR1) | (1 << CPOL) | (1 << CPHA);

SetCSLCD;   // PORTB = (1 << PB0);
SetCSDAC;   // PORTB = (1 << PB7);
```

Dále je ještě nutné nastavit signály CS pro obě zařízení do logické 1, to znamená, že komunikace s nimi neprobíhá. Pro zahájení komunikace s některým zařízením musí být tento signál CS nastaven na logickou 0. Pro zahájení přenosu dat pak stačí požadovaná data uložit do registru SPDR, po ukončení přenosu se nastaví bit SPIF do logické 1. Funkce pro vyslání jednoho bytu přes SPI sběrnici vypadá následovně

```
void spiPosliData(char data)
{
      SPDR = data;                  // Uložení dat do registru SPDR
      while(!(SPSR&(1<<SPIF)));     // Čekání dokud se neukončí přenos
}
```

4.1.1 Komunikace s LCD displejem

Jelikož komunikace s LCD displejem probíhá přes expandér MCP23S17, je nutné ho nejprve inicializovat. Pro komunikaci s expandérem je nutné vyslat tři byte, první byte obsahuje opcode, druhý byte adresu a třetí byte obsahuje data. Pomocí adresy určíme, kam se budou data zapisovat, jestli do kontrolních registrů, nebo do vstupních / výstupních portů. Pro vyslání této trojice bytů slouží funkce spiExpSend, která vypadá následovně

```
void spiExpSend(char adr, char data )
{
	ClearCSLCD;		// Signál CS do log. 0 – zahájení komunikace
	spiPosliData(opc);	// Na počátku komunikace s expandérem nutné
vyslat opade – opc = 0b01000000
	spiPosliData(adr);	// Adresa v expandéru na kterou chceme
zapisovat
	spiPosliData(data);	// Data která chceme poslat
	SetCSLCD;		// Signál CS do log. 1 – ukončení komunikace
}
```

Jelikož, s LCD displejem komunikujeme pomocí 4 bitového režimu, je nutné pro zapsání instrukce do paměti, vyslat jeden byte s horním půlbytem kódu instrukce a signál E v 1. Poté opakovat s tou změnou, že signál E nastavíme do 0. Tento postup zopakujeme i pro dolní půlbyte. Je tedy nutné vyslat 4 byty na jeden byte pro displej. Pro expandér je nutné vyslat 3 byty na jeden datový. Celkem dostaneme 12 bytů na jeden znamenající „užitečná“ data pro displej.

Funkce, které zajistí rozdělení bytu na dvě poloviny, tyto poloviny umístí do horní části nových bytů a u nich nastaví hodnoty bitů signálů E a RS a následně je odešle, se jmenují dataDoLcd a instDoLcd. Tyto funkce vypadají následovně

```
void instDoLcd(char data)
{
	int i, j;
	char data2;

	for(j = 0; j < 2; j++)
	{
		data2 = (data & 0b11110000) + 0b00001000;//vynuluje dolní
půlbyte, nastaví E = 1, RS = 0
		for(i = 0; i < 2; i++)
		{
			spiExpSend(expandVyst, data2);	// expandVyst = adresa
výstupního registru IO, expandVyst = 0b00010101
			data2 &= 0b11110111;	// Opětovné vyslaní dat s E=0
		}
	data <<= 4;			// Posun o 4 - vyslání dolního půlbyte
	}
}
```

Výše uvedená funkce slouží k odeslání instrukce do LCD displeje (signál RS = 0). Funkce, která odešle data do LCD displeje je uvedena níže (signál RS = 1).

```
void dataDoLcd(char data)
{
	int i, j;
	char data2;

	for(j = 0; j < 2; j++)
	{
		data2 = (data & 0b11110000) + 0b00001100;//vynuluje dolní
půlbyte, nastaví E = 1, RS = 1
		for(i = 0; i < 2; i++)
		{
			spiExpSend(expandVyst, data2);	// expandVyst = adresa
výstupního registru IO, expandVyst = 0b00010101
			data2 &= 0b11110111;	// Opětovné vyslaní dat s E=0
		}
	data <<= 4;			// posun o 4 - vyslaní dolního půlbyte
	}
}
```

Pro toto zařízení je potřeba nastavit sériový vstup a paralelní výstup na portu B expandéru MCP23S17. Zapojení expandéru umožňuje 4 bitovou komunikaci s LCD displejem.

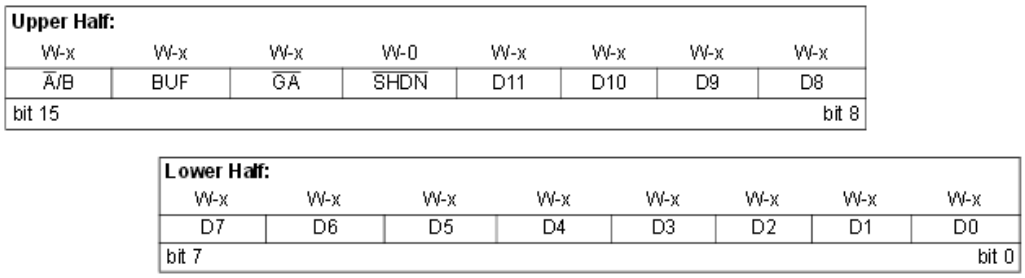
Inicializace expandéru a LCD displeje bude vypadat následovně

```
void inicializaceLCD(void)
{
// Nastavení portu B  expandéru jako výstup
	spiExpSend(0b00000001, 0b00000000);
	_delay_ms(30);

// Nastavení 4 bitového komunikačního režimu a dvouřádkového režimu LCD
displeje
	spiExpSend(expandVyst, 0b00101000);	// expandVyst = 0b00010101
	spiExpSend(expandVyst, 0b00100000);	// expandVyst = 0b00010101
	_delay_us(40);

	instDoLcd(0b00101000);
	_delay_us(40);

// Zapnutí LCD displeje a vypnutí kurzoru
	instDoLcd(0b00001100);
	_delay_us(40);
}
```

Při návrhu funkcí pro práci s LCD displejem jsem vycházel z diplomové práce pana Špinky [15]. Další funkce pro práci s LCD displejem naleznete v příloze E.

4.1.2 Komunikace s D/A převodníkem MCP4922

D/A převodník převádí 12-ti bitové číslo na analogové napětí. Tudíž je nutné přes SPI sběrnici poslat 12 bitů, k těmto 12-ti bitům jsou přidány další 4 bity, které nesou informace o nastavení funkce převodníku. Tedy přes SPI sběrnici posíláme 16 bitů do D/A převodníku a 4 horní bity nesou informace o nastavení převodníku a dalších 12 bitů reprezentuje číslo, které se má převést na analogové napětí.



Obr. 24 – Význam bitů pro komunikaci s D/A převodníkem

bit 15 – A/B (výběr kanálu A nebo B)

log. 0 = výběr kanálu A

log. 1 = výběr kanálu B

bit 14 – BUF (ovládání vstupního bufferu)

log. 0 = vypnuto

log. 1 = zapnuto

bit 13 – GA (nastavení výstupního zisku)

log. 0 = zisk 2×

log. 1 = zisk 1×

bit 12 – SHDN (ovládání vypnutí výstupu)

log. 0 = výstupní buffer zakázán, výstup ve stavu vysoké impedance

log. 1 = zapnuto

Jelikož je registr pro odesílání dat pomocí SPI sběrnice SPDR 8-mi bitový a číslo které bude posláno do D/A převodníku je 16-ti bitové, musí dojít k rozdělení tohoto čísla a následně odeslat nejprve horních 8 bitů a poté spodních 8 bitů. Před zahájením komunikace se musí signál CS pro D/A převodník nastavit do logické 0 a po ukončení přenosu dat zpět do logické 1. Toto zajišťuje funkce posliDataDAC, která vypadá následovně

```
void posliDataDAC(unsigned short data)
{
      char data1, data2;

      data2 = data;                 // data2 = spodních 8 bitů z data
      data >>= 8;             // posun data doprava o 8
      data1 = data;                 // data1 = horních 8 bitů z data
      data1 |= 0b11110000;          // bit 15, 14, 13, 12 = 1


      //------Vyslání dat-----
      ClearCSDAC;                   // Signál CS do 0
      SPDR = data1;                       // Odeslaní horních 8 bitů
      while(!(SPSR & (1 << SPIF)));// Čekání dokud se neukončí přenos
      SPDR = data2;                       // Odeslaní spodních 8 bitů
      while(!(SPSR & (1 << SPIF)));// Čekání dokud se neukončí přenos
      SetCSDAC;                     // Signál CS do 1
}
```

4.2 Generování PWM signálu

Signál PWM je generován za pomocí čítače/časovače3, který je 16-ti bitový. Nastavení čítače/časovače3 vypadá následovně

```
DDRE |= (1 << PE5);     // Nastavení PE5 jako PWM výstup
// Nastavení fastpwm s TOP v OCR3A
TCCR3A = (1 << COM3C1) | (1 << WGM31) | (1 << WGM30);
TCCR3A &= ~(1 << COM3C0);
// Nastaveni předděličky /8 => jeden krok trvá 0.5us
TCCR3B = (1 << WGM32) | (1 << WGM33) | (1 << CS31);
ETIMSK = (1 << TOIE3);  // Povolení přerušení od čítače/časovače3
OCR3A = 20000;          // Nastavení TOP (maxima) na 20000
OCR3C = 0;              // Střída 0% => větrák se netočí
```

Čítač/časovač3 je nastaven jako fastpwm s TOP (maximální hodnotou) uloženou v registru OCR3A, dále je použita předdělička kmitočtu $f_{OSC}/8$, což znamená, že jeden krok čítače bude trvat

$$t = \frac{1}{\frac{f_{OSC}}{8}} = \frac{1}{\frac{16 \cdot 10^6}{8}} = 0{,}5 \cdot 10^{-6} \quad [\text{s}] \tag{7}$$

Jelikož je horní hranice čítače/časovače3 nastavena na 20000 a jeden krok trvá 0,5 µs, tak to znamená, že perioda (střída) bude 10 ms. Na začátku je registr OCR3C nastaven na hodnotu

0, což znamená, že je střída = 0 % a tudíž se velký ventilátor nebude točit. Dále je u tohoto čítače/časovače povoleno přerušení při přetečení, obsluha tohoto přerušení vypadá následovně

```
ISR(TIMER3_OVF_vect)    // Obsluha přerušení od čítače/časovače3 - PWM
{
      OCR3C = nastavStridu;   // Do registru OCR3C zapsaní střídy
}
```

Aby se velký ventilátor roztočil, je potřeba nastavit registr OCR3C na jinou hodnotu, než 0 a tato hodnota musí být menší, nebo rovna 20000. Nastavení tohoto registru probíhá v obsluze přerušení. Pro výpočet střídy lze použít následující vzorec

$$strida = \frac{hodnota\ OCR3C}{hodnota\ OCR3A} \cdot 100 \qquad [\%] \qquad (8)$$

Z výše uvedeného vztahu vyplývá, že čím bude hodnota registru OCR3C větší, tím bude i střída větší a tudíž se bude i velký ventilátor točit rychleji.

4.3 Měření otáček ventilátoru

Měření otáček ventilátoru bude realizováno pomocí měření doby zaclonění signálu jednou lopatkou ventilátoru, z čehož bude vypočítána rychlost otáčení ventilátoru. Pro měření doby zaclonění jednou lopatkou bude použit čítač/časovač1. Bude měřen časový interval mezi dvěma po sobě jdoucími náběžnými hranami signálu čidla, který vyvolá změnu na vstupu Input Capture. Nastavení čítače/časovače1 s využitím Input Capture vypadá následovně

```
// Nastaveni hrany detekce pro Input Capture, předdělička /1024 => jeden
krok trvá 64us
TCCR1B = (1 << ICES1) | (1 << CS10) | (1 << CS12);
// Povoleni přerušení od události Input Capture čítače/časovače1
TIMSK = (1 << TICIE1) | (1 << TOIE1);
```

Pro čítač/časovač1 byla zvolena předdělička kmitočtu $f_{OSC}/1024$, což znamená, že jeden krok čítače/časovače1 bude trvat

$$t = \frac{1}{\frac{f_{OSC}}{1024}} = \frac{1}{\frac{16 \cdot 10^6}{1024}} = 64 \cdot 10^{-6} \qquad [s] \qquad (9)$$

Příchozí náběžná hrana z čidla vyvolá přerušení od Input Capture, které bude obslouženo pomocí následující obsluhy přerušení

```
ISR (TIMER1_CAPT_vect)          // Obsluha přerušení od čítače/časovače1
{                               // (vyvoláno změnou Input Capture)
      namereno = TCNT1;         // Do proměnné namereno se uloží hodnota
                                TCNT1 (počet kroků)
      TCNT1 = 0;                // Počítá se vždy od 0
}
```

Při tomto přerušení dojde k tomu, že se do proměnné *namereno* uloží hodnota TCNT1, což je počet inkrementací čítače/časovače1. Z této hodnoty se poté vypočítá rychlost otáčení ventilátoru. Pro tento výpočet lze použít vztah

$$rychlost\ otaceni = \frac{1}{t \cdot namereno \cdot pocet\ lopatek} \cdot 60 \qquad [\text{ot/min}] \qquad (10)$$

kde *t* je doba inkrementace čítače/časovače1

Z výše uvedeného vztahu lze stanovit i minimální otáčky ventilátoru, které je systém schopný změřit. Ventilátor, který je použit ve větrném tunelu má 7 lopatek a jelikož je čítač/časovač1 16-ti bitový, tak může čítat maximálně do 65535. Při dosazení těchto hodnot do vztahu (10) lze tedy stanovit minimální otáčky ventilátoru, který je systém schopný změřit.

$$rychlost\ otaceni_{\min} = \frac{1}{64 \cdot 10^{-6} \cdot 65535 \cdot 7} \cdot 60 = 2{,}04 \qquad [\text{ot/min}] \qquad (11)$$

Minimální rychlost otáčení, kterou lze systémem změřit je tedy 2,04 ot/min.

5 Závěr

Návrh elektronické části řídící jednotky pneumatického systému proběhl bez větších komplikací. U měřícího systému se vyskytli problémy s měřenými signály, ale tyto problémy byly odstraněny po úpravě elektronického zapojení optických čidel. Řídící jednotka dokáže ovládat za pomocí spínání PWM signálu rychlost otáčení primárního ventilátoru a zároveň dokáže měřit rychlost otáčení na obou ventilátorech. Řídící jednotka také dokáže komunikovat s ovládacím panelem a je připravena pro připojení externího regulátoru. Řídící jednotka pneumatického systému byla ještě doplněna o digitální komunikaci, konkrétně se jedná o rozhraní USB. Díky této komunikaci bude možné provádět další experimenty. Pro odzkoušení funkčnosti jednotlivých částí řídící jednotky pneumatického systémy byly vytvořeny jednoduché programy v jazyce C. V budoucnu bude řídící jednotka pneumatického systému doplněna o komplexnější software.

Literatura

[1] ATmega128-16AI. In: *GM Electronic* [online]. [cit. 2013-05-06]. Dostupné z: http://www.gme.cz/mikroprocesory-atmel-avr-mega/atmega128-16ai-atmega128-16au-p958-080/

[2] Serial Peripheral Interface. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 11. 3. 2013 [cit. 2013-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/SPI

[3] RS-232. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 9. 3. 2013 [cit. 2013-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/RS-232

[4] Universal Serial Bus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 27. 4. 2013 [cit. 2013-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/USB

[5] USB. In: ŘEHÁK, Jan. *Hw.cz* [online]. 2009 [cit. 2013-05-06]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/usb/usb-universal-serial-bus-popis-rozhrani.html

[6] Pulzně šířková modulace. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 10. 3. 2013 [cit. 2013-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/Pulzn%C4%9B_%C5%A1%C3%AD%C5%99kov%C3%A1_modulace

[7] A/D převodník. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 9. 3. 2013 [cit. 2013-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/A/D_p%C5%99evodn%C3%ADk

[8] AD a DA převodníky. In: *Johnykovy poznámky* [online]. 2008, 4. 1. 2008 [cit. 2013-05-06]. Dostupné z: http://jjohnyk.sweb.cz/elektrotechnika/23.htm

[9] STROLENÝ, Jaroslav. Znakové LCD displeje. In: *Doveda.byl.cz* [online]. 2007, 25.2.2007 [cit. 2013-05-06]. Dostupné z: http://doveda.byl.cz/lcd/index.htm

[10] DRÁPALÍK, M. *Pneumatický systém – konstrukce, měření, matematický model: diplomová práce.* Pardubice: Univerzita Pardubice, Fakulta elektrotechniky a informatiky, 2012. 61 stran.

[11] AVR-H128-CAN. In: *Olimex* [online]. [cit. 2013-05-06]. Dostupné z: https://www.olimex.com/Products/AVR/Header/AVR-H128-CAN/

[12] Serial LCD/GLCD Adapter Board. In: *RLX COMPONENTS* [online]. [cit. 2013-05-06]. Dostupné z: http://www.rlx.sk/product.php?id_product=515&id_lang=1

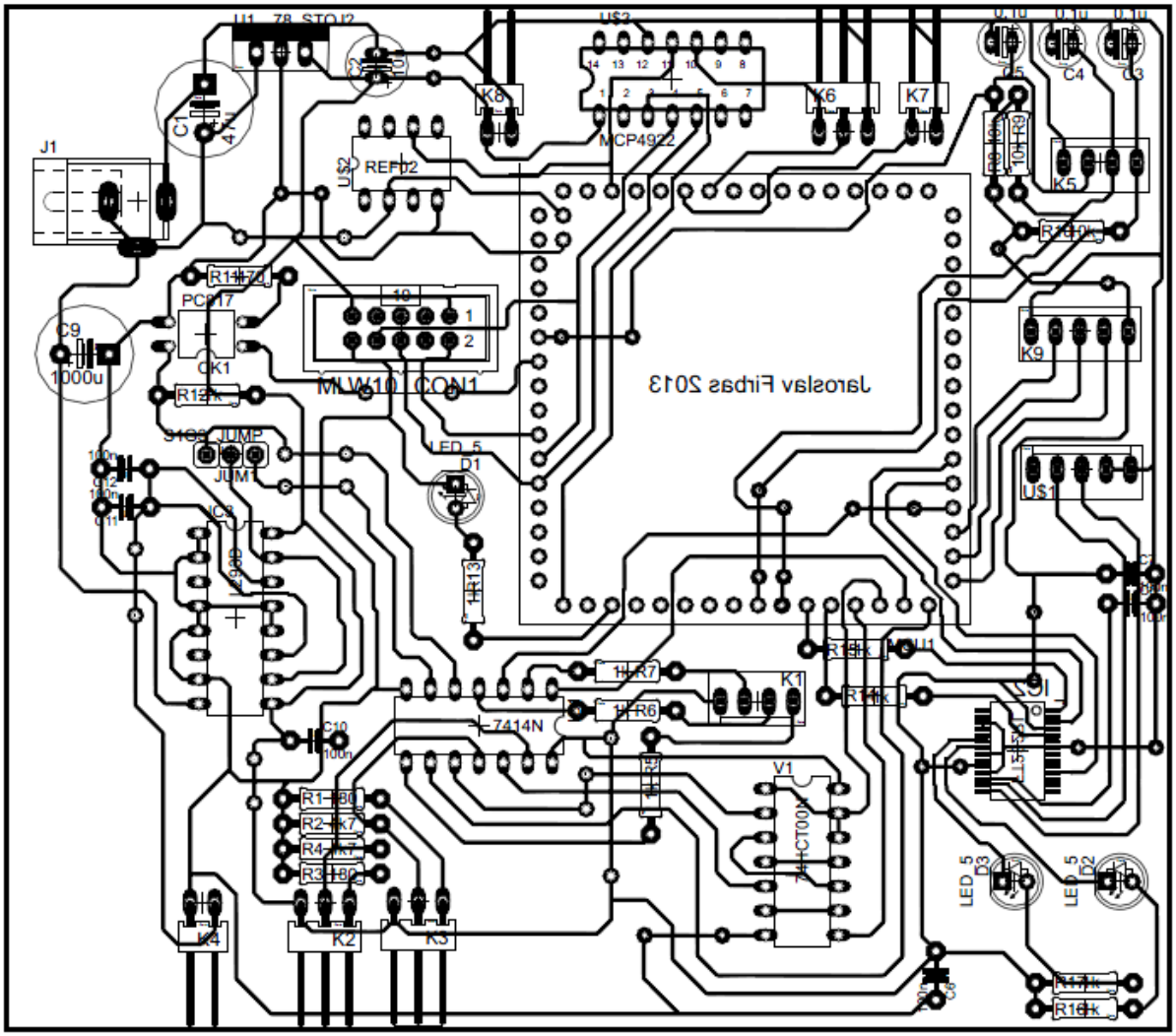
[13] MCP4922-E/P. In: *GM Electronic* [online]. [cit. 2013-05-06]. Dostupné z: http://www.gme.cz/da-prevodniky-seriove/mcp4922-e-p-p321-017/#dokumentace

[14] FT232RL USB to Serial Adapter for PIC AVR ATMEGA ARDUINO MCUs. In: *Electronics-DIY.com* [online]. [cit. 2013-05-06]. Dostupné z: http://electronics-diy.com/FT232RL_USB_to_Serial_Adapter_for_PIC_AVR_ATMEGA_ARDUINO_MCUs.php

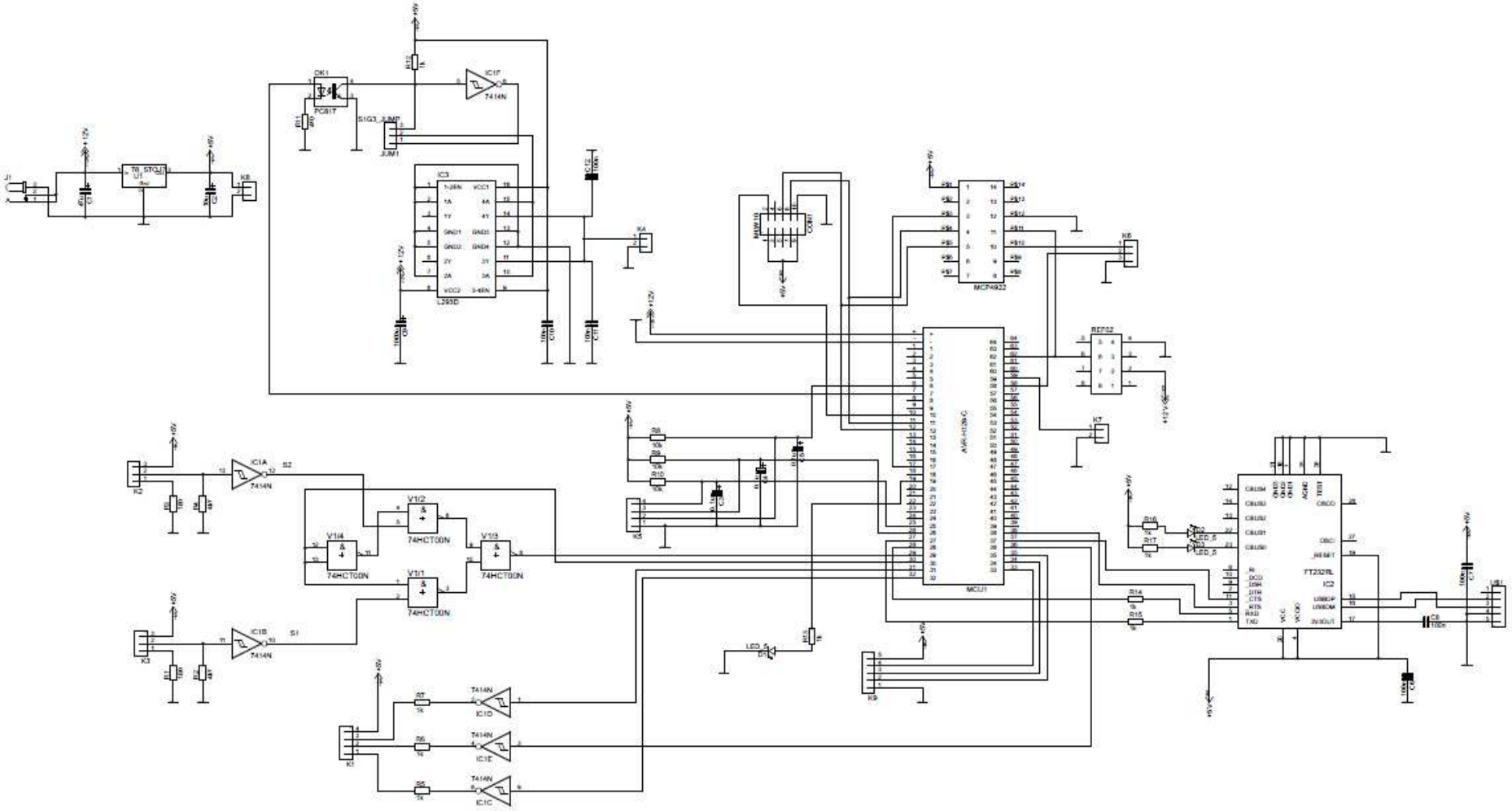
[15] ŠPINKA, R. *Řídicí jednotka a programové vybavení pro pneumatické systémy: diplomová práce.* Pardubice: Univerzita Pardubice, Fakulta elektrotechniky a informatiky, 2012. 65 stran.

Přílohy

Příloha A – Návrh desky plošných spojů



Příloha B – Schéma zapojení řídící jednotky pneumatického systému



Příloha C – Program pro ověření funkčnosti D/A převodníku

```
#define F_CPU 16000000
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/iom128.h>
#include <util/delay.h>

#define SetCSDAC     PORTB |= (1 << PB7);        // Nastavení signálu CS do log. 1
#define ClearCSDAC   PORTB &= ~(1 << PB7);       // Nastavení signálu CS do log. 0

int main(void)
{
       double napeti = 0.2;                      // Žádaná hodnota napětí na výstupu
       unsigned short vypocitaneNapeti = 0;      // Vypočítaná hodnota napětí

       DDRB|=(1<<PB1)|(1<<PB2)|(1<<PB7); // Nastavení CLK, CS, MISO, CSDAC jako výstupy
       SPCR|=(1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<CPOL)|(1<<CPHA);      //Povoleni SPI,
       jako master, preddelicka /128
       SetCSDAC;            // Nastavení signálu CS do log. 1
       DDRG |= (1 << PG4);  // Port PG4 nastav jako výstup

       vypocitaneNapeti = (short) ((napeti * 4096) / 5);      // Výpočet hodnoty napětí
       posliDataDAC(vypocitaneNapeti);          // Odeslání vypočítaného napětí na D/A
                                                převodník

    while(1)
    {
        PORTG |= (1 << PG4);      // Cyklus blikani kontrolni diody
        _delay_ms(300);
        PORTG &= ~(1 << PG4);
        _delay_ms(300);
       }
       return 0;
}

void posliDataDAC(unsigned short data)          // Funkce pro odeslání dat D/A
převodníku
{
       char data1, data2;

       data2 = data;              //data2 = spodni byte z data
       data >>= 8;                // posun data doprava o 8
       data1 = data;              // data1 = horni byte z data
       data1 |= 0b11110000;       // bit 15, 14, 13, 12 = 1

       //------Vyslani dat-----
       ClearCSDAC;                        // Nastavení signálu CS do log. 0
       SPDR = data1;                      // Odeslání horních 8-mi bitů
       while(!(SPSR & (1 << SPIF)));      // Čekání na ukončení přenosu
       SPDR = data2;                      // Odeslání spodních 8-mi bitů
       while(!(SPSR & (1 << SPIF)));      // Čekání na ukončení přenosu
       SetCSDAC;                          // Nastavení signálu CS do log. 1

}
```

Příloha D – Program pro ověření funkčnosti generování PWM signálu a měření otáček

```
#define F_CPU 16000000
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "spiLCD.c"
#define pocetLopatek 7

volatile double namereno = 0;      // Proměnna pro zjištění hodnoty čítače/časovače1
volatile unsigned char pocitadlo = 0;    // Proměnna pro zjištění počtu přetečení
                                         čítače/časovače1
unsigned char strida = 0;          // Proměnna pro nastavení střídy
unsigned int nastavStridu = 0;     // Proměnna pro nastavení střídy na velký ventilátor

ISR(TIMER3_OVF_vect)               // Obsluha přerušení od čítače/časovače3 - PWM
{
       OCR3C = nastavStridu;       // Do registru OCR3C zapsaní střídy
}

ISR (TIMER1_CAPT_vect)             // Obsluha přerušení od čítače/časovače1
{                                  // (vyvolaného změnou Input Capture)
       namereno = TCNT1;           // Do proměnné namereno se uloží hodnota TCNT1
                                   (počet kroku)
       pocitadlo = 0;              // Vynulování počítadla přetečení čítače/časovače1
       TCNT1 = 0;                  // Počítá se vždy od 0
}

ISR (TIMER1_OVF_vect)              // Obsluha přerušení od čítače/časovače1
{                                  // (vyvolaného při přetečení TCNT1)
       pocitadlo++;                // Inkrementace proměnné počítadlo
       TCNT1 = 0;                  // Počítá se od 0
}

ISR(INT1_vect)                     // Obsluha přerušení od tlačítka -
{
       cli();                      // Zakázáno přerušen/ - kvůli zákmitům
       strida = strida - 5;        // Proměnná strida se zmenší o 5 => střída se mění
                                   po kroku +-5
       sei();                      // Povolení přerušení
}

ISR(INT4_vect)                     // Obsluha přerušení od tlačítka +
{
       cli();                      // Zakázáno přerušení - kvůli zákmitům
       strida = strida + 5;        // Proměnná strida se zvětší o 5 => střída se mění
                                   po kroku +-5
       sei();                      // Povolení přerušení
}

int main(void)
{
       double frekvence = 0;               // Proměnná pro výpočet frekvence
       unsigned int ot = 0;                // Proměnná pro výpočet otáček

       // inicializace SPI pro LCD displej
       DDRB |= (1 << PB0) | (1 << PB1) | (1 << PB2);
                     // SPI - nastavení CS, MOSI, CSLK jako výstup
       SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPR1) | (1 << CPOL) | (1 << CPHA);
                     // Povoleni SP
```

```
	SetCSLCD;	// Nastaveni signalu CS pro LCD

	// inicializace čítače/časovače1 - měření otáček
	TCCR1B = (1 << ICES1) | (1 << CS10) | (1 << CS12);	// Nastavení hrany
detekce pro Input Capture, předdělička /1024 => jeden krok trvá 64us
	TIMSK = (1 << TICIE1) | (1 << TOIE1);			// Povolení přerušení od
události Input Capture a přetečení čítače/časovače1

	// inicializace čítače/časovače3 - PWM
	DDRE |= (1 << PE5);		// Nastavení PE5 jako PWM výstup
	TCCR3A = (1 << COM3C1) | (1 << WGM31) | (1 << WGM30);		// Nastavení
fastpwm s TOP v OCR3A
	TCCR3A &= ~(1 << COM3C0);
	TCCR3B = (1 << WGM32) | (1 << WGM33) | (1 << CS31);		// Nastavení
předděličky /8 => jeden krok trvá 0.5us
	ETIMSK = (1 << TOIE3);			// Povolení přerušení od čítače/časovače3
	OCR3A = 20000;				// Nastavení TOP (maxima) na 20000
	OCR3C = 0;				// Střída 0% => větrák se netočí

	// inicializace tlačítek
	PORTE |= (1 << PE4);			// Na PE4 1, kvůli přerušení
	PORTD |= (1 << PD1);			// Na PD1 1, kvůli přerušení
	EIMSK |= (1 << INT1) | (1 << INT4);// Povolení externích přerušení od INT1, INT4
	EICRB |= (1 << ISC41);			// INT4 reaguje na sestupnou hranu
	EICRA |= (1 << ISC11);			// INT1 reaguje na sestupnou hranu

// inicializace přepínače, pro přepnutí měření otáček velkého nebo malého ventilátoru
	DDRG &= ~(1 << PG0);		// Port PG0 nastav jako vstup
	DDRD |= (1 << PD5);		// Port PD5 nastav jako výstup


	sei();				// Globální povolení přerušení
	inicializaceLCD();		// Inicializace LCD displeje
	vypisText("Strida:");		// Funkce pro vypsání textu na LCD
	nastavKurzor(0, 10);		// Funkce nastaví kurzor na první řádek a 10 sloupec
	vypisText("%");			// Funkce pro vypsání textu na LCD
	nastavKurzor(1, 0);		// Funkce nastaví kurzor na druhy řádek a 0 sloupec
	vypisText("Otacky:");		// Funkce pro vypsání textu na LCD
	nastavKurzor(1, 13);		// Funkce nastaví kurzor na první řádek a 13 sloupec
	vypisText("o/m");		// Funkce pro vypsání textu na LCD

    while(1)
    {
	 if (PING & (1 << PG0))		// Když je přepínač v horní poloze
	 {
		PORTD |= (1 << PD5);	// Na port PD5 1, čidlo malý ventilátor
	 }
	 if (!(PING & (1 << PG0)))	// Když je přepínač v dolní poloze
	 {
		PORTD &= ~(1 << PD5);	// Na port PD5 0, čidlo velký ventilátor
	 }
	      nastavStridu = strida * 200;		// Výpočet střídy PWM signálu
	      if (strida >= 100)		// Omezení pro nastavení střídy - max 100%
	      {
		     nastavStridu = 20000;
		     strida = 100;
	      }
	      if (strida <= 99)			// Kvůli správnému výpisu na displej
	      {
		     nastavKurzor(0, 9);
		     vypisText(" ");
	      }
```

```
                nastavKurzor(0, 7); // Funkce nastaví kurzor na první řádek a 7 sloupec
                vypisCislo(strida);              // Funkce vypíše celé číslo
                frekvence = 1 / (((namereno / 15625) + (pocitadlo * 4.194)) *
                pocetLopatek);       // Výpočet frekvence
                frekvence = ceil(frekvence);      // Zaokrouhlení vypočítané frekvence na
                                                  celé číslo směrem nahoru
                ot = 60 * frekvence;              // Výpočet otáček ventilátoru [ot/min]
                if (ot <= 999)                    // Kvůli správnému výpisu na displej
                {
                      nastavKurzor(1, 11);
                      vypisText(" ");
                }
                nastavKurzor(1, 8); // Funkce nastaví kurzor na druhý řádek a 8 sloupec
                vypisCislo(ot);     // Funkce vypíše celé číslo
                _delay_ms(100);
    }
       return 0;

}
```

Příloha E – Funkce pro práci s LCD displejem

```
#define SetCSLCD     PORTB = (1 << PB0);        // Signál CS na 1
#define ClearCSLCD   PORTB &= ~(1 << PB0);      // Signál CS na 0
#define opc          0b01000000                 // opcode
#define expandVyst   0b00010101                 // Expandér výstup port B

// Funkce pro poslání dat
void spiPosliData(char data)
{
       SPDR=data;
       while(!(SPSR&(1<<SPIF)));  //Čekání dokud se neukončí přenos
}

// Funkce pro poslání dat do expandéru
void spiExpSend(char adr, char data )
{
       ClearCSLCD;          // CS do log. 0
       spiPosliData(opc);   // Na počátku komunikace s expandérem nutné vyslat opcode
       spiPosliData(adr);   // Adresa v expandéru na kterou chceme zapisovat
       spiPosliData(data);  // Odeslání dat
       SetCSLCD;            // CS do log. 1
}

// Funkce pro poslání instrukce pro LCD
void instDoLcd(char data)
{
       int i, j;
       char data2;

       for(j = 0; j < 2; j++)
       {
              data2 = (data & 0b11110000) + 0b00001000;       // Vynuluje dolní
                                                    pulbyte, nastaví 5. bit do 1 - E=1
              for(i = 0; i < 2; i++)
              {
                     spiExpSend(expandVyst, data2);           // Hodnotu data2 pošle na
                                                    výstupní port expandéru, vstup LCD
                     data2 &= 0b11110111;       // Opětovné vyslání dat s E=0,
              }
       data <<= 4;          // Posun o 4 - vyslání dolního pulbytu -viz. 4-bitový mód
       }
}

// Funkce pro poslání dat do LCD displeje
void dataDoLcd(char data)
{
       int i, j;
       char data2;

       for(j = 0; j < 2; j++)
       {
              data2 = (data & 0b11110000) + 0b00001100;       // Vynuluje dolní
                                                          pulbyte, nastaví E=1, RS=1
              for(i = 0; i < 2; i++)
              {
                     spiExpSend(expandVyst, data2);    // expandVyst = adresa
                                                          výstupního registru IO
                     data2 &= 0b11110111;       // Opětovné vyslání dat s E=0,
              }
       data <<= 4;                        // posun o 4 - vyslání dolního pulbytu
       }
}
```

```
// Funkce pro inicializaci LCD displeje a expandéru
void inicializaceLCD(void)
{
	spiExpSend(0b00000001, 0b00000000);	//adr IODIRA, nastavení portu B
						expandéru jako výstup
	_delay_ms(30);
	spiExpSend(expandVyst, 0b00101000);	//Nastavení 4-bitového režimu
						komunikace, dvouřádkový režim
	spiExpSend(expandVyst, 0b00100000);
	_delay_us(40);

	instDoLcd(0b00101000);
	_delay_us(40);

	// Displej on, Kurzor off
	instDoLcd(0b00001100);
	_delay_us(40);

}

// Funkce pro smazání LCD displeje
void smazLCD(void)
{
	instDoLcd(0b00000001);	// Odeslání instrukce pro smazání displeje
	_delay_us(2);
	instDoLcd(0b00000010);	// Odeslání instrukce pro nastavení kurzoru na
				pozici 0, 0 displeje
	_delay_us(2);
}

// Funkce pro nastavení kurzoru LCD displeje
void nastavKurzor(char radek, char sloupec)
{
	char instrukce;
	switch (radek)
	 {
		case 0:
		instrukce = (0b10000000) | sloupec;
		break;
		case 1:
		instrukce = (0b10000000) | (0x40 + sloupec);
		break;
	}
	instDoLcd(instrukce);
	_delay_us(40);
}

// Funkce pro vypsání textu na LCD displej
void vypisText(char* retezec)
{

	while (*retezec != '\0') //Dokud nenarazí na konec(\0) tak posílá znaky
	{
		dataDoLcd(*retezec);
		_delay_us(40);
		retezec += 1;		// Posouvá pointer v poli charu
	}
```

```
// Funkce pro vypsání celého čísla na LCD displeje
void vypisCislo(int cislo)
{
	char str[4];
	sprintf(str, "%d", cislo);
	vypisText(str);
}

// Funkce pro vypsání desetinného čísla na LCD displeje
void vypisRealneCislo(double cislo)
{
	int cele, pred, za;
	cele = cislo *10;
	pred = cele / 10;
	za = cele % 10;
	char str[4];
	sprintf(str, "%d", pred);
	vypisText(str);
	vypisText(".");
	dataDoLcd(0x30+za);
}
```