UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Mikroprocesorová řídicí jednotka laboratorního dvourotorového systému

Pavel Jiránek

Diplomová práce
2013

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

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel Jiránek**
Osobní číslo:	**I11357**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a řídicí technologie**
Název tématu:	**Mikroprocesorová řídicí jednotka laboratorního dvourotorového systému**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

Cílem práce bude návrh hardware a software řídicí jednotky laboratorního dvourotorového systému. Řídicí jednotka bude umožňovat kompletní ovládání systému a úpravu měřených signálů pro zpracování v nadřazeném systému - PC. Součástí softwarového řešení bude návrh vybraného typu regulátoru pro řízení systému.
Teoretická část
V teoretické části práce bude uveden stručný popis řízeného systému a možnosti jeho řízení. Dále bude proveden popis návrhu vhodného typu regulátoru pro řízení systému.
Implementační část
V implementační části práce bude proveden návrh řídicí jednotky a její realizace. Dále bude navržen příslušný regulátor pro řízení soustavy. Experimentální část práce bude obsahovat naměřené statické a dynamické charakteristiky systému a příslušné regulační pochody s realizovaným regulátorem.

Rozsah grafických prací: **20 listů**

Rozsah pracovní zprávy: **90 - 100 stran**

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

Seznam odborné literatury:

Šulc, B. - Vítečková, M.: Teorie a praxe návrhu regulačních obvodů. Vydavatelství ČVUT Praha, 2004.
Matoušek David: Práce s mikrokontroléry ATMEL AVR - 3.díl - edice uP a praxe 2. vydání, BEN-technická literatura, 2006, ISBN 80-7300-209-4.
Plíva Zdeněk: Eagle Prakticky, BEN-technická literatura, 2010, ISBN 978-80-7300-252-7.
Záhlava Vít: Návrh a konstrukce DPS, BEN-technická literatura, 2010, ISBN 978-80-7300-266-4.
www.atmel.com
www.elcad.cz

Vedoucí diplomové práce: **Ing. Libor Havlíček, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

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. 9. 2013 Pavel Jiránek

Poděkování

Rád bych tímto poděkoval vedoucímu diplomové práce, Ing. Liboru Havlíčkovi, Ph.D., za trpělivost, cenné rady a připomínky, kterých se mi při psaní této práce dostalo. Dále Ing. Jaroslavu Pázlerovi z firmy MEDTEC za výrobu hliníkové krabičky.

Anotace

Diplomová práce se zabývá návrhem hardware i software mikroprocesorové řídící jednotky laboratorního dvourotorového systému. Řídící jednotka umožňuje kompletní ovládání řízeného systému, zpracování signálů z optických snímačů polohy a tachogenerátorů pro použití v samotné jednotce, ale i pro použití s nadřazeným systémem - PC. Zdrojové kódy pro mikroprocesory jsou napsané v programovacím jazyce C ve vývojovém prostředí AVR Studio 4.

Klíčová slova

Jazyk C, PSD, regulátor, mikroprocesor, ATmega32, ATmega8

Title

Microprocessor control unit for two-rotor laboratory system.

Annotation

This master’s thesis deals with the design of hardware and software microprocessor control unit for two-rotor laboratory system. The control unit allows complete control of the controlled system, processing signals from position optical sensors and tachogenerators for use with the control unit, or for use in superior system - PC. Source codes for microprocessors are written in the programming language C. As a development environment is used AVR Studio 4.

Keywords

C, PSD, regulator, microprocessor, ATmega32, ATmega8

Obsah

Seznam zkratek

IRC	Inkrementální rotační spínač
PWM	Pulse-width modulation
URO	Uzavřený regulační obvod
DPS	Deska plošného spoje
AD	Analogově digitální
DA	Digitálně analogový
SS	Stejnosměrné
ST	Střídavé
LCD	Liquid crystal display
USART	Universal Synchronous/ Asynchronous Receiver and Transmitter
SPI	Seriál Peripheral Interface
I^2C	Inter-Integrated Circuit

Seznam obrázků

Seznam tabulek

Úvod

Již dlouho mě lákalo zkusit si navrhnout a vyrobit nějaké elektronické zařízení řízené mikroprocesorem, ale stále se mi nedostávalo času pro vlastní projekt. To byl jeden z hlavních důvodů, proč jsem využil nabídky pana doktora Havlíčka, který nám nabízel několik témat diplomových prací. Z těchto mě nejvíce zaujalo téma, které se zabývá návrhem a realizací mikroprocesorové řídící jednotky pro laboratorní dvourotorový systém. Obojí bude sloužit ve školních laboratořích pro studijní účely.

Dokumentace diplomové práce je rozdělena do několika částí. První část je věnována popisu ovládaného systému jak z praktického, tak teoretického hlediska, rozboru možností ovládání a popisu regulátoru PSD včetně metod návrhu. Ve druhé části je uveden popis jednotlivých částí řídící jednotky a třetí část je věnována naměřeným charakteristikám.

Vlastní řídící jednotka je složena z několika plošných spojů spojených mezi sebou distančními sloupky a celek je uzavřen v hliníkové krabičce vyrobené na míru. Tato krabička slouží zároveň jako chladič výkonových členů obou motorů. Zhotovení výrobních podkladů k výrobě krabičky bylo použito programu ProfiCAD. Pro pohodlnější ovládání je řídící jednotka vybavena čtyřřádkovým LCD displejem, dvěma tlačítky pro listování menu a dvěma rotačními enkodéry pro inkrementování, nebo dekrementování hodnot proměnných. Řídící jednotka samozřejmě umí pracovat samostatně, a umožňuje kompletní ovládání řízeného systému. Z důvodů použití celkem 4 rotačních enkodérů a tedy potřeby použití 4 externích přerušení, bylo zapotřebí zařadit do zapojení k mikroprocesoru ATmega32 ještě druhý mikroprocesor ATmega8. Prvně zmíněný mikroprocesor je srdcem celé jednotky. Zajišťuje zpracování signálů z tachogenerátorů, dat přijímaných z druhého mikrokontroléru, ovládání LCD displeje, vyhodnocování stavů ovládacích tlačítek a enkodérů a v neposlední řadě také generování dvou signálů PWM jako řídících signálů pro výkonové členy obou motorů. ATmega8 zpracovává signály z IRC snímačů natočení a naklonění. Tyto informace dále předává jednak druhému mikrokontroléru pomocí sériového komunikačního rozhraní USART a jednak v závislosti na velikosti úhlů generuje dva signály PWM, které jsou dále dolnofrekvenční propustí převedeny na úroveň stejnosměrného napětí a přivedeny na konektory v zadní stěně krabičky. Tento konektor slouží pro spojení s PC a dále obsahuje také vodiče nesoucí informace o otáčkách motorů a dva vodiče pro ovládání výkonových členů motorů z PC.

Řídící jednotka je univerzální a po vhodných úpravách použitelná i pro zcela odlišné systémy nežli je použitý v této práci. V tomto případě se jedná o dvourotorový aerodynamický systém s vlastnostmi připomínající skutečnou helikoptéru. Samozřejmě je mezi obojím celá řada odlišností. Například střed otáčení, který u helikoptéry je pod hlavním rotorem, kdežto u tohoto systému mezi rotory. Více rozdílů je uvedeno dále v práci.

Pro naprogramování obou mikroprocesorů jsem si mohl vybrat mezi programovacími jazyky Assembler a C. Zvolil jsem si jazyk C, neboť je mi příjemnější pro práci a mám

s ním více zkušeností. Vývoj software obou mikrokontrolérů a následné jejich programování je prováděno ve vývojovém prostředí AVRStudio4.

1 Popis řízeného systému

1.1 Konstrukce modelu

Jedná se o dvourotorový, vícerozměrový, aerodynamický systém, který je ve svém chování do jisté míry podobný chování skutečné helikoptéry. Hlavní rozdíly laboratorního systému a helikoptéry jsou střed otáčení, který se u našeho systému nachází mezi rotory, kdežto u helikoptéry u hlavního rotoru, vztlaková síla, ta je dána rychlostí otáčení hlavního rotoru, u skutečné helikoptéry náběžným úhlem rotorových listů a náklon ve vertikálním směru, ten je dán také rychlostí hlavního rotoru, oproti helikoptéře, kde je dán náklonem hlavního rotoru.

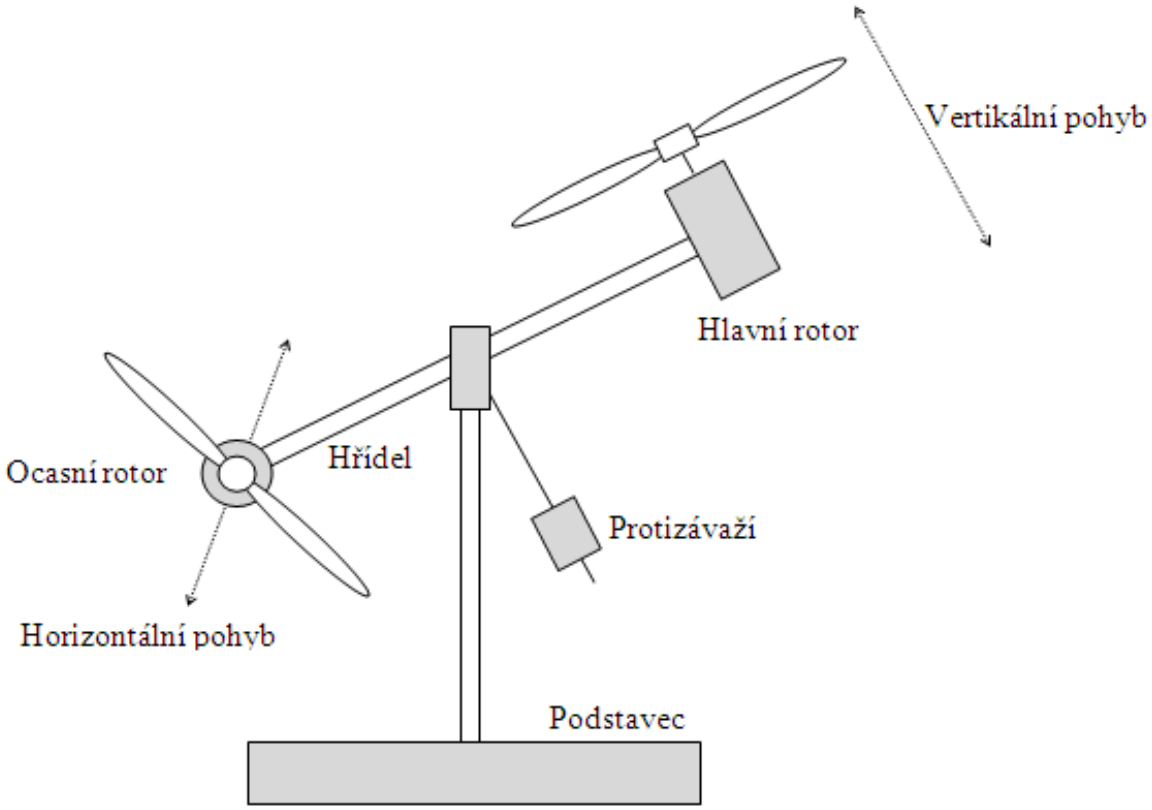
Bylo by vhodné zmínit i k čemu je takový model helikoptéry vlastně dobrý. Na tomto modelu můžeme navrhovat a testovat různé algoritmy řízení. Dále se na něm dá do jisté míry testovat i chování skutečného vrtulníku, například chování mezi překážkami a podobně.

Obrázek 1 - Laboratorní model

Jak je vidět z obrázku (Obrázek 1), laboratorní model se skládá z velkého počtu prvků. Důležitým prvkem je hřídel, která má na obou koncích uchycené stejnosměrné motory s vrtulemi. Tato hřídel je upevněna tak, že dovoluje jak vertikální tak horizontální pohyb po omezené kulové ploše. Motor, představující hlavní rotor vrtulníku, je uložen vertikálně a druhý motor, představující ocasní rotor, je v horizontální poloze. Vrtule vertikálního motoru je poháněna přes převodovku s převodovým poměrem 1,5:1. Důvodem použití převodovky je zvýšení tažné síly. U horizontálního motoru je vrtule poháněna přímo bez převodovky.

Úhly naklonění a natočení hřídele jsou dány úhlovou rychlostí elektromotorů a zjišťují se pomocí inkrementálních rotačních enkodérů. Úhlové rychlosti elektromotorů se měří pomocí tachogenerátorů, které jsou již součástí elektromotorů.

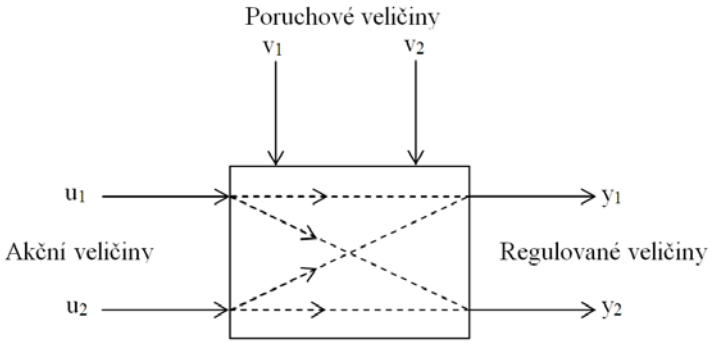
Celý model je připevněn k dostatečně dimenzovanému podstavci. Z podstavce vystupuje trojce nosných tyčí z konstrukční oceli (rozměr 12mm x 450mm). V horní části těchto tyčí je upevněno uložení ložiska hřídele umožňující horizontální natáčení. V obrázku (Obrázek 2) jsou schematicky naznačeny možné pohyby soustavy. Vertikální pohyb směrem vzhůru je zajištěn tažnou silou hlavního rotoru, pohyb směrem dolů silou gravitační. Horizontální natáčení hřídele je zajištěno ocasním rotorem, u kterého je možná změna smyslu otáčení pro otáčení jak v levém, tak pravém směru, navíc tento rotor má za úkol i kompenzaci odstředivé síly hlavního rotoru. Vertikální pohyb i stranové natočení je možné až po konstrukční omezovače pohybu - mechanické zarážky. V tomto pracovním režimu je možné systém natočit ±45° ve vertikálním směru a 0-225° v horizontálním směru. Součástí zarážek jsou optické závory, které slouží pro nastavení rozsahů polohy. Pokud dojde k dosažení závory, nastaví se údaj v čítači na 0°.



Obrázek 2 - Laboratorní model - popis

1.2 Vícerozměrové soustavy

Vícerozměrové soustavy jsou soustavy s větším počtem vstupů a výstupů, které jsou mezi sebou nějakým způsobem provázány a mohou se tedy navzájem ovlivňovat. Provázání a vzájemné ovlivňování se nazývá interakce - v obrázku (Obrázek 3) jsou uvedeny čárkovaně.



Obrázek 3 - Vícerozměrová soustava

Vstupem do soustavy jsou akční spolu s poruchovými veličinami a výstupem regulované veličiny [2].

- *Akční veličiny* $u_i(i=1,2…n)$ - výstupní veličiny regulátoru a zároveň vstupní veličiny regulované soustavy. Regulace se provádí změnou akčních veličin.
- *Poruchové veličiny* $v_i(i=1,2…n)$ - obvykle se předpokládá, že jsou náhodné.
- *Regulované veličiny* $y_i(i=1,2…n)$- Výstupní veličiny soustavy. Jejich hodnoty jsou regulací udržovány na žádaných hodnotách.

Maticový zápis[1]:

- vektory akčních, poruchových a regulovaných veličin

$$\bar{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \bar{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \bar{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

- přenosová matice akčních veličin

$$G = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}$$

- přenosová matice poruchových veličin

$$G_V = \begin{bmatrix} G_{V11}(s) & G_{V12}(s) \\ G_{V21}(s) & G_{V22}(s) \end{bmatrix}$$

- Rovnice obrazu výstupní veličiny

$$\bar{y} = G * \bar{u} + G_V * \bar{v}$$

[1] Operátor s pro spojité přenosy, nebo z pro diskrétní přenosy

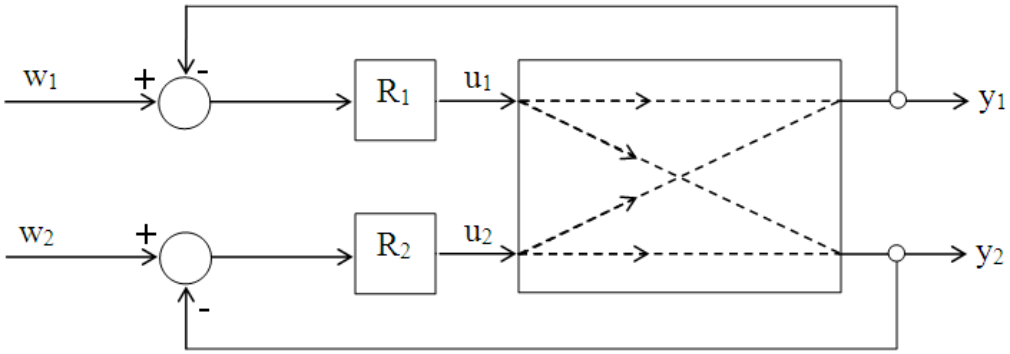
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} * \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} G_{V11}(s) & G_{V12}(s) \\ G_{V21}(s) & G_{V22}(s) \end{bmatrix} * \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

V tomto případě jde o systém se dvěma vstupy a dvěma výstupy. Vstupy jsou napájecí napětí obou elektromotorů a výstupy úhlové rychlosti motorů spolu s úhly náklonu hřídele. Akčními zásahy jsou změny rychlosti otáčení motorů.

1.2.1 Možnosti řízení

Pro řízení vícerozměrových soustav existují tři základní možnosti:

a) Decentralizované řízení
   - používá se regulátor s diagonální přenosovou maticí
   - jednorozměrové regulátory berou interakce v úvahu (všechny regulační pochody se zpomalí natolik, aby všechny jednorozměrové systémy byly stabilní i za cenu zhoršení kvality regulace)

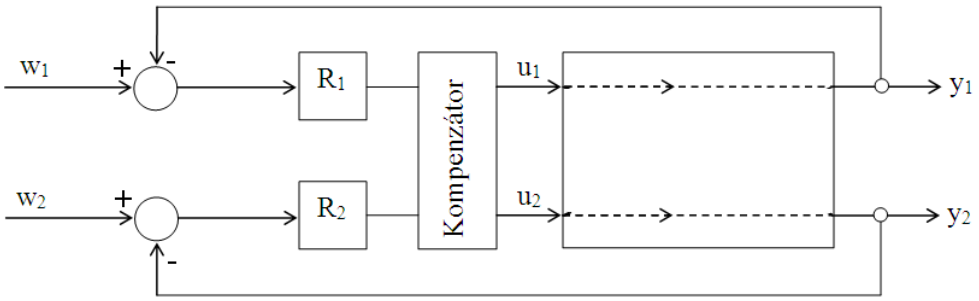


Obrázek 4 - Decentralizované řízení

(w_1,w_2 - žádané veličiny; R_1, R_2 - regulátory; u_1, u_2 - akční veličiny; y_1, y_2 - regulované veličiny)

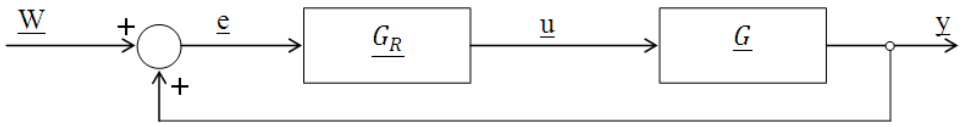
b) Autonomní řízení
   - eliminace nežádoucích křížových vazeb - interakcí
   - regulátor složen ze dvou částí
     i. kompenzátor - navržen tak, aby přenosová matice kompenzátoru a soustavy byla diagonální[2]. Tím dojde k rozdělení na jednorozměrné soustavy.
     ii. soubor jednorozměrových regulátorů - řídí výše zmíněné jednorozměrové soustavy

[2] Diagonální matice - čtvercová matice, která má nenulové prvky pouze na hlavní diagonále



Obrázek 5 - Autonomní řízení

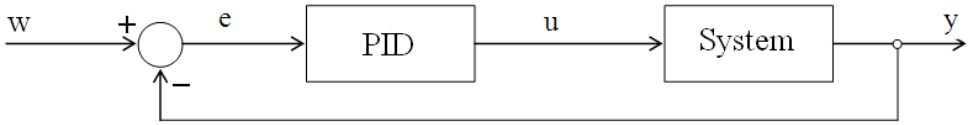
c) Vícerozměrové regulátory
   - vzniknou rozšířením jednorozměrového regulátoru
   - pro popis přenosů se využívají matice
   - přechod z jednorozměrového na vícerozměrový regulátor je složité



Obrázek 6 - Regulační obvod[3]

1.3 Regulátor

Pro řízení systému byl zvolen regulátor PSD. Jde o diskrétní verzi spojitého regulátoru PID. Tento regulátor je tvořen třemi částmi: proporcionální, integrační a derivační. U PSD je integrační část nahrazena částí sumační.



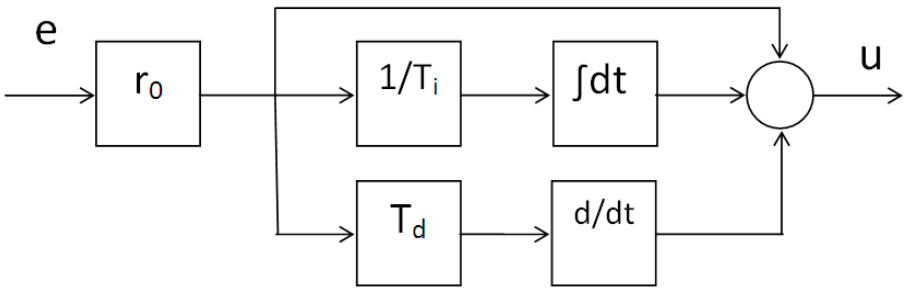
Obrázek 7 - URO s PID regulátorem

Na obrázku výše (Obrázek 7) je vidět uzavřený regulační obvod s PID regulátorem. Regulační odchylka *e* je tvořena rozdílem žádané veličiny *w* a regulované veličiny *y*.

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

Vstupem PID regulátoru je tedy regulační odchylka *e*. Regulátor na základě *e*, pomocí akční veličiny *u*, mění vlastnosti systému a to do té doby, než je regulační odchylka *e* v předem stanovené normě (malá nebo nulová).

[3] Podtržení značí, že jde o matice



Obrázek 8 - PID - blokové schéma

Základní rovnice jednoduchého PID regulátoru:

$$u(t) = r_0 * \left[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right]$$

r_0 - proporcionální složka
T_i - integrační časová konstanta
T_d - derivační časová konstanta

Základní rovnice ekvivalentní diskrétní formy, regulátor PSD:

$$u(k) = r_0 * \left\{ e(k) + \frac{T}{T_i} \left[\frac{e(0)}{2} + \sum_{j=1}^{k-1} e(j) + \frac{e(k)}{2} \right] + \frac{T_d}{T} * [e(k) - e(k-1)] \right\}$$

Přepis do přírůstkového tvaru:

$$u(k) = u(k-1) + q_0 * e(k) + q_1 * e(k-1) + q_2 * e(k-2)$$

$$e(k) = w(k) - y(k)$$
$$e(k-1) = w(k-1) - y(k-1)$$
$$e(k-2) = w(k-2) - y(k-2)$$

Přepis do polohového tvaru - z důvodu zlepšení vlastností, především lepší chování při změnách *w*:

$$u(k) = u(k-1) + (q_0 + q_1 + q_2) * w(k) + q_0 * y(k) + q_1 * y(k-1) + q_2 * y(k-2)$$

Kde k značí pořadí diskrétního času, q_0, q_1, q_2 jsou parametry diferenční rovnice.

Při použití polohového tvaru je nutné přepočítat parametry spojitého regulátoru na parametry diferenční rovnice dle následujících vztahů:

$$q_0 = r_0 * \left(1 + \frac{T}{2 * T_i} + \frac{T_d}{T}\right)$$

$$q_1 = -r_0 * \left(1 - \frac{T}{2 * T_i} + \frac{2 * T_d}{T}\right)$$

$$q_2 = r_0 * \frac{T_d}{T}$$

Kde T je interval vzorkování, který musí být dostatečně malý, aby bylo chování ekvivalentní s PID regulátorem.

1.3.1 Proporcionální člen

P-regulátor, je prostý zesilovač, kde regulační odchylka je přímo úměrná akční veličině. Tedy, odečte se regulovaná veličina od žádané a jejich rozdíl se vynásobí konstantou, výsledek je například výkon, který se musí dodat do soustavy. Pokud je regulovaná veličina o hodně menší než žádaná, je vliv velký. Čím více je regulovaná veličina bližší žádané, tím je vliv nižší. Až jsou obě stejné, tak je vliv nulový.

Základní rovnice P regulátoru:

$$u(t) = r_0 * e(t)$$

1.3.2 Integrační člen

I regulátor - regulátor, kde je akční veličina přímo úměrná integrálu regulační odchylky (r_i je zesílení regulátoru). I regulátor vynásobí regulační odchylku konstantou a přičte ji ke své složce. Jestliže je regulovaná veličina nižší než žádaná, bude se integrační složka zvyšovat. V případě, že je regulovaná veličina vyšší než žádaná, bude se snižovat. Čím je *e* vyšší, tím rychleji se bude integrační složka měnit. Přesto použití pouze I regulátoru má pomalou odezvu a často dochází k oscilacím kolem požadované hodnoty.

Základní rovnice I regulátoru:

$$u(t) = \frac{1}{T_i}\int_0^t e(t)dt$$

Integrační člen se obvykle kombinuje s proporcionálním členem. Jejich kombinací vznikne PI regulátor.

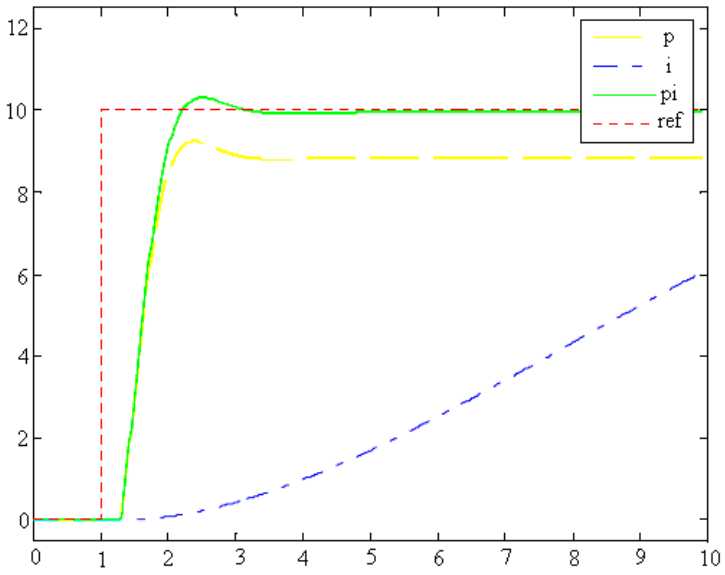
1.3.3 Derivační člen

Při použití pouze derivačního členu vznikne D regulátor. Zde je akční veličina přímo úměrná derivaci regulační odchylky. Derivační regulátor vynásobí konstantou rychlost změny odchylky. V případě, že regulovaná veličina klesá, regulátor zvyšuje výkon (například). Čím je pokles rychlejší, tím vyšší je dodávaný výkon. A opačně, když regulovaná veličina stoupá, regulátor bude výkon snižovat.

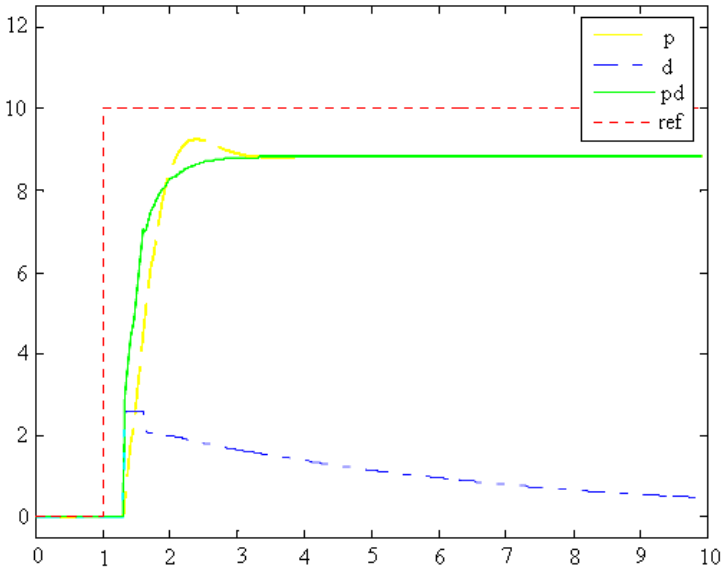
Obvykle se derivační člen používá společně s proporcionálním (PD regulátor) a integračním (PID regulátor).

Základní rovnice PD regulátoru:

$$u(t) = r_0 * \left[e(t) + T_d \frac{de(t)}{dt}\right]$$



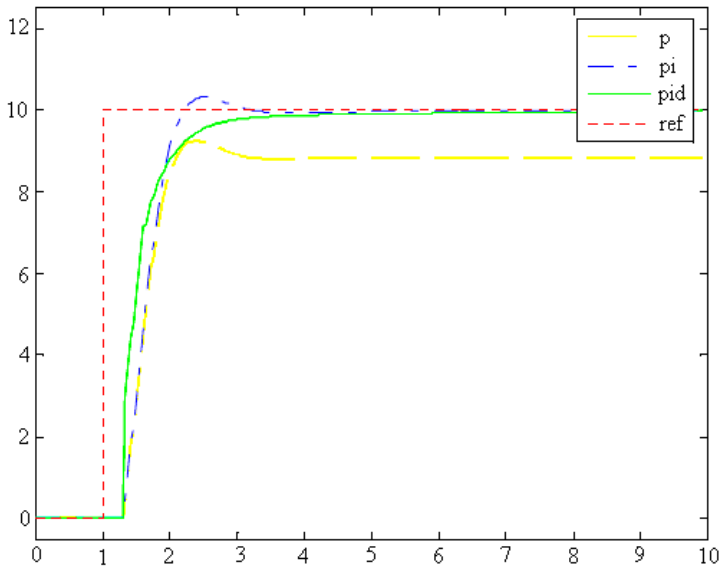
Obrázek 9 - Regulátor P, I, PI [7]



Obrázek 10 - Regulátor P, D, PD [7]

1.3.4 Kombinace P, I, D

Nejlepší je ovšem kombinace P, I a D členu, tedy PID regulátor. Porovnání odezev na jednotkový skok mezi regulátory typu P, PI a PID je vidět na obrázku níže (Obrázek 11)



Obrázek 11 - Regulátor P, PI, PID [7]

Pro nastavování parametrů regulátorů PID existuje několik metod. Mezi experimentální metody přímé patří *metoda pokus/omyl, Ziegler-Nichols* a metoda *relé ve zpětné vazbě.*

1.3.5 Metoda *Ziegler-Nichols*

Též nazývána *Metoda kritického zesílení regulátoru*. Jde o jednoduchou a často využívanou metodu. Je zde zapotřebí vyřadit integrační a derivační složku, tedy přivést regulační obvod na mez stability. I a D složka se vyřadí takto:

$$T_I = \infty;\ T_D = 0$$

Následně se zvyšuje zesílení proporcionální složky r_0 dokud regulační obvod nekmitá netlumenými kmity. V tomto okamžiku je obvod na mezi stability. Odečtou se kritické hodnoty zesílení r_{0k} a perioda netlumených kmitů, neboli periodu kritického zesílení T_k.

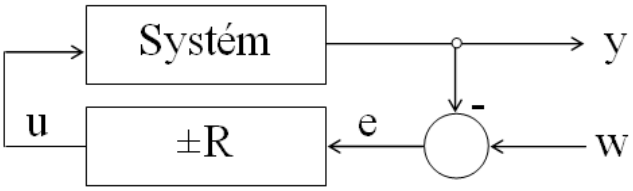
Tyto kritické hodnoty se dosadí do vzorců z tabulky (Tabulka 1) pro stanovení parametrů regulátoru:

Typ regulátoru	r_0	T_I	T_D
P	$0{,}5*r_{0k}$		
PD	$0{,}65*r_{0k}$		$0{,}12*T_k$
PI	$0{,}45*r_{0k}$	$0{,}85*T_k$	
PID	$0{,}65*r_{0k}$	$0{,}5*T_k$	$0{,}125*T_k$

Tabulka 1 - Vzorce pro návrh parametrů regulátoru

1.3.6 Metoda relé ve zpětné vazbě

Další z možností zjištění parametrů regulátoru je metoda, kdy se do zpětné vazby zapojí relé:

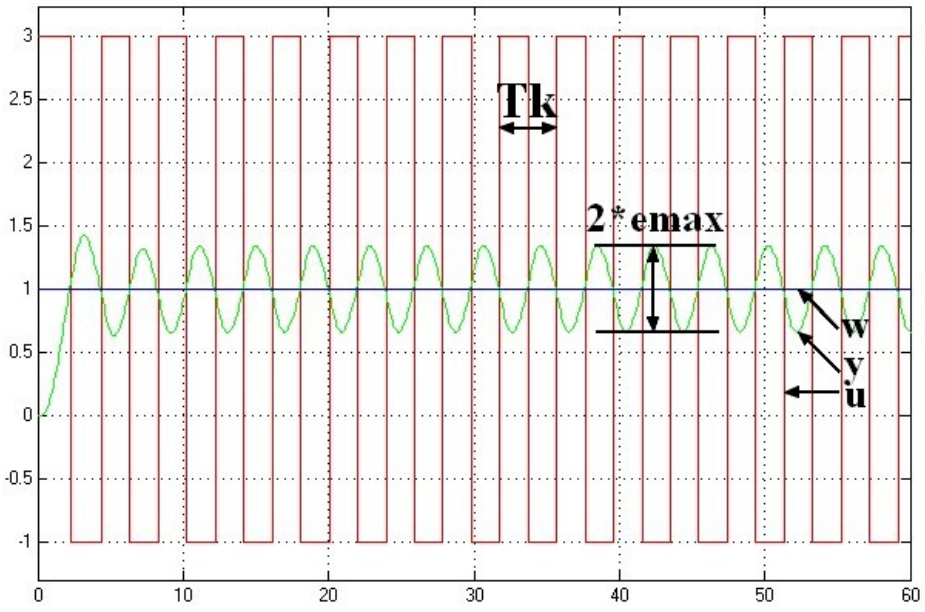


Obrázek 12 - Relé ve zpětné vazbě

Relé je zde z toho důvodu, že přepíná mezi úrovněmi (±R) žádané veličiny podle regulační odchylky e. Princip je vidět z obrázku (Obrázek 13) níže, tedy vždy, když regulační odchylka projde nulou, relé přepne. Akční veličina je díky tomu ve tvaru obdélníkového signálu, který po projití soustavou dostává tvaru sinusoidy. Ve stejném obrázku jsou vyznačeny potřebné veličiny pro výpočet. T_k - kritická perioda, $2*e_{max}$ - amplituda ustálených kmitů (kritickou periodu lze zjistit buď ze vstupního, nebo výstupního průběhu). Následně je nutné spočítat hodnotu kritického zesílení dle vztahu:

$$r_0 = \frac{4 * R}{\pi * e_{max}}$$

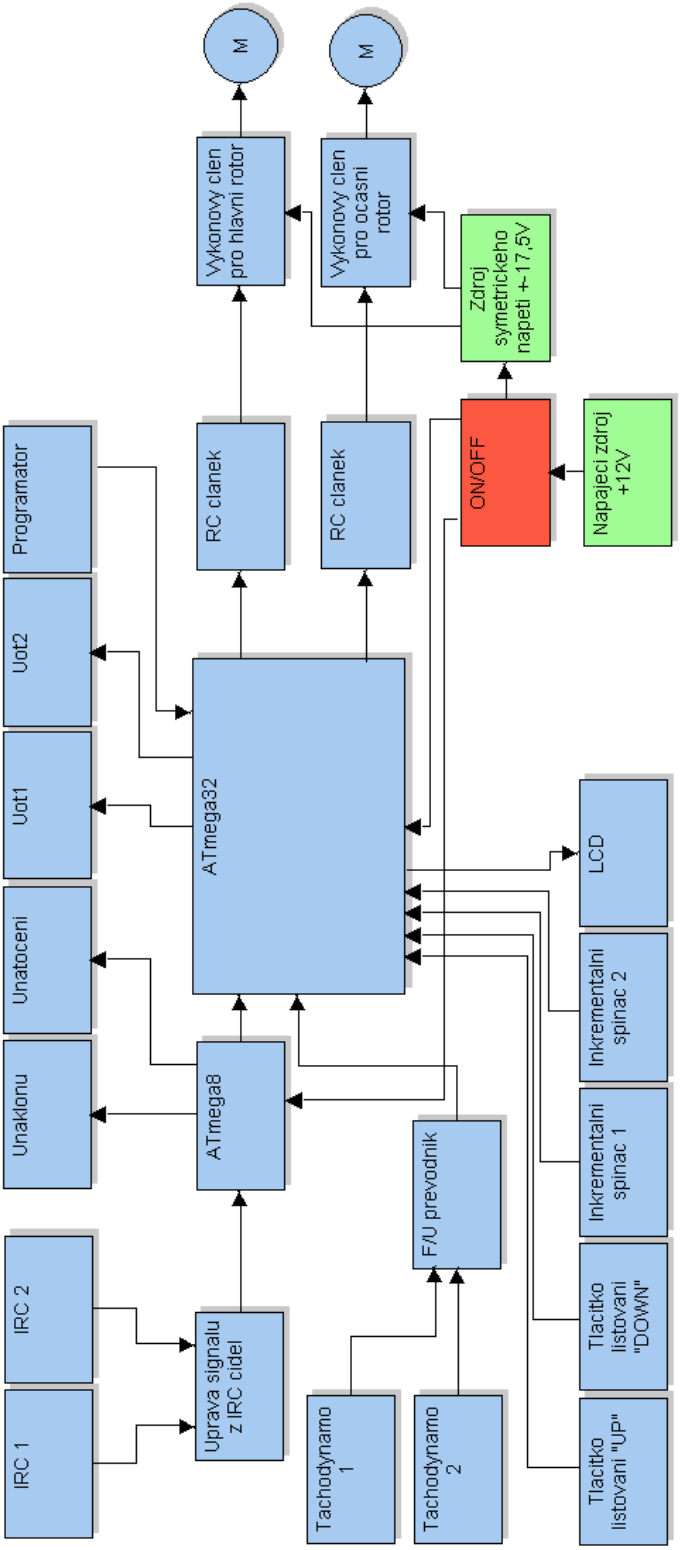
Kde R vyjadřuje amplitudu relé a e_{max} amplitudu výstupního signálu.



Obrázek 13 - Zjištění kritických hodnot

2 Návrh a realizace řídící jednotky

2.1 Blokové schéma a popis



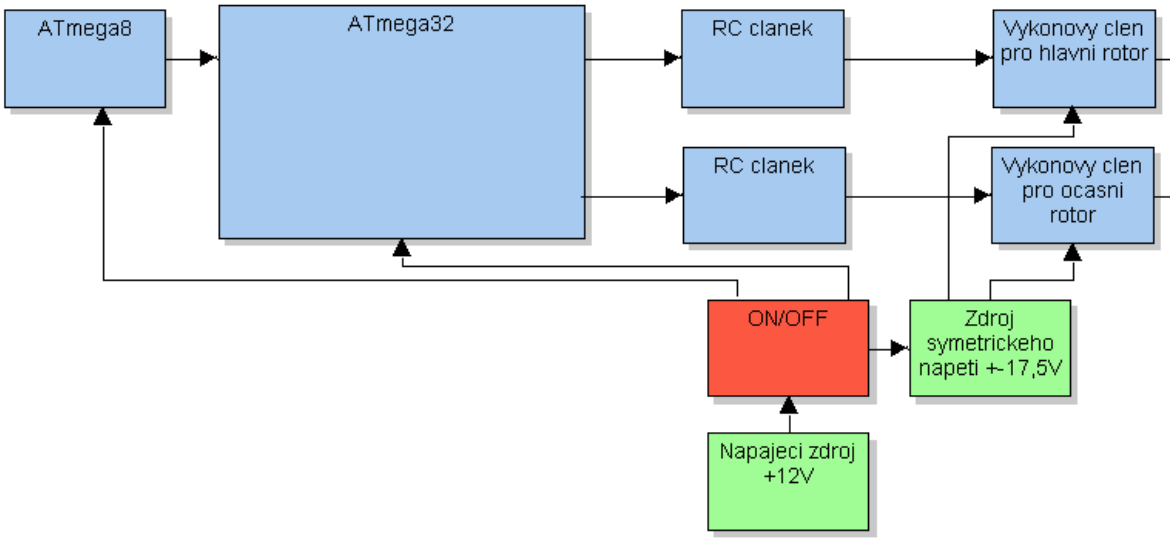
Obrázek 14 - Blokové schéma řídící jednotky

Celé zapojení řídící jednotky je rozděleno do jednodušších bloků a realizováno na několika plošných spojích popsaných níže.

Z blokového schématu je patrný princip celého zapojení. Vpravo dole je uveden zdroj napájení, který napájí moduly ATmega8, ATmega32 a zdroj symetrického napětí přes kolébkový spínač. ATmega8 obstarává informace o naklonění/ natočení a tyto informace předává modulu ATmega32, která je zobrazuje na displej. Tyto informace jsou také přivedeny po vhodné úpravě na výstupní konektor pro spojení s počítačem. ATmega32 dále zpracovává signály z převodníků frekvence/ napětí vyjadřující otáčky motorů (které jsou také přivedeny na konektor pro spojení s počítačem), obstarává obsluhu ovládacích prvků (tlačítka, rotační enkodéry), řídí výpis na LCD displej, a také generuje signál PWM pro řízení motorů. Tento signál je převeden RC článkem na úroveň stejnosměrného napětí a tímto jsou ovládány výkonové členy, které řídí otáčky motorů.

2.2 Napájecí zdroj

O napájení jednotky se stará adaptér podobný těm co se využívají například u notebooků. Jak je vidět z obrázku (Obrázek 15) tento adaptér napájí blok s mikropočítačem ATmega8, ATmega32 a zdroj symetrického napětí k napájení výkonových členů pro motory.



Obrázek 15 - Napájení řídící jednotky

Parametry napájecího adaptéru si můžeme prohlédnout v tabulce níže (Tabulka 2).

Výrobce	SUNNY Computer technology co., LTD.
Model	SYS1097-5012
Vstup	100-240V AC 1.6A MAX, 50-60Hz, 100-200VA
Výstup	+12V DC 4.17A, 50W MAX

Tabulka 2 - Parametry napájecího adaptéru

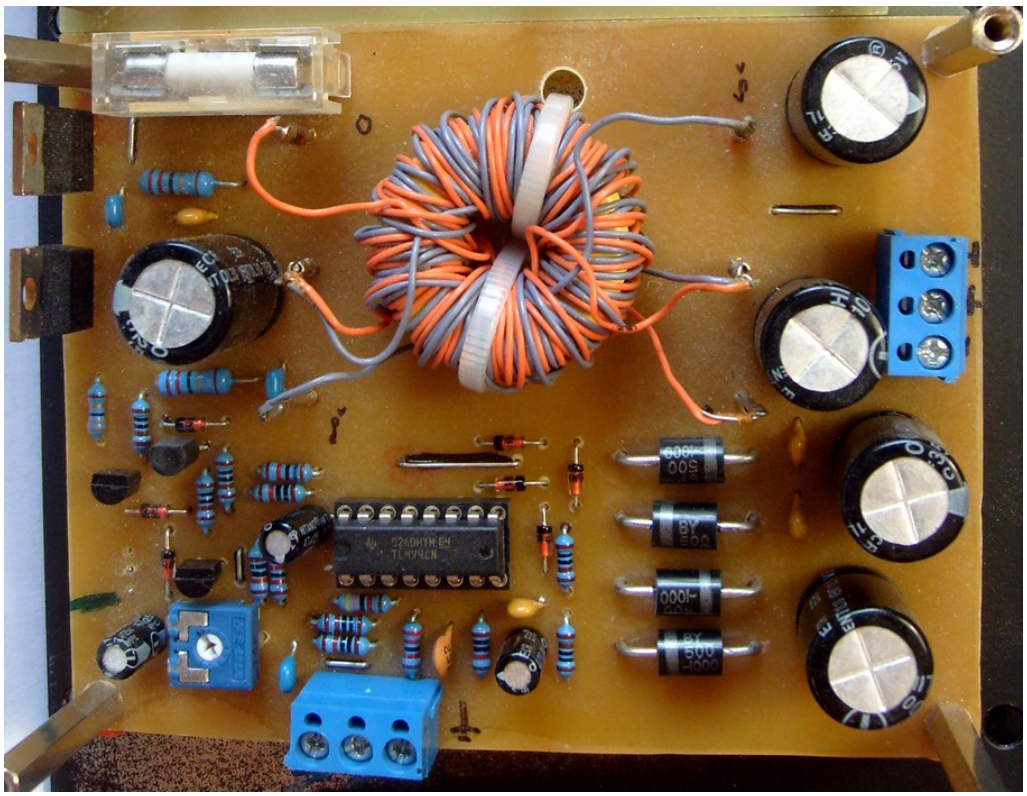
Zdroj symetrického napětí ±17.5V je vlastně DC-DC převodník. Vstupuje do něj stejnosměrné napětí +12V a vystupuje také stejnosměrné napětí, ale vyšší a symetrické o

velikosti ±17,5V. Jeho schéma je na obrázku níže (Obrázek 6). Jedná se o zapojení uvedené v literatuře [1]. Oproti původnímu zapojení je změněn počet závitů na transformátoru z důvodu potřeby jiného výstupního napětí. Přibližný počet závitů byl vypočten dle rovnice ideálního transformátoru:

$$p = \frac{U_1}{U_2} = \frac{N_1}{N_2}$$

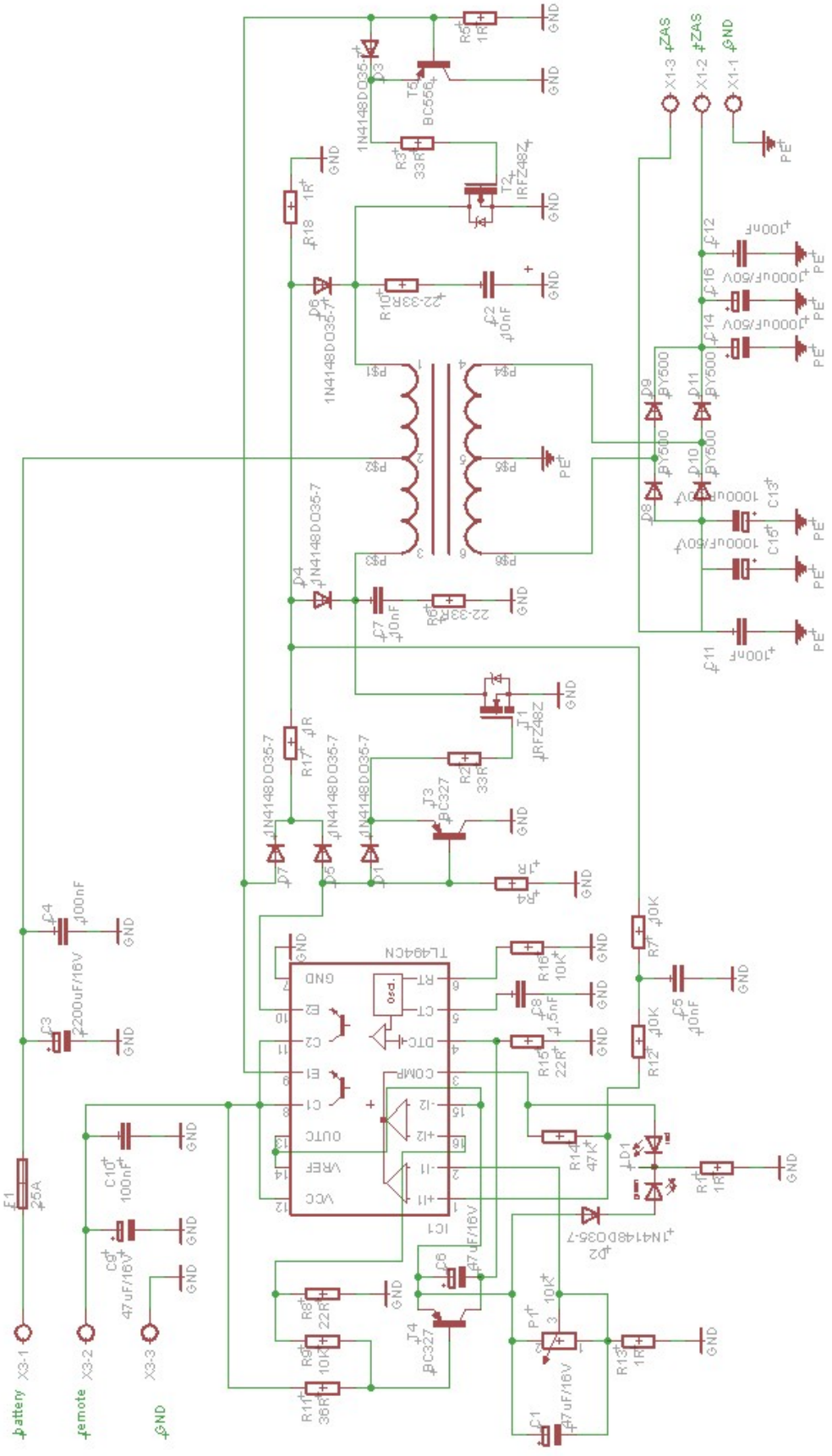
Počet závitů byl poté experimentálně upraven na námi požadované výstupní napětí.

Nejdůležitější částí převodníku je výše zmíněný transformátor, který pro svoji činnost potřebuje střídavé napětí. To se vytváří v obvodu TL494 s připojenými výkonovými MOSFET tranzistory. Za sekundárním vinutím transformátoru je napětí usměrněno můstkovým usměrňovačem a dále filtrováno kondenzátory.[4]



Obrázek 16 - Zdroj symetrického napětí

[4] POWIRSKI, Ireneusz. *Przetwornica do CAR-AUDIO*



Obrázek 17- Schéma zapojení zdroje symetrického napětí

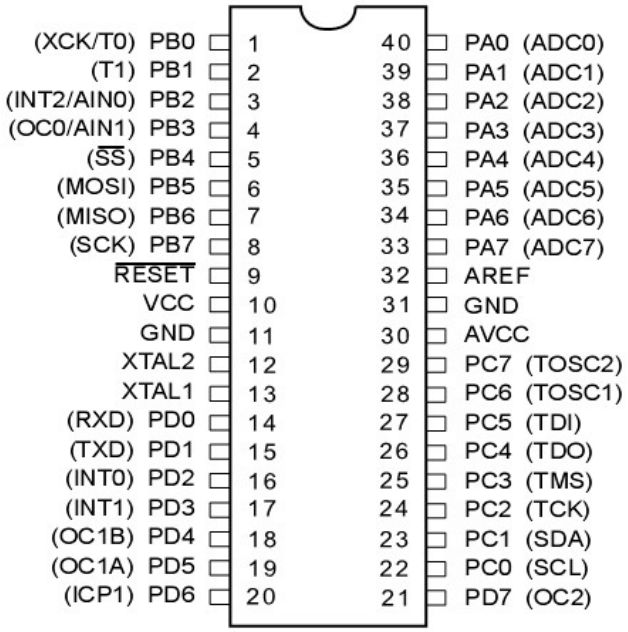
2.3 Mikroprocesory

2.3.1 Modul ATmega32

Jak již bylo zmíněno výše, srdcem celého zařízení je mikrokontrolér ATmega32. Jedná se o výrobek firmy Atmel. Tyto mikrokontroléry jsou velmi oblíbené, neboť poskytují solidní výkon za dobrou cenu, která se pohybuje okolo 145Kč. Jedná se tedy o jednočipový, 8-bitový mikropočítač využívající RISC architekturu. Více podrobnějších informací lze nalézt v dokumentaci [9].

Obsahuje:

- Paměti:
  - FLASH 32KB
  - EEPROM 1024B
  - SRAM 2KB
- Dva 8-bitové čítače/ časovače
- Jeden 16-bitový čítač/ časovač
- Taktovací frekvence 1-16MHz
- 4 obvody PWM
- AD převodník
- SPI, TWI, USART
- Watchdog
- A jiné…



Obrázek 18 - Pouzdro ATmega32

Tento modul tedy zpracovává informace z obou tachogenerátorů, modulu ATmega8 (odkud získává informace o úhlech natočení a naklonění hřídele), ovládacích prvků (dvě tlačítka, dvojice inkrementálních rotačních spínačů), generuje PWM signály pro výkonové členy motorů a ovládá LCD displej. Napájecí napětí mikropočítače ATmega32 je +5V a je přivedeno na pin VCC (č.10). Toto napětí zajišťuje lineární regulátor LM7805, který upravuje napětí +12V (z napájecího zdroje) na potřebných +5V.

Zapojení vývodů vypadá následovné:

PORT A	
PA0	Spodní tlačítko
PA1	Horní tlačítko
PA2	Pravý rotační enkodér
PA3	Levý rotační enkodér
PA4	Tlačítko pravého enkodéru
PA5	Tlačítko levého enkodéru
PA6, PA7	Převodník F/U

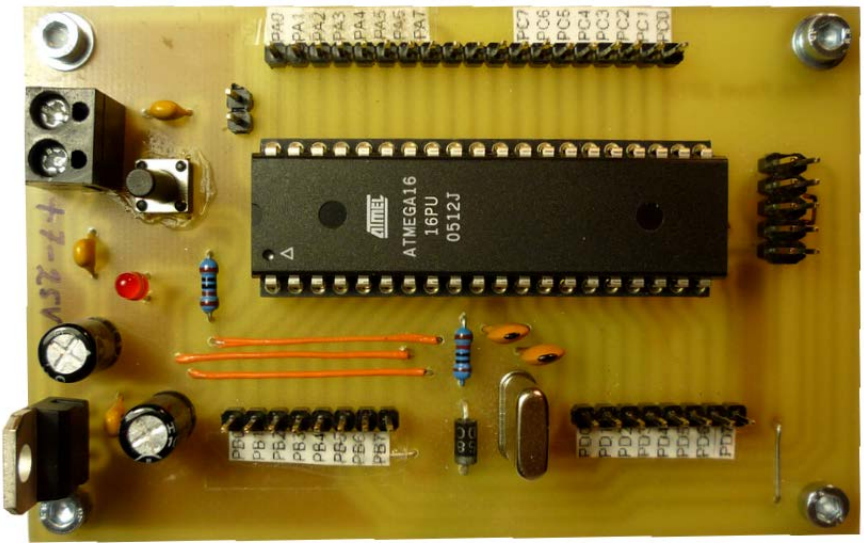
Tabulka 3 - Zapojení portu A, ATmega32

PORT C	
PC0	Datový vodič LCD, DB3
PC1	Datový vodič LCD, DB2
PC2	Datový vodič LCD, DB1
PC3	Datový vodič LCD, DB0
PC4	Povolovací vodič LCD, E
PC5	Register Select, RS

Tabulka 4 - Zapojení portu C, ATmega32

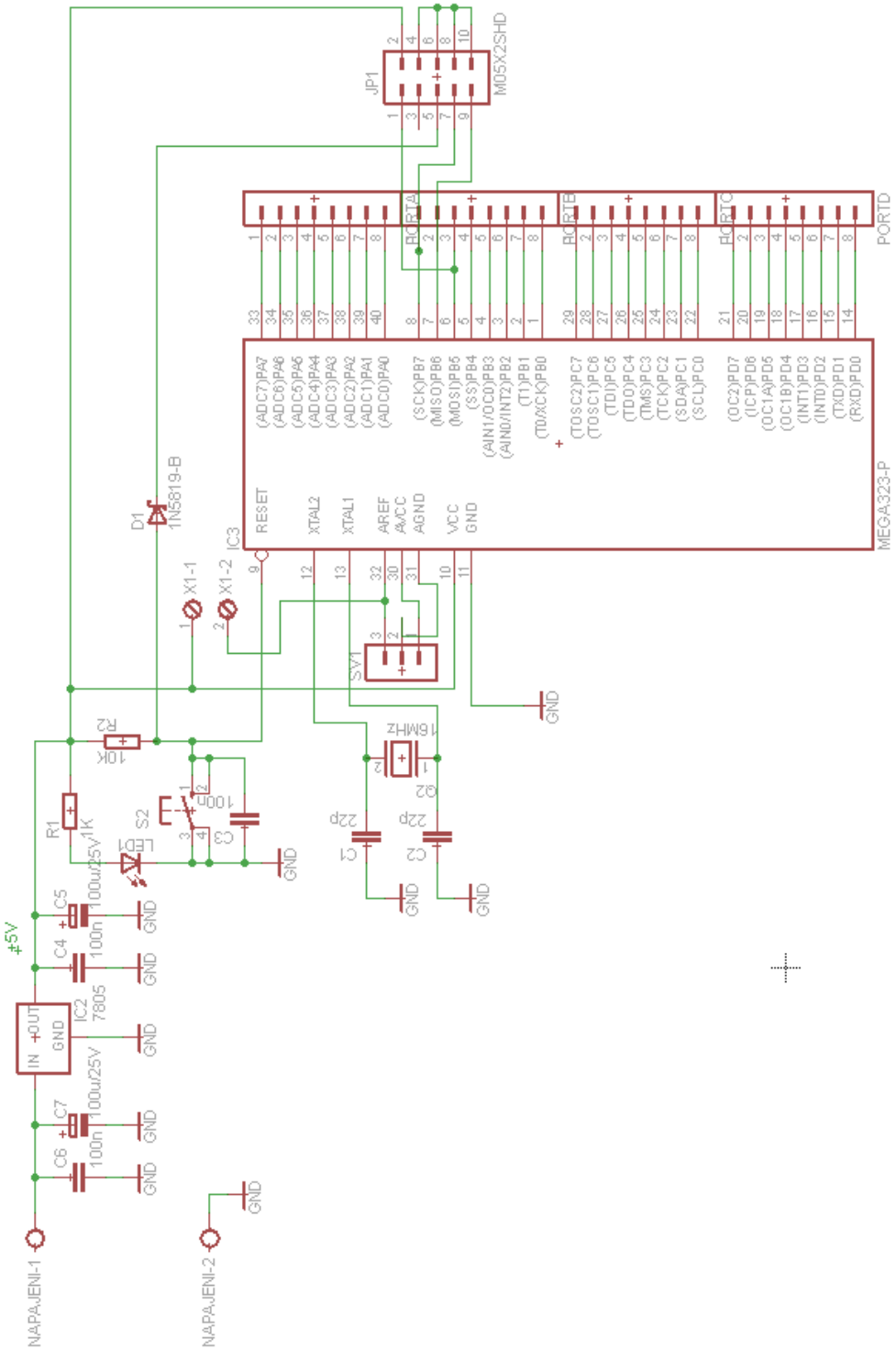
PORT D	
PD0	UART, RxD
PD1	UART, TxD
PD2	Pravý rotační enkodér
PD3	Levý rotační enkodér
PD4	PWM, OC1B
PD5	PWM, OC1A

Tabulka 5 - Zapojení portu D, ATmega32



Obrázek 19 - DPS ATmega32

Na obrázku výše (Obrázek 19) je uveden osazený plošný spoj modulu. Zapojení vychází z katalogového listu [9]. V levé horní části je konektor pro napájecí napětí. Napájecí napětí je dále upraveno lineárním regulátorem LM7805 [12] (opatřeným chladičem) v levém dolním rohu. Podél horní a dolní strany jsou vyvedeny piny mikrokontroléru. Mezi spodními je vidět mimo jiné i krystal, který je součástí generátoru hodinového signálu s kmitočtem 16MHz. V pravé části plošného spoje jsou vyvedeny piny pro připojení programátoru.



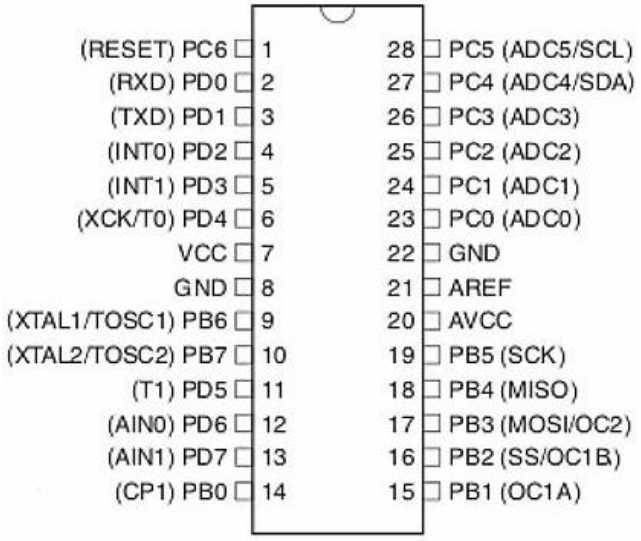
Obrázek 20 - Schéma zapojení ATmega32

2.3.2 Modul ATmega8

Vzhledem k tomu, že ATmega32 umožňuje použít pouze 3 zdroje externího přerušení a v tomto případě jsou potřeba 4 – pro dvě IRC čidla (náklonu/ natočení) a pro dva inkrementální rotační spínače ovládající jednotku, je tedy potřeba použít ještě druhý mikrokontrolér. Zde je možné použít například také ATmega32, ale stačí použít levnější ATmega8, který podporuje dva externí zdroje přerušení. Tento modul tedy zpracovává signály z IRC čidel náklonu a natočení hřídele. Po zpracování je odešle do modulu s ATmegou32, kde dochází k vyhodnocení. Dále generuje dva signály PWM se střídami danými úhly (natočení/ naklonění) a tyto signály přes dolnofrekvenční propusti přivádí na patici v zadní stěně krabičky (pro spojení s počítačem). Zapojení tohoto modulu je obdobné jako u modulu ATmega32, jen nejsou zapojeny piny PB0, PD5, PD6, PD7, neboť pro naše účely nebudou potřebné a vzhledem k potřebným rozměrům tohoto plošného spoje by se i obtížně vyváděly. Stejně jako u modulu s ATmega32, tak i zde jsou vyvedeny téměř všechny piny mikroprocesoru a to z toho důvodu, že řídící jednotka je navržena jako univerzální, pro řízení i odlišných systémů nežli je výše popisovaný a je proto nutné zajistit možnost připojení dalších vstupních, nebo výstupních zařízení.

ATmega8 obsahuje:

- Paměti:
  - FLASH 8KB
  - EEPROM 512B
  - SRAM 1KB
- Dva 8-bitové čítače/ časovače
- Jeden 16-bitový čítač/ časovač
- Taktovací frekvence 1-16MHz
- 4 obvody PWM
- AD převodník
- SPI, TWI, USART
- Watchdog
- A jiné…



Obrázek 21 - Pouzdro ATmega8

Zapojení vstupně/ výstupních pinů vypadá následovně:

PORT B	
PB1	PWM, OC1A
PB2	PWM, OC1B

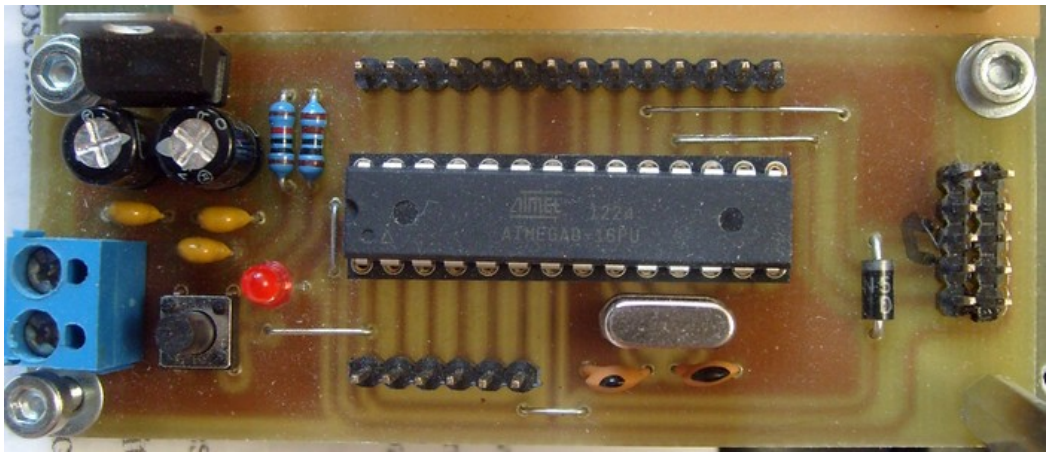
Tabulka 6 - Zapojení portu B, ATmega8

PORT C	
PC2	IRC snímač natočení
PC3	Optická závora, natočení
PC4	IRC snímač, naklonění
PC5	Optická závora, naklonění

Tabulka 7 - Zapojení portu C, ATmega8

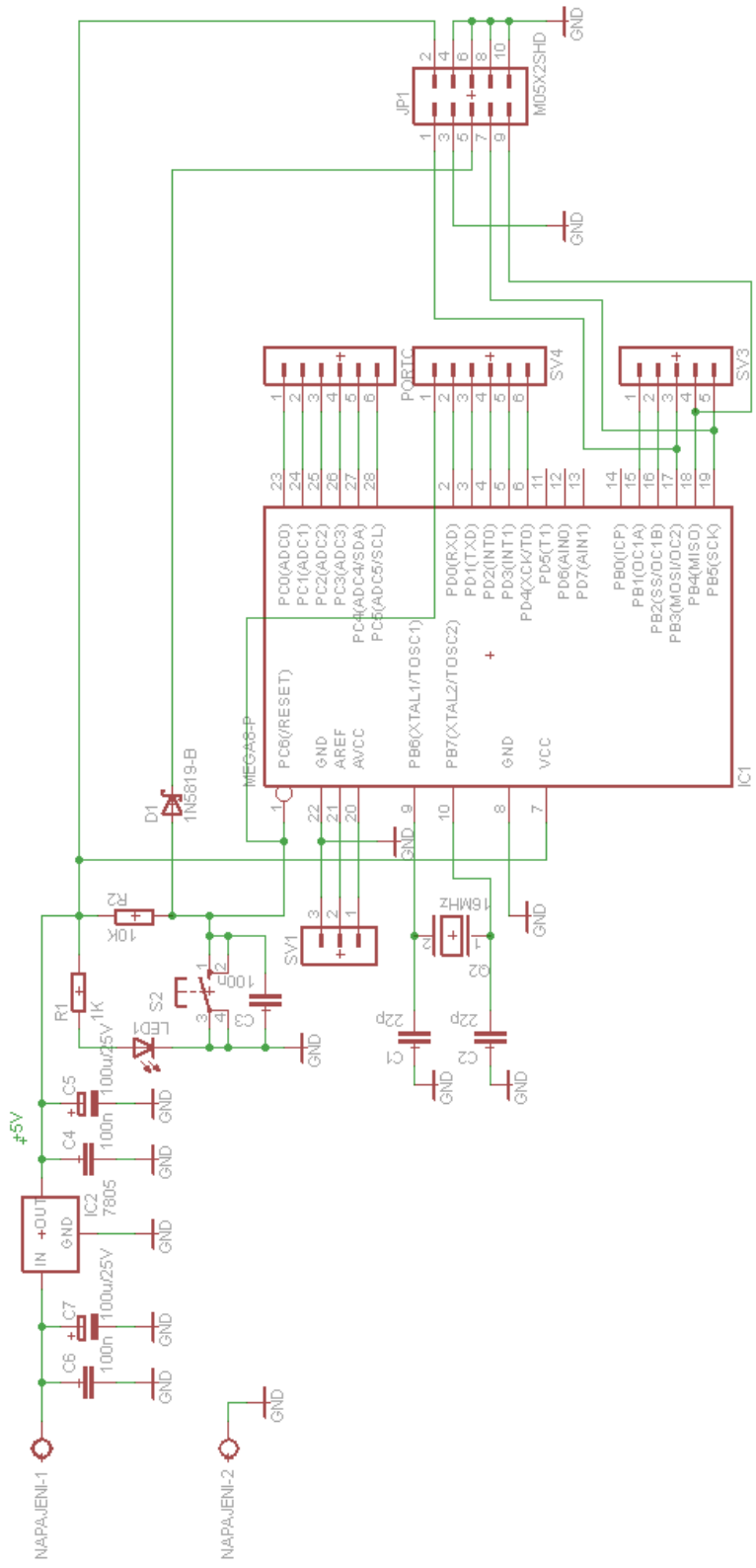
PORT D	
PD0	UART, RxD
PD1	UART, TxD
PD2	IRC snímač, natočení
PD3	IRC snímač, naklonění

Tabulka 8 - Zapojení portu D, ATmega8



Obrázek 22 - DPS ATmega8

Na obrázku (Obrázek 22) je zobrazen plošný spoj s mikroprocesorem. Vlevo dole je připájena svorkovnice pro přivedení napájecího napětí modulu, jehož velikost je daná možnostmi lineárního regulátoru. Stejně jako v předchozím modulu i v tomto je použit lineární regulátor LM7805 dovolující přivádět vstupní napětí až do velikosti 35V. Výstupní napětí regulátoru je samozřejmě +5V. Podrobnější informace se lze dozvědět z dokumentace [12]. Vpravo od svorkovnice se nachází resetovací tlačítko s LED u jeho pravého horního rohu. LED signalizuje zapnutí modulu. Podél horní a dolní strany jsou vyvedeny piny mikrokontroléru. V pravé části plošného spoje se nacházejí vyvedené piny dovolující připojení programátoru.



Obrázek 23 - Schéma zapojení ATmega8

2.3.3 Komunikace

Informace o velikostech úhlů naklonění a natočení hřídele získává ATmega8. Jak je ale přenést do mikroprocesoru ATmega32 a následně zobrazit na LCD? Pro přenos dat mezi oběma mikroprocesory je možné využít různá komunikační rozhraní, například *SPI*, *I^2C*, nebo *USART*.

SPI (Serial Peripheral Interface) - sériové periferní rozhraní. Sběrnice SPI pracuje se třemi, nebo čtyřmi vodiči. Jsou to:

- CLK - hodinový signál pro synchronizace přenosu po datovém vodiči.
- DATA - datový vodič, ten může obsahovat buď vysílaná, nebo přijímaná data. Případně se použijí dva vodiče, jeden pro vysílaná a druhý pro přijímaná data.
- CS (Chip Select) - výběr obvodů se kterým bude master zrovna komunikovat.

Výhodou SPI tedy je, že k jednomu tzv. *master* zařízení lze připojit více tzv. *slave* zařízení. Adresace těchto zařízení se provádí pomocí vodičů CS (někdy označováno jako SS). Při log. 0 je aktivní příjem a vysílání zvoleného zařízení. *Master* zařízení řídí komunikaci pomocí hodinového signálu a také určuje jedno ze *slave* zařízení se kterým bude právě komunikovat. *Slave* zařízení, pokud je aktivováno jeho SS, vysílá data dle hodinového signálu.

I^2C (Inter-Integrated Circuit) - respektive TWI, je dvouvodičová obousměrná sériová sběrnice vyvinutá firmou Philips pro používání převážně pro komunikaci na krátké vzdálenosti.

- SDA - pro přenos sériových dat
- SCL - hodinový signál přenosu

Obě linky musejí být připojeny přes *pull-up* rezistory ke kladnému pólu napájecího napětí. Tím se zajistí práce linek v obou směrech. Každý obvod může tedy pracovat jako vysílač i jako přijímač. Taktéž zde je možné propojit více obvodů. Maximální počet je 128. Každé zařízení má přidělenou 7 bitovou adresu.

USART (Universal Synchronous/ Asynchronous Receiver and Transmitter) - pro sériovou komunikaci. Lze nastavit buď pro synchronní, nebo asynchronní režim.

- Synchronní - v tomto režimu potřebujeme ještě další vodič pro synchronizaci přenosu dat. Oproti asynchronnímu režimu, můžeme dosahovat vyšších rychlostí.
- Asynchronní - častěji využívaný, hlavně proto, že počítačové rozhraní RS-232 umí komunikovat pouze asynchronně. Využívá dvou vodičů, TxD (transmit/ vysílání) a RxD (receive/ příjem), samozřejmostí je společná zem obou zařízení. Vysílání a příjem může probíhat zároveň. Toto se označuje slovním spojením *plný duplex*.

Pro komunikaci mezi oběma mikrokontroléry bylo zvoleno rozhraní USART v asynchronním režimu. K realizaci komunikace je nutné propojit určité piny obou mikroprocesorů, a to pin PD0 (Rxd) mikroprocesoru ATmega32 s pinem PD1 (TxD) mikr. ATmega8 a podobně pin PD1 (TxD) s pinem PD0 (RxD). Dále je zapotřebí provést

inicializaci rozhraní, kde se nastaví mimo jiné i přenosová rychlost v registru UBRR. Zde registr UBRRH obsahuje 4 bity nejvyšší váhy a registr UBRRL osm bitů nižší váhy. Hodnota UBRR se vypočte podle následujícího vztahu:

$$UBRR = \frac{f_{osc}}{16 * BaudRate} - 1$$

(Tento vztah se liší v případě synchronního módu i v případě synchronního módu s dvojitou rychlostí).

Dále v registru UCSRB se nastaví na log. 1 bity TXEN (povolení činnosti vysílače), RXEN (povolení činnosti přijímače). V registru UCSRC je potřeba nastavit na úroveň log. 1 bity URSEL (výběr registru), UCSZ1 a UCSZ0 pro nastavení počtu datových bitů ve vysílacím a přijímacím rámci na hodnotu 8 bitů. Možnosti nastavení jsou uvedeny v tabulce:

UCSZ2	UCSZ1	UCSZ0	Délka znaku
0	0	0	5 bitů
0	0	1	6 bitů
0	1	0	7 bitů
0	1	1	8 bitů
1	0	0	-
1	0	1	-
1	1	0	-
1	1	1	9 bitů

Tabulka 9 - Nastavení délky datové části rámce

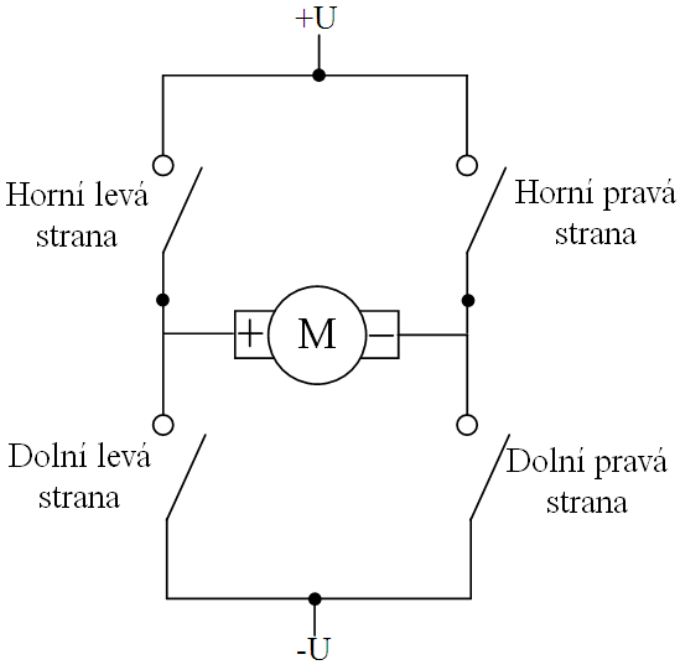
Tím je provedena inicializace. Vysílání i příjem se provádí po jednotlivých znacích. Před odesláním je potřeba, aby příznakový bit UDRE registru USCRA, byl na log. 1, což značí, že vyrovnávací registr je prázdný a připravený na zápis nového znaku. V případě, že UDRE není v log. 1, je nutné vyčkat, než se vyrovnávací registr vyprázdní. Odeslání dat se provede uložením dat do údajového registru UDR. Při příjmu dat se čeká, nežli je příznakový bit RXC registru UCSRA v log. 0. RXC v log. 0 značí, že přijímací zásobník je prázdný. V případě, že RXC je v log.1, tak přijímací zásobník obsahuje nepřečtená data.

2.4 Výkonový člen elektromotorů

Jak pro hlavní rotor, tak pro ocasní je použito stejné zapojení. Jedná se o regulátory napětí, které zajišťují regulaci otáček motorů při ovládacím napětí 0-5V. V případě hlavního rotoru je požadavek pouze na jeden směr otáčení, čehož se dosáhne odstraněním propojky X4, zatímco u ocasního je nutné zajistit i změnu polarity na svorkách motoru z důvodu možnosti otáčení na obě strany - zde se propojka ponechá. Tato propojka připojuje, resp. odpojuje trimr P2, kterým se posouvá klidová poloha, tedy poloha kdy je na motoru nulové napětí.

V zapojení jsou použity operační zesilovače (LM358), které jsou zapojeny v tak zvané protifázi, tedy pokud na výstupu operačního zesilovače OZ2A je kladné napětí, tak na výstupu OZ2B je napětí záporné[5].

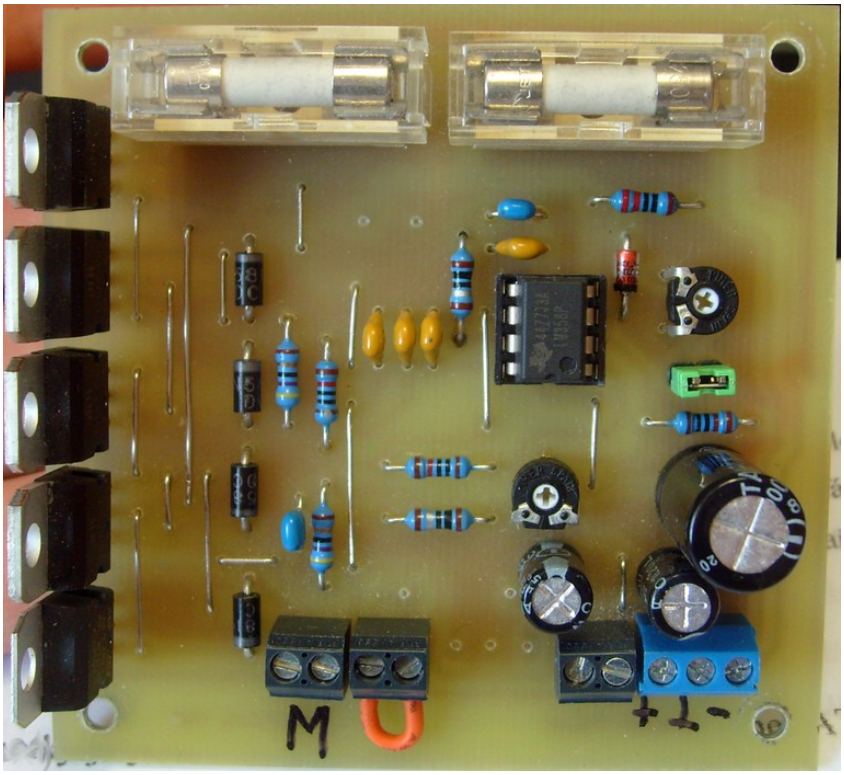
Dále, čtveřice tranzistorů je zapojena jako tzv. H-můstek. Obecně, označení H se mu dostalo tak, že zapojení čtyř spínacích prvků v „rozích“ a motor tvořící příčku připomínají písmeno H. Spínací prvky jsou zapínány v párech a to tak, že buď jsou sepnuty horní levá a dolní pravá strana, nebo dolní levá a horní pravá strana, nikdy ne horní levá a horní pravá, resp. dolní levá a dolní pravá. Pokud by byly sepnuty oba spínací prvky na jedné straně (horní levá a dolní levá), došlo by ke zkratu.



Obrázek 24 - H-můstek - obecně

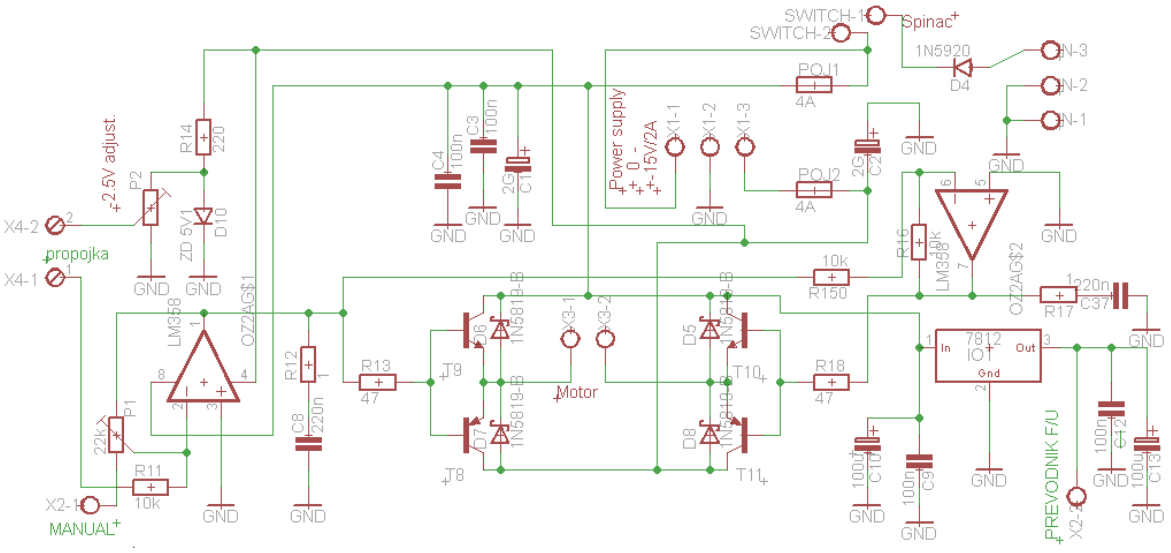
Po sepnutí horního levého a dolního pravého spínacího prvku se bude motor otáčet jedním směrem. Když sepneme horní pravý a dolní levý spínací prvek, bude se motor točit směrem opačným. Jako spínací prvky se dají použít například relé, nebo tranzistory (zde použito).[6]

[5] Havlíček Libor, Modelování a řízení vícerozměrové soustavy, 46
[6] MCmanis Chuck. H-Bridges: Theory and Practice.



Obrázek 25 - Výkonový člen

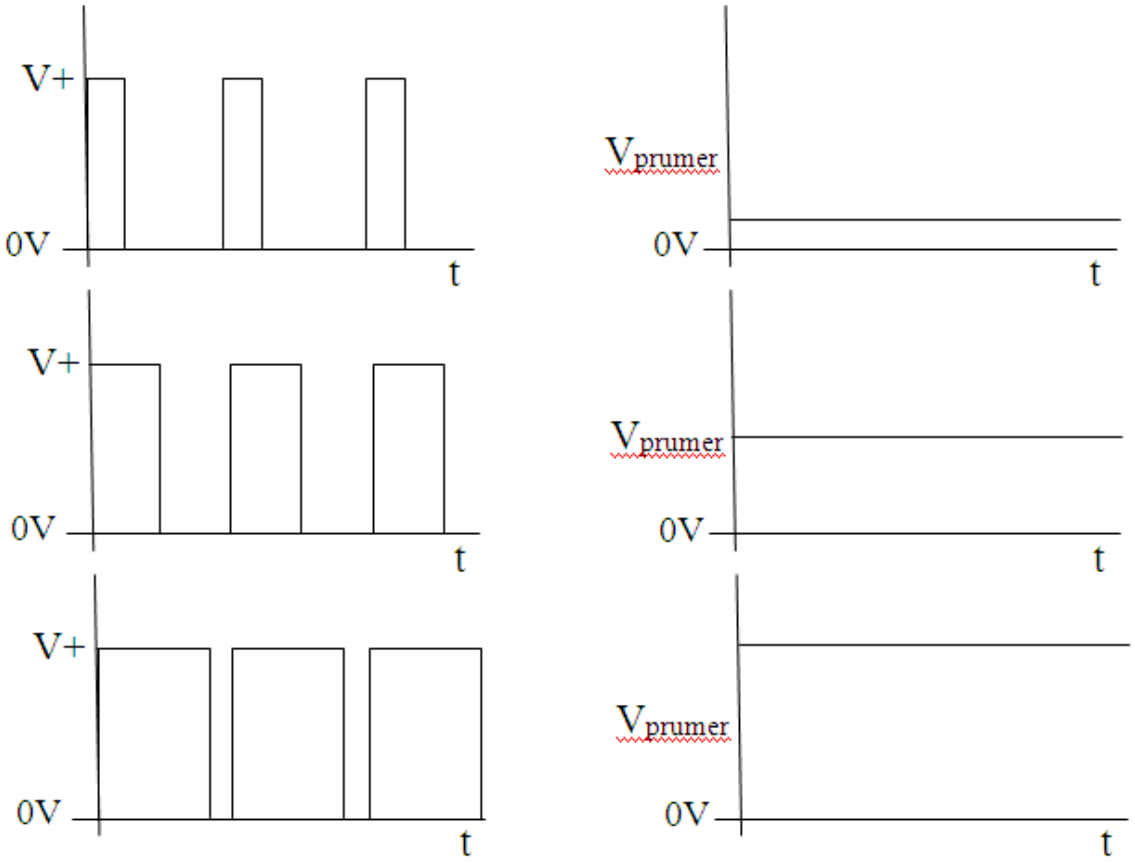
Součástí tohoto zapojení je i lineární regulátor napětí 7812, který slouží pro napájení převodníku frekvence/ napětí.



Tabulka 10 - Schéma zapojení výkonového členu motoru

2.5 RC článek

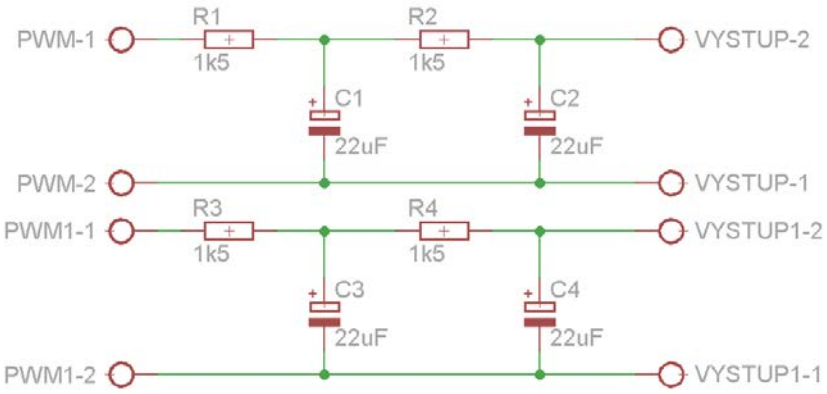
Integrační článek, neboli dolnofrekvenční RC filtr, převádí PWM signál na úroveň stejnosměrného napětí, viz. obrázek níže (Obrázek 26). Tedy pracuje jako jednoduchý D/A převodník.



Obrázek 26 - Převod PWM na SS

Toto SS napětí vstupuje do výkonových členů motorů, kde zajišťuje jejich řízení a tedy i řízení otáček motorů. Možná by se na první pohled zdálo, že je zbytečné používat výše zmíněné výkonové členy, neboť by se motory daly řídit jen přes integrační článek, ale tento způsob by nám dovoloval používat změnu napětí pouze od 0 do 5V, což v nějakých případech není dostatečné a je nutné použít operační zesilovač pro zvýšení napětía samozřejmě i proudu. Stejná dolnofrekvenční propust je použita ještě na dalším místě a to pro převod PWM signálu generovaného mikroprocesorem ATmega8 se změnou střídy v závislosti na naklonění, respektive natočení. Tento signál je po převodu přiveden na patici v zadní stěně pouzdra pro spojení s akviziční kartou a následnému zpracování v počítači.

Zapojení dolnofrekvenčního filtru je na obrázku níže (Obrázek 27). Jsou zde použity dvojité RC články.



Obrázek 27 - Schéma zapojení RC článku

2.5.1 PWM

Pulse Width Modulation, pulsně šířková modulace. Jde o diskrétní modulaci využívanou pro přenos analogového signálu využitím dvoustavového signálu. Signál se přenáší pomocí střídy [8]. Střída *D* je rovna poměru času zapnutí a celkové periodě:

$$D = \frac{t_{zapnutí}}{t_{perioda}}$$

Na obrázku (Obrázek 26) vlevo jsou vidět tři průběhy signálu. Horní pro střídu 33%, prostřední pro střídu 50% a poslední pro 83%. Střída 33% znamená, že 33% celkového času periody je nastaveno napětí na log.1, po zbytek času na log. 0. Na výstupu se tedy objevují napěťové pulsy, které se vyhladí (demodulují) dolnofrekvenční propustí na hodnotu napětí dané střídou, což ve výsledku funguje jako jednoduchý D/A převodník (viz. odstavec 2.5). Pulsně šířková modulace se využívá především pro regulování motorů, nebo třeba i jasu svitu žárovek LED diod, a podobně.

Pro generování PWM signálu jsou použity výstupní piny PD4 (OC1B) a PD5 (OC1A) v případě mikrokontroléru ATmega32 a PB1 (OC1A) a PB2 (OC1B) u ATmega8. Pro nastavení režimu PWM, frekvence a ovládání výstupních pinů se používají registry TCCR1A a TCCR1B čítače/časovače1. Dále popsáno patří k mikroprocesoru ATmega8, pro ATmega32 je nastavení obdobné.

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Tabulka 11 - Registr TCCR1A

Bity COM1A1 a COM1A0, respektive COM1B1 a COM1B0, slouží pro nastavení ovládání výstupních pinů při jednotlivých režimech. Režimy existuje několik. Například *normální režim, režim rychlé PWM, fázově korektní a fázově a frekvenční korektní.* V této řídící jednotce byly v obou případech použity režimy *rychlé PWM* proto zde uvedu nastavení jen toto. Ostatní je možné nalézt v dokumentaci příslušného mikroprocesoru.

COM1A1/ COM1B1	COM1A0/ COM1B0	Popis
0	0	OC1A a OC1B jsou odpojené.
0	1	Změna logické úrovně na pinu OC1A při režimu 15. OC1B je odpojený.
1	0	OC1A a OC1B v log. 0 při shodě registrů a v log. 1 při hodnotě BOTTOM - spodní hodnota čítače.
1	1	OC1A a OC1B v log. 1 při shodě registrů a log. 0 při BOTTOM hodnotě.

Tabulka 12 - ovládání výstupních pinů, *rychlá PWM*

Dále, bity FOC1A a FOC1B jsou pro režimy, kde se nepoužívá PWM.

Bity WGM11 a WGM10, respektive WGM13 a WGM12 registru TCCR1B, se nastavuje režim PWM. Těchto režimů je celkem 15.

	WGM13	WGM12	WGM11	WGM10	Režim PWM	TOP	Aktualizace OCR1x	T0V1 je nastaven na:
0	0	0	0	0	Normální	0xFFFF	Ihned	MAX
1	0	0	0	1	Fázově korektní, 8bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	Fázově korektní, 9bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	Fázově korektní, 10bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Ihned	MAX
5	0	1	0	1	Rychlá PWM, 8bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Rychlá PWM, 9bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Rychlá PWM, 10bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	Fázově a frekvenčně korektní	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	Fázově a frekvenčně korektní	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	Fázově korektní	ICR1	TOP	BOTTOM
11	1	0	1	1	Fázově korektní	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Ihned	MAX
13	1	1	0	1	Rezervováno	-	-	-
14	1	1	1	0	Rychlá PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Rychlá PWM	OCR1A	BOTTOM	TOP

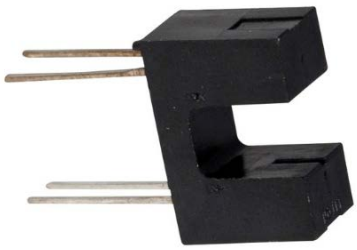
Tabulka 13 -Režimy PWM

V této řídící jednotce byl použit režim číslo 5, *Rychlá PWM, 8bit*. Při tomto režimu se čítač inkrementuje dokavad není roven hodnotě TOP, tedy hodnotě 255. Při dosažení této hodnoty se nastaví příznak přetečení T0V1. Poté, v dalším hodinovém cyklu, se jeho obsah vynuluje.

2.6 Optické snímače polohy

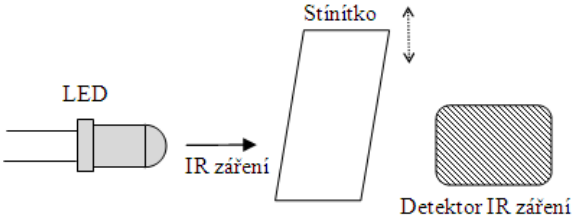
2.6.1 Optická závora

Optické závory mají široké uplatnění. Běžně se používají například v zabezpečovací technice, výrobních linkách, dopravě, jako různé senzory, a podobně. Většinou pracují na principu infračerveného záření. To proto, že potlačuje rušení okolním světlem. Nevýhoda je krátký dosah, který se dá zvětšit použitím laserové optické závory. Laserové závory mají dosah až stovky metrů.



Obrázek 28 - Optická závora

Pro aplikaci ve výše zmíněném laboratorním modelu samozřejmě plně postačují optické závory s LED produkující infračervené záření. Taková závora se skládá z diody, detektoru „infra“ záření a stínítka přerušující světelný tok z diody na detektor. Samozřejmě se optická závora prodává i jako jeden blok, viz. obrázek (Obrázek 28).

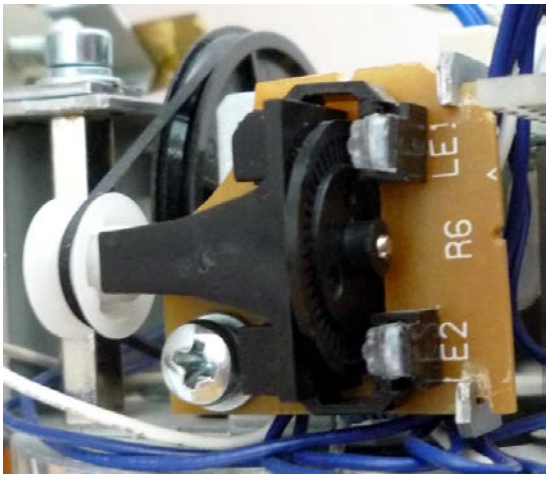


Obrázek 29 - Princip optické závory

Důvod použití optických závor je nulování obsahů čítačů mikroprocesoru po dosažení výchozí polohy v horizontálním a vertikálním směru. Nulování je nutné pro správné odečítání úhlů náklonu a natočení.

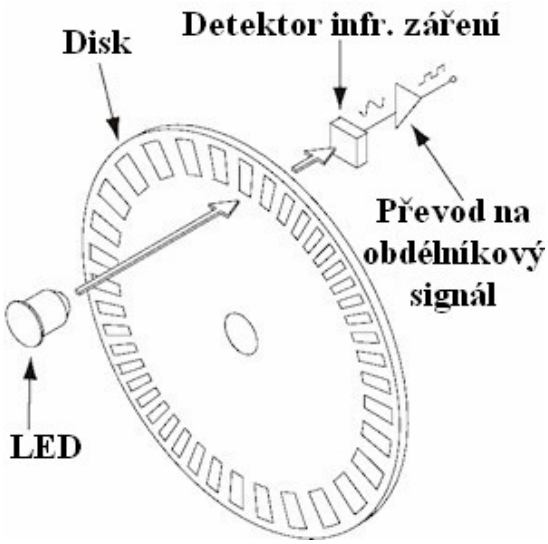
2.6.2 Inkrementální rotační čidla

Pro snímání náklonu a natočení ramena jsou použita tzv. IRC čidla, neboli optické rotační inkrementální snímače (resp. enkodéry). Jedná se o poměrně rozšířené součástky používající se v mnoha elektronických zařízeních. Například v počítačových myších, tiskárnách, scannerech, apd.



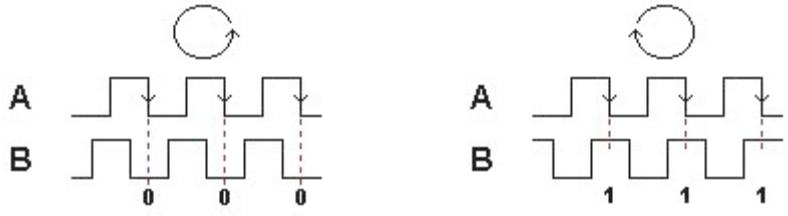
Obrázek 30 - Inkrementální rotační enkodér

IRC čidlo se skládá z disku obsahujícím pravidelné otvory, LED diody produkující infračervené záření a detektorem infračerveného záření. Paprsek záření prochází pouze otvory v disku, zbytek disku je neprůhledný a pohlcuje světlo. Při současném otáčení disku dochází ke vzniku světelných impulsů dopadajících na detektor, který je převede na obdélníkový signál. Tento signál je dále zesílen a zpracován.



Obrázek 31 - Princip IRC

Enkodér nejčastěji generuje dva stejné obdélníkové průběhy, které jsou vzájemně fázově posunuty o 90°. Dva z toho důvodu, že je potřeba rozeznat nejen rychlost, ale i směr otáčení, což by z jednoho signálu obdélníkového průběhu bylo nemožné.



Obrázek 32 - IRC - rozeznání směru otáčení

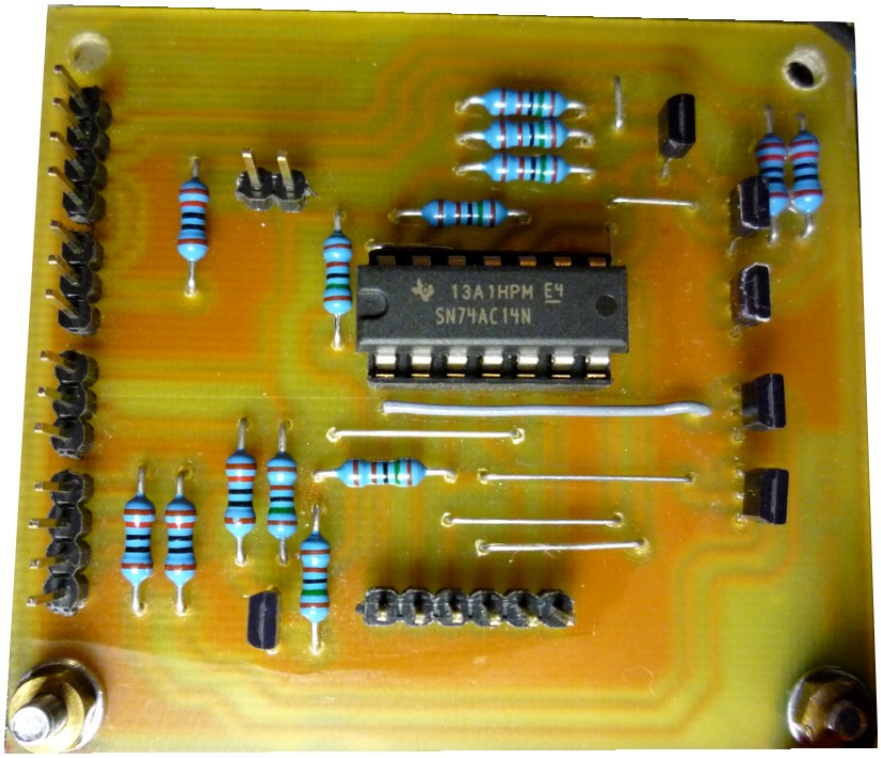
Vyhodnocování se provádí softwarově v mikroprocesoru ATmega8. Jeden signál je přiveden na pin určený pro externí přerušení a reaguje na změnu logické úrovně. Druhý signál může být do mikroprocesoru přiveden prakticky kamkoliv. Při vyvolání přerušení se porovnávají logické hodnoty na těchto dvou pinech a dle toho se určí směr otáčení, respektive inkrementace nebo dekrementace proměnné. Například takto:

```
ISR (INT0_vect){
    if ((PIND & (1<<PD2))==4 && (PINA & (1<<PA5))==32){
        pocitadlo=pocitadlo-1;      //dec 4
        if(OCR1A>10)
            OCR1A -= 5; //ubrat stridu PWM
    }
```

```
        if ((PIND & (1<<PD2))==4 && (PINA & (1<<PA5))==0){
                pocitadlo=pocitadlo+1;      //inc 32
                if(OCR1A<250)
                        OCR1A += 5; //pridej stridu PWM
        }
}
```

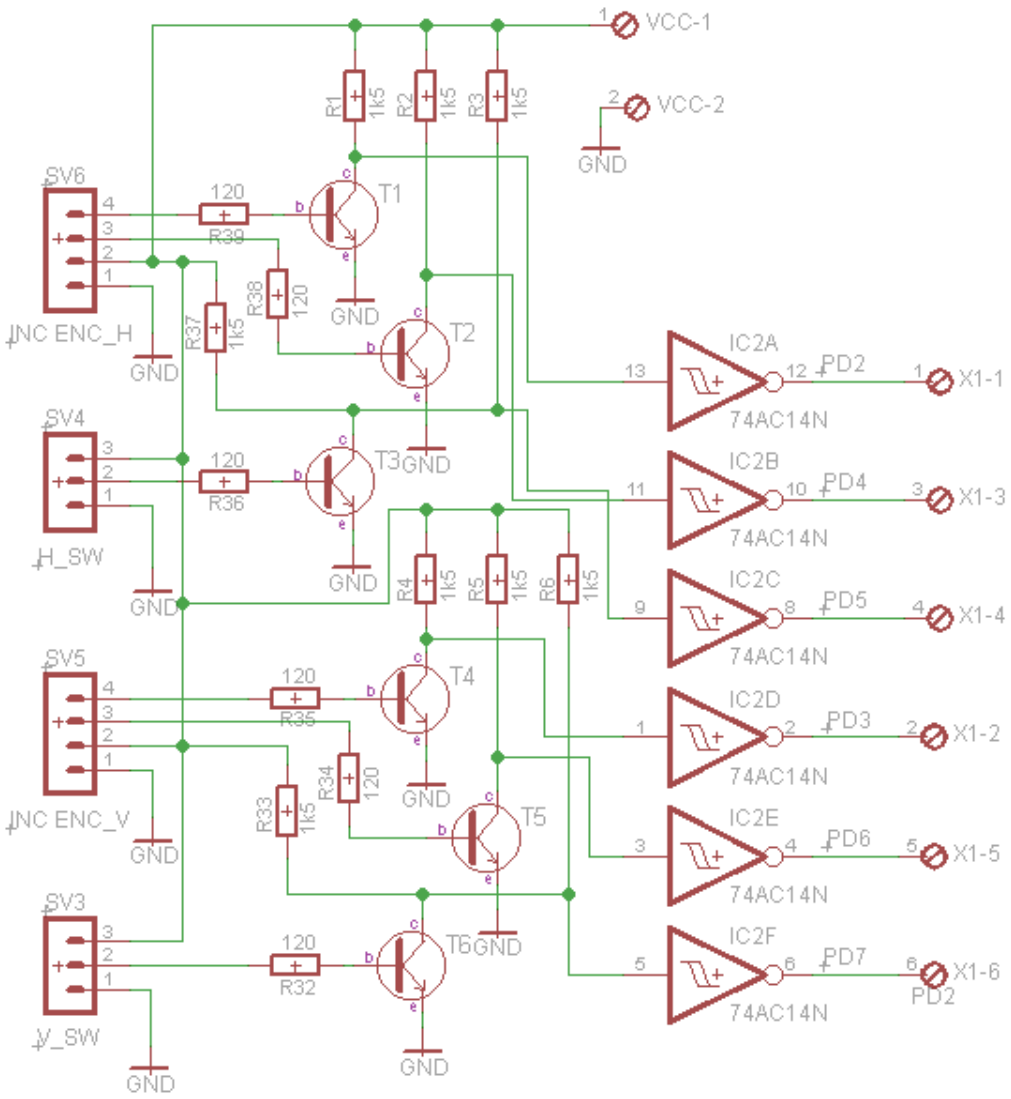
2.6.3 Modul pro úpravu signálu z optických čidel

Signály, které nám vystupují z optických enkodérů a optických závor jsou dále upraveny tranzistory zapojenými ve spínacím režimu a následně vytvarovány hradly 74HC14N, aby byly získány pravidelné obdélníkové signály, neboť původní signály mají poměrně dlouhé hrany a mohlo by docházet přinejmenším k překmitům. Výše zmíněný obvod obsahuje šest invertujících Schmittových klopných obvodů[7].



Obrázek 33 - Modul úpravy signálů z optických čidel

[7] HAVLÍČEK Libor, *Modelování a řízení vícerozměrové soustavy*



Obrázek 34 - Schéma zapojení modulu pro úpravu signálů z optických čidel

2.7 Měření otáček motorů

2.7.1 Tachogenerátor

Tachogenerátor je součástí obou použitých motorů a slouží pro měření úhlové rychlosti otáčení. Na výstupu generuje střídavý signál přímo úměrný otáčkám. Při změně rychlosti otáčení nedochází jen ke změně napětí, ale i frekvence.

Existují tachogenerátor s otočným magnetem, nebo s bubínkovým rotorem. U prvně zmiňovaného je rotor tvořen permanentním magnetem. Tento magnet při otáčení indukuje elektromotorické (střídavé) napětí do statoru tvořeného cívkou, takže střídavé napětí se získává ze statických cívek a ne ze sběračů (ty tachogenerátor ani neobsahuje).

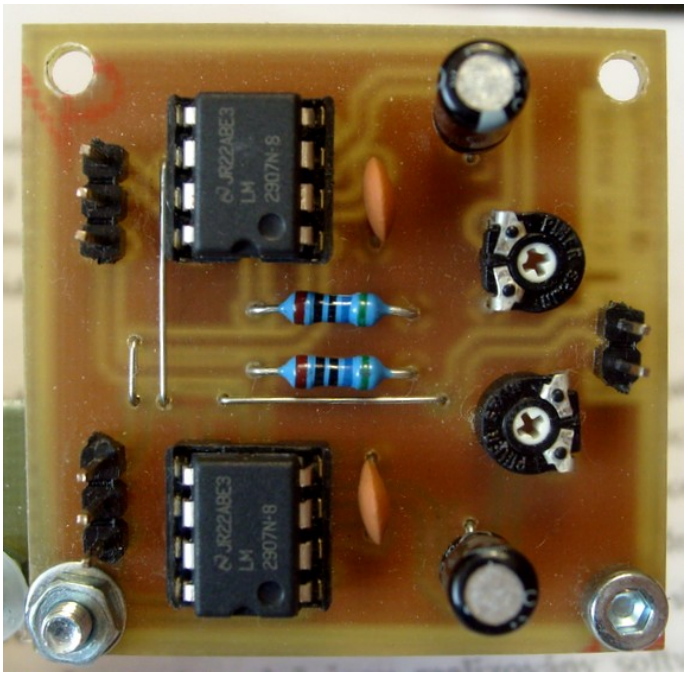
Tachogenerátor s bubínkovým rotorem má stator tvořený dvěma cívkami vzájemně prostorově otočené o 90°. Jedna cívka je napájena střídavým napětím a označuje se jako budící. Na druhé cívce se indukuje napětí.

V současnosti se dají sehnat buď jako samostatná součást, nebo společně v jednom pouzdře s elektromotorem. Příklad samostatného tachogenerátor je na obrázku (Obrázek 24).

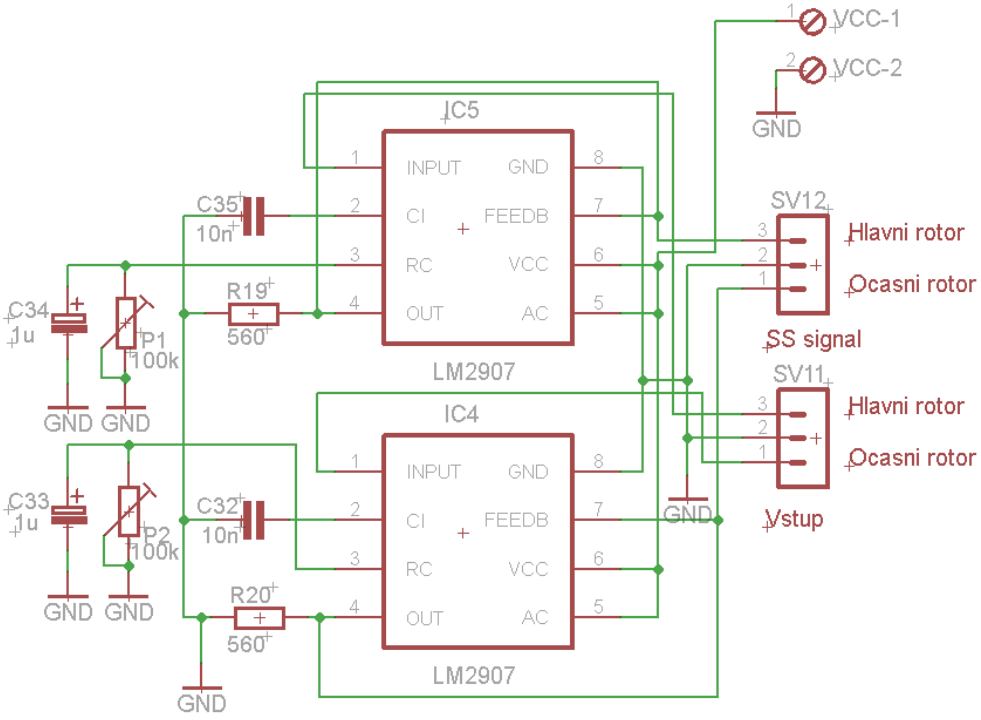
Obrázek 35 - Tachogenerátor

2.7.2 Převodník frekvence/ napětí

Jak již bylo zmíněno výše, z tachogenerátoru vystupuje sinusový signál, který má napětí a kmitočet přímo úměrný otáčkám a je potřeba ho usměrnit. Což se na první pohled zdá jako jednoduchý úkol, ale není tomu tak. Jde o signál ve formě malého střídavého napětí, které bývá zašuměno. Šum je znatelnější při nízkých otáčkách. Z těchto důvodů je výstupní napětí vytvarováno na obdélníkový signál. Ten se dá snadno převést například na signál PWM a dále zpracovávat. Úpravu signálu zajišťuje integrovaný obvod LM2907. Jde o převodník frekvence/ napětí. Z převodníku již vystupuje stejnosměrné napětí, které je přímo úměrné otáčkám tachogenerátoru a tedy vlastně i motoru, neboť jsou součástí jednoho celku. Toto napětí je přivedeno na vstupní pin mikroprocesoru ATmega32 a dále zpracováno A/D převodníkem.



Obrázek 36 - Převodník F/U



Obrázek 37 - Schéma zapojení převodníku F/U

2.7.3 A/D převodník

Analogově digitální převodník je zařízení pro převod analogového signálu na digitální, neboli číslicový. Nejčastější použití je pro měření napětí, proudů, teploty a podobně. Mezi základní parametry převodníků patří [6]:

- *Rozlišovací schopnost* - počet rozlišitelných úrovní analogového signálu.
- *Rozsah* - rozdíl maximální a minimální hodnoty analogového signálu. Nejčastěji bývá 0-V_{CC}.
- *Rychlost převodu* - počet možných převodů za jednotku času, nebo čas potřebný pro vykonání jednoho převodu.
- *Krok kvantování* = citlivost, nejmenší hodnota, kterou převodník dokáže rozlišit.
- *Chyba kvantování* - maximální rozdíl mezi hodnotou analogového signálu a její maximální hodnotou odpovídající digitální hodnotě.

Výstupní digitální hodnotu převodu lze vypočíst dle vztahu:

$$ADC = \frac{(V_{in} * 2^n)}{V_{ref}}$$

(V_{ref} - referenční napětí, V_{in} - vstupní napětí, n - rozlišení)

Pro výpočet kroku kvantování (citlivosti) je možné použít vztah:

$$LSB = \frac{V_{ref}}{2^n}$$

ATmega32 obsahuje zabudovaný 10bitový AD převodník. 10bitový znamená, že dokáže rozlišit 2^10, tedy 1024 úrovní. Další z vlastností tohoto převodníku jsou:

- absolutní přesnost ±2 LSB (krok kvantování);
- čas převodu 65-260us;
- vstupní rozsah 0-5V;
- přerušení po skončení převodu, atd.

Než se převodník začne používat, musí se provést drobná nastavení. Ta se provádí v registrech ADMUX a ADCSRA.

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
Read/write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Tabulka 14 - registr ADMUX

Bity REFS1 a REFS0 se vybírá zdroj referenčního napětí. Možnosti jsou uvedeny v tabulce (Tabulka 15).

REFS1	REFS0	Výběr referenčního napětí
0	0	Referenční napětí přivedeno na pin AREF. Interní reference je vypnuta.
0	1	Referenční napětí představuje napájecí napětí. Na pin AREF je potřeba připojit blokovací kondenzátor.
1	0	Rezervované.
1	1	Vnitřní referenční napětí 2,56V. Na pin AREF je potřeba připojit blokovací kondenzátor.

Tabulka 15 - Výběr referenčního napětí AD převodníku

Bitem ADLAR je možné zarovnat výsledek převodu v registrech ADCL a ADCH (jde o výstupní registry obsahující výsledek převodu, dohromady tvoří 16bitový registr ADC). V případě, že je do bitu ADLAR zapsána logická 0, vypadají registry následovně:

Bit	15	14	13	12	11	10	9	8
ADCH	-	-	-	-	-	-	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
	7	6	5	4	3	2	1	0

Tabulka 16 - ADLAR = 0

Opačný případ, ADLAR = 1:

Bit	15	14	13	12	11	10	9	8
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	-	-	-	-	-	-
	7	6	5	4	3	2	1	0

Tabulka 17 - ADLAR = 1

Dále bity MUX3, MUX2, MUX1, MUX0 se vybírá analogový vstupní kanál, který je připojen na vstup AD převodníku. Možnosti jsou:

MUX3	MUX2	MUX1	MUX0	Vstup/pin
0	0	0	0	ADC0/PA0
0	0	0	1	ADC1/PA1
0	0	1	0	ADC2/PA2
0	0	1	1	ADC3/PA3
0	1	0	0	ADC4/PA4
0	1	0	1	ADC5/PA5
0	1	1	0	ADC6/PA6
0	1	1	1	ADC7/PA7
1	1	1	0	1.30V
1	1	1	1	0V

Tabulka 18 - Výběr analogového vstupního kanálu

Registr ADCSRA je řídící a stavový registr. Zde je potřeba upozornit na malou drobnost. Celé nastavování a rozložení bitů registrů AD převodníku je stejné pro ATmega8 i ATmega32 až na bit číslo 5 registru ADCSRA. U ATmega8 jde o bit ADFR a nastavuje se

jím opakovací režim. Zatímco u ATmega32 je označení bitu ADATE a po zapsání log. 1 je povoleno automatické spouštění ADC.

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Tabulka 19 - registr ADCSRA

Bit ADEN slouží pro povolení činnosti převodníku, 1 - zapnuto, 0 - vypnuto. V případě, že převodník je nastaven na jednorázový režim, tak musí být do ADSC zapsána log. 1 před každým převodem. Po ukončení převodu se výsledek zapíše do registrů ADCH/ ADCL a do ADSC se zapíše logická 0. ADIF, neboli příznak přerušení se nastaví na log. 1 vždy, když převod skončí a registry ADCH, ADCL obsahují nové hodnoty. Nastavením bitu ADIE na log. 1 povolíme přerušení, které se vyvolá po skončení AD převodu. Pro nastavení předděličky slouží bity ADPS2, ADPS1, ADPS0. Volby jsou uvedené v tabulce [Tabulka 20].

ADPS2	ADPS1	ADPS0	Dělící faktor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	14
1	0	1	32
1	1	0	64
1	1	1	128

Tabulka 20 - Volba předděličky AD převodníku

Nejprve je tedy nutné provést inicializaci převodníku funkcí *ADCinit()*. Dále pro přečtení hodnoty v registru ADC se zavolá funkce *ADCcti*. Nutno zmínit, že je potřeba číst ze dvou analogových vstupů (dva tachogenerátory), konkrétně PA6 a PA7. V této funkci se tedy nejprve nastaví do registru ADMUS požadovaný pin, ze kterého se má číst, dále spustí převod, vyčká se než je převod dokončen a zakáže se činnost zapsáním log. 0 do bitu ADSC registru ADCSRA.

2.8 LCD

2.8.1 DEM 20486 SYH-LY

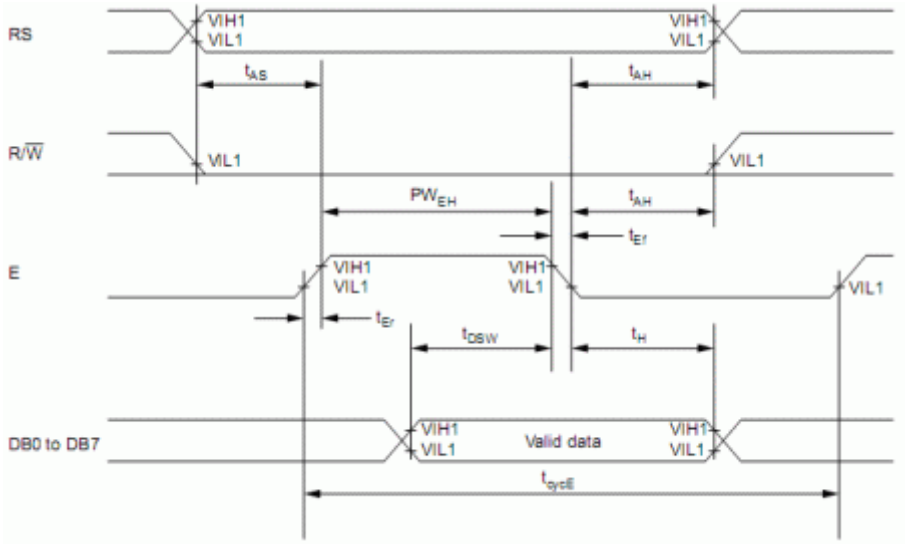
Jak je zmíněno v úvodu, řídící jednotka je provedena jako samostatné zařízení, které pracuje samostatně. Právě z toho důvodu je opatřena LCD displejem, který umožňuje pohodlnější práci s jednotkou. Jedná se o výrobek firmy *DISPLAY Elektronik GmbH* a je schopen zobrazit celkem 80 znaků (4 řádky po dvaceti znacích). Napájecí napětí je v rozmezí 2,7 až 5V a obstarává jej lineární regulátor napětí LM7805, který napájí i modul ATmega32. Rozložení pinů na konektoru vypadá následovně:

Číslo pinu	Symbol	Funkce
1	V_{SS}	Uzemnění GND
2	V_{DD}	Napájecí napětí 2,7 - 5,5V
3	V_0	Napětí nastavující kontrast
4	RS	Datový pin určující zda chceme přenášet data nebo instrukce: RS = 0…Instrukční registr RS = 1…Datový registr
5	R/W	Nastavujeme buď zápis, nebo čtení z displeje: R/W = 1…čtení R/W = 0…zápis
6	E	Povolení - zapnutí/vypnutí displeje
7	DB0	Obousměrné datové piny
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED- (K)	Podsvětlení displeje - katoda
16	LED + (A)	Podsvetlení displeje - anoda

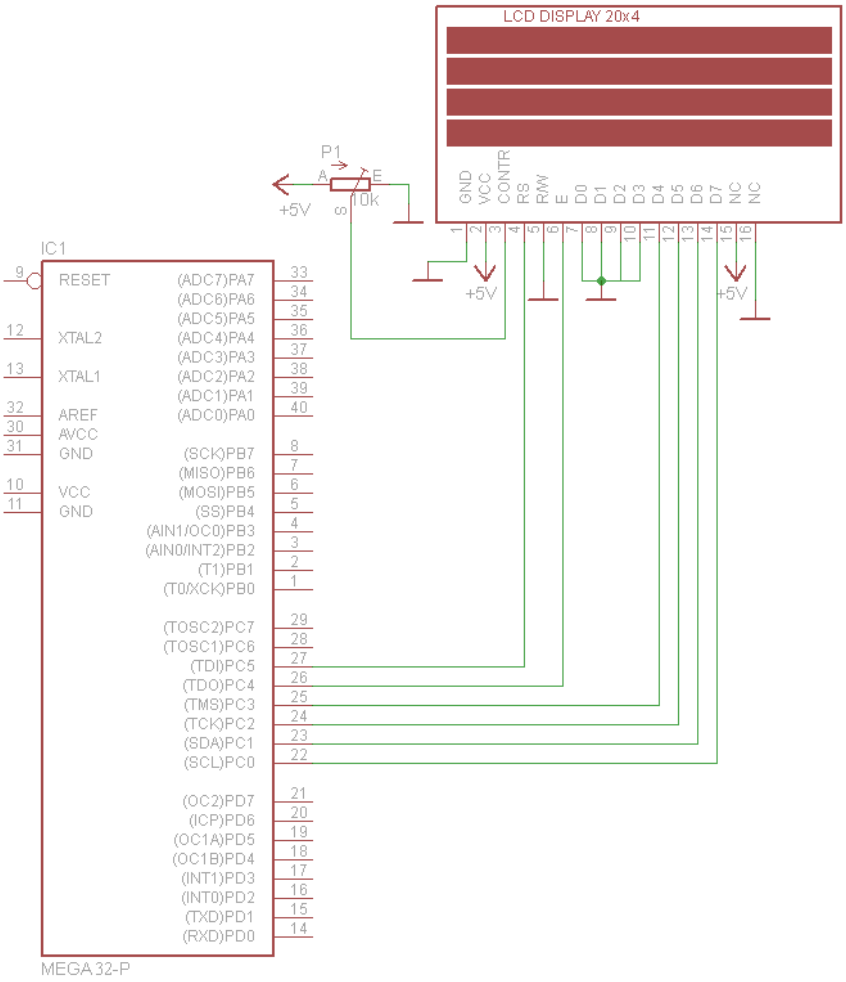
Tabulka 21 - Význam pinů u displeje

Existují dva režimy ovládání.:

- **8bitová komunikace** - k přenosu dat je použito všech 8 datových vodičů (DB0 - DB7). Komunikaci řídí piny RS, R/W a E. Dle hodnoty námi nastavené na pinu RS se určí, zda se budou posílat/číst data nebo instrukce. U pinu R/W se zvolí zápis nebo čtení z displeje - v našem případě je tento pin trvale spojen se zemí, neboť čtení z displeje nebudeme potřebovat. Následně se aktivuje vstup E (E=1) a na datové vodiče se přivedou data určená pro odeslání. Odeslání dat se potvrdí deaktivací vstupu E (E= 0).
- **4bitová komunikace** - v tomto režimu se pro přenos používají pouze čtyři datové piny (DB4 - DB7), ostatní je potřeba uzemnit. Po zapnutí displeje se provede inicializace 4bitové komunikace, následně se postupuje obdobně jako u 8bitové. Nastaví se RS, R/W máme trvale spojen se zemí (log. 0), aktivuje se vstup E (E=1) a na datové piny DB4 - DB7 se přivedou horní 4 bity dat, které se potvrdí deaktivací vstupu E (E=0), následně se úplně stejně zapíšou dolní 4 bity dat a také se potvrdí deaktivací vstupu E. Oproti 8bitové komunikaci se tedy data posílají nadvakrát. Výhodou je, že ušetříme 4 datové vodiče, nevýhodou je mírně složitější a déle trvající komunikace.



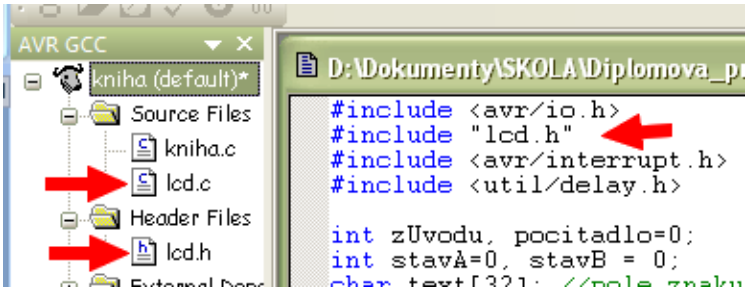
Obrázek 38 - Princip komunikace s displejem[8]



Obrázek 39 - Schéma zapojení displeje

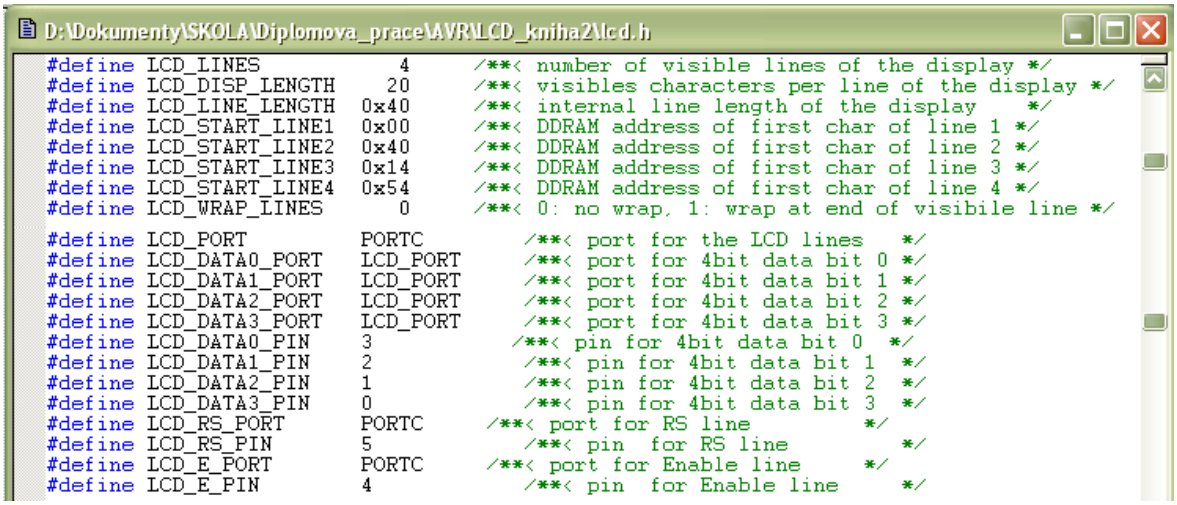
[8] PANDATRON. Ovládání znakových LCD s řadičem HD44780 – 1. díl

Knihovny potřebné pro práci s displejem je možné vytvořit si samostatně, nebo použít již napsané a volně přístupné. Pro urychlení práce jsem zvolil druhou variantu a použil knihovny dostupné z odkazu, uvedeném ve článku [6]. Knihovna z článku obsahuje dva soubory, *lcd.c* a *lcd.h*. Tyto soubory je potřeba zkopírovat do složky kde je vytvořený projekt, dále je nutné přidat je v AVR Studiu do vytvořeného projektu. To se provede tak, že se klikne pravým tlačítkem na položku v levém menu „*Source Files*“ -> „*Add existing source files*“ -> *lcd.c*. Dále se provedou obdobné kroky pro soubor *lcd.h*. Tedy: „*Header Files*“ -> „*Add existing source files*“ -> *lcd.h*. Následně se vloží hlavičkový soubor pomocí direktivy #include „lcd.h“ na začátek hlavního souboru *main.c*.



Obrázek 40 - AVR studio 4 - vložení knihovny pro LCD

Před používáním displeje je potřeba provést nastavení pro konkrétní displej a to v souboru *lcd.h* (Obrázek 40).



Obrázek 41 - Nastavení v lcd.h

Zde se nastaví počet řádků (4), počet znaků na řádek (20), nastavení DDRAM adres prvních znaků v jednotlivých řádcích. Tyto údaje se dají zjistit z datasheetu příslušného displeje [11]. Dále se nastaví piny, ke kterým je displej připojen.

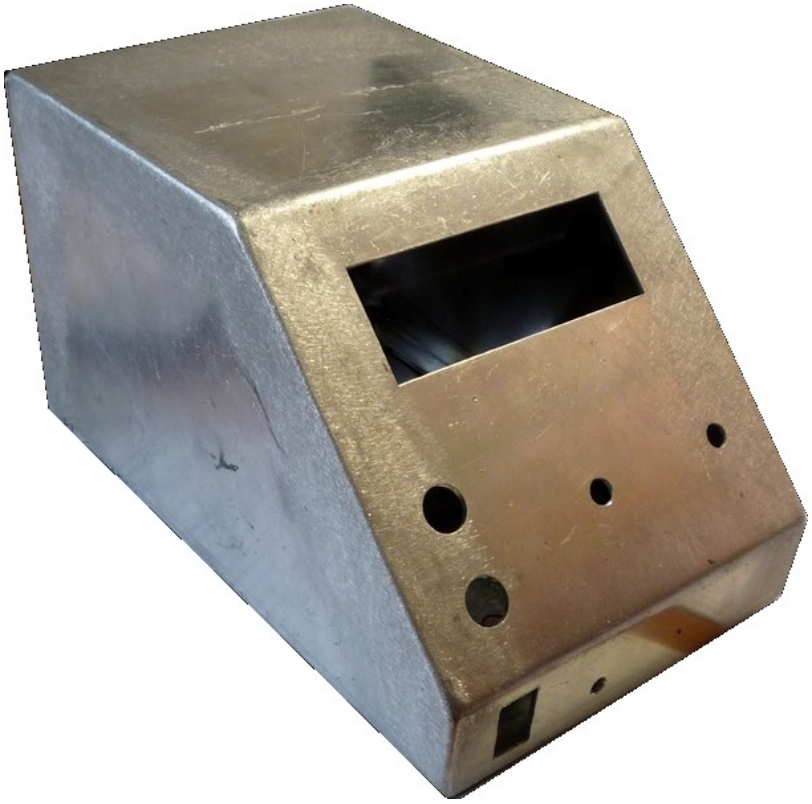
Knihovna obsahuje veškeré funkce potřené k ovládání LCD. Jsou jimi například

- *lcd_init(uint8_t dispAttr)* - inicializace displeje
- *lcd_clrscr(void)* - smazání obsahu displeje

- *lcd_gotoxy(uint8_t x, uint8_t y)* - posunutí kurzoru na uvedenou pozici. Řádky jsou označeny od 0 do 3 shora, podobně i znaky v řádcích jsou označeny od 0 do 19, zleva
- *lcd_putc(char c)* - zobrazení znaku
- *lcd_puts(const char *s)* - zobrazení řetězce znaků

2.9 Pouzdro

Jako každý výrobek, tak i tento si zaslouží být umístěn v nějaké vzhledově přijatelné krabičce jednak chránící elektroniku a zdraví obsluhy, ale také zajišťující pohodlnější ovládání. Materiál, pro stavbu byl zvolen hliníkový plech o síle 2mm. Z důvodu nevybavenosti domácí dílny byla oslovena firma *Medtec*, která zajistila výrobu krabičky dle dodaných výkresů.



Obrázek 42 - Krabička - mezistupeň výroby

Předběžný návrh a veškeré finální výkresy krabičky jsou provedeny v programu *ProfiCad 6.7.2*, který je volně dostupný pro nekomerční účely. Práce s tímto programem je velmi intuitivní a dá se bez potíží zvládnout i bez předchozích zkušeností. Program samozřejmě disponuje i exportem výkresu do formátu *png*.

Krabička je rozkreslena do 5 výkresů: čelní, vrchní, zadní, spodní a boční část. Výkresy jsou přiloženy v příloze. Čelní, horní a boční stěny jsou k sobě svařeny a tvoří jeden celek. Spodní a zadní díl se k celku přichycují devíti šrouby M4. Ke spodnímu dílu jsou šesti šrouby M3 uchyceny všechny plošné spoje tvořící řídící jednotku. Spodek krabičky je

samozřejmě opatřen nalepovacími gumovými nožičkami, aby se zamezilo poškrábání stolu vyčnívajícími šrouby držící plošné spoje. Rozložení prvků na čelní stěně je zřejmé z obrázku (Obrázek 43). Zadní stěna obsahuje konektor pro připojení k počítači, dále konektor připojující laboratorní model a jako poslední, konektor k napájecímu adaptéru. První dva zmíněné jsou zapájeny do plošného spoje umístěného přímo pod celou zadní stěnou. Napájecí konektor je matičkou uchycen jak za plošný spoj, tak za hliníkovou stěnu a slouží i pro pevnější spojení obojího.

Povrchová úprava byla původně zamýšlena eloxováním, ale po jeho aplikaci by byly vidět veškeré pozůstatky broušení. Z toho důvodu a dále i kvůli větší mechanické odolnosti byl zvolen černý polomatný komaxit. Lakování zajistila taktéž firma *Medtec*.

Obrázek 43 - Krabička

2.10 Ovládání řídící jednotky

2.10.1 Ovládací prvky

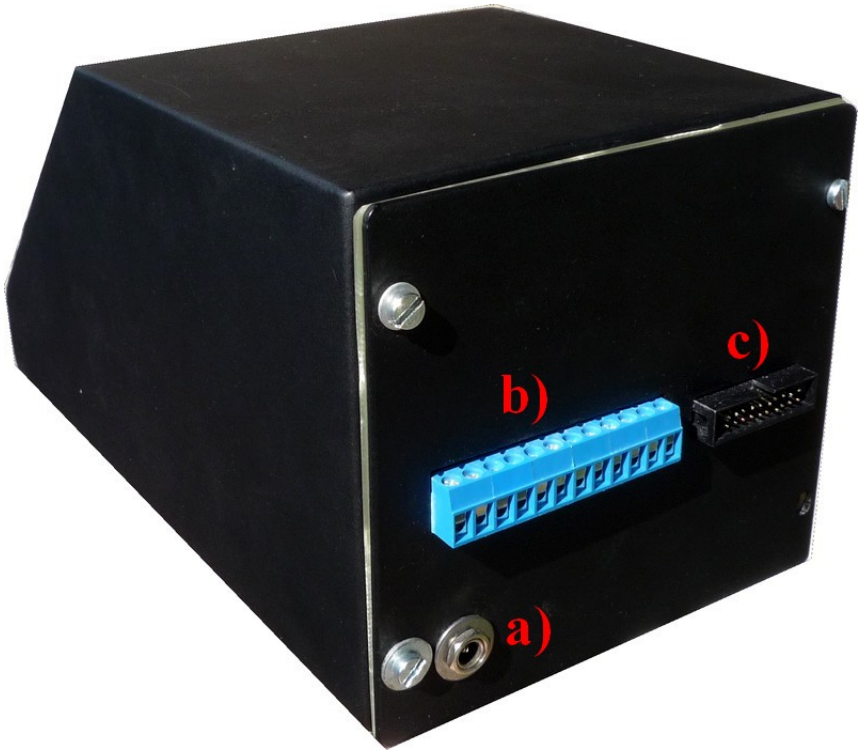
Ovládání je zajištěno třemi prvky. První je klasický kolébkový spínač (e) zapínající napájení. Vedle spínače je LED (f) signalizující zapnutí symetrického zdroje napětí - svítí zeleně, nebo případnou poruchu - svítí červeně. V levé části jsou umístěna dvě tlačítka (a, b). Tato tlačítka nakonec nebyla použita. Původně bylo zamýšleno využití při listování v menu a potvrzování hodnot, ale tuto funkci obstarávají rotační enkodéry (c, d) s vestavěnými axiálními spínači.

Obrázek 44 - Rozložení ovládacích prvků

IRC jsou výrobkem firmy Bourns a jejich označení je PEC11-4225F-S0024. Obsahují 5 konektorů. Dva jsou určeny pro vestavěný axiální spínač a další tři, umístěné na protější straně, k detekci směru otáčení. Princip funkce je obdobný jako v kapitole (2.6.2) s tím rozdílem, že zde jde o čistě mechanické spínání. Tedy oproti optickým enkodérům je zde kratší životnost a také větší nebezpečí vzniku zákmitů. Životnost je výrobcem udávána 30000 cyklů pro rotační snímač a 20000 cyklů pro axiální spínač.

2.10.2 Obsluha jednotky

Před zapnutím jednotky je nutno připojit napájecí adaptér a řízený systém přes konektory v zadní stěně, viz. obrázek (Obrázek 44). Po propojení a přepnutí kolébkového spínače do polohy „zapnuto“ se jednotka uvede do provozu.



Obrázek 45 - Rozmístění konektorů v zadní stěně

(a-napájecí konektor, b-propojení s PC, c-propojení s řízeným systémem)

V menu jsou na výběr dvě položky, mezi kterými se přechází otáčením pravého enkodéru na jednu, či druhou stranu. Výběr se provede stiskem pravého enkodéru, po kterém se dostaneme do příslušného submenu na jehož začátku je potřeba provést kalibraci snímačů polohy natočením a nakloněním hřídele do krajních, výchozích poloh, kde dojde k přerušení paprsku v optických závorách a tím vynulování obsahů čítačů mikroprocesoru pro zjišťování hodnot úhlů.

Obrázek 46 - Menu

Položka *„Rizeni s regulaci“* umožňuje ovládání jednotky s PSD regulátory. V tomto režimu se na LCD vypisuje jak pro V - vertikální motor (naklonění), tak pro H - horizontální motor (natočení) Y-regulovaná veličina (aktuální úhel), W- žádaná veličina

(nastavený požadovaný úhel), E - regulační odchylka a OT - otáčky motoru. Otáčení levým enkodérem se nastavuje úhel naklonění a otáčením pravým enkodérem úhel natočení. Návrat do hlavního menu se provede stiskem pravého enkodéru.

Obrázek 47 - První položka menu

Volbou druhé položky v menu je možné ovládat oba motory přímo, bez regulace. Otáčením enkodérů se nenastavuje žádaná hodnota - úhel naklonění, respektive natočení, ale přímo se jimi mění střída PWM signálů ovládající otáčky motorů. Po nastavení hodnoty a stisknutí levého enkodéru se hodnoty potvrdí a motory začínají pracovat. Tato volba je vhodná například pro měření statické charakteristiky hlavního motoru. Na LCD se zobrazuje hodnota proměnné OCR - pro změnu střídy a úhel naklonění jí odpovídající.

Obrázek 48 - Druhá položka menu

3 Experimentální část

3.1 Nastavení regulátorů

Kvalitní nastavení obou regulátorů by vystačilo na vlastní bakalářskou, nebo diplomovou práci. Bylo by vhodné sestavit model celého systému například v programu Matlab - Simulink a provést jednu z metod pro odhad parametrů. Z časových důvodů je od tohoto v práci upuštěno a parametry byly odhadnuty metodou *pokus/ omyl*.

Parametr	Hlavní motor	Ocasní motor
r_0	0,2745	0,2736
T_i	2,5	13,815
T_d	0,475	1,75

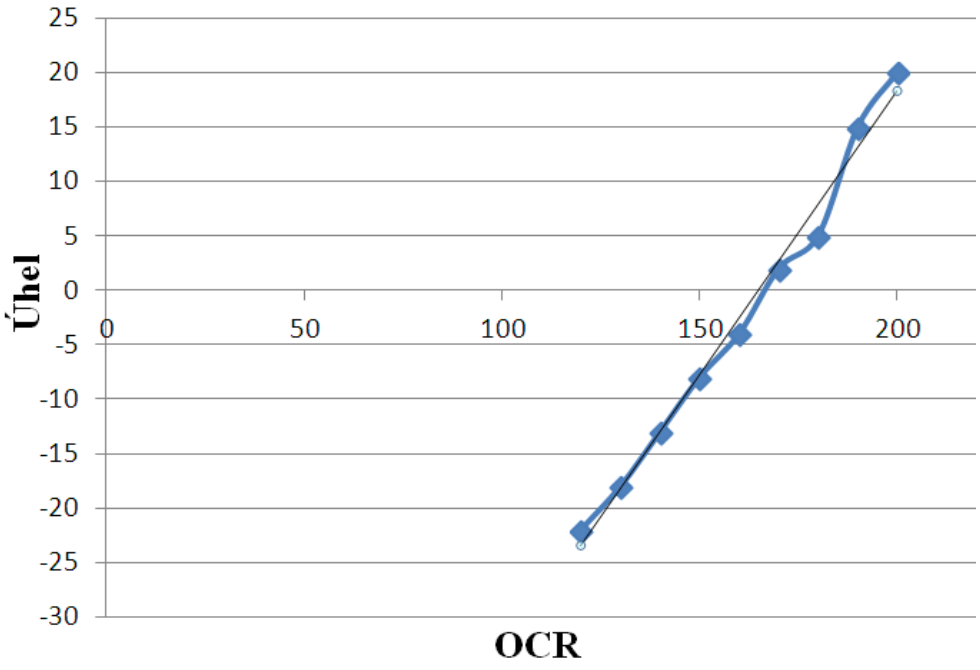
Obrázek 49 - Parametry PSD

Parametry, zvláště pro regulátor ocasního motoru nejsou odhadnuty příliš ideálně. Při větším akčním zásahu se soustava stává nestabilní a kmitá, ale pro demonstraci funkčnosti řídící jednotky je to dostatečné.

3.2 Naměřené charakteristiky

3.2.1 Statická charakteristika

Naměření statické charakteristiky je možné pouze u hlavního motoru, neboť jde o statickou soustavu. Ocasní motor je soustava astatická. Jde o závislost úhlu naklonění na hodnotě OCR (hodnota řídící střídu PWM signálu), tedy jaká hodnota OCR odpovídá příslušnému úhlu naklonění. Závislost je naměřena jen v pracovním režimu modelu.



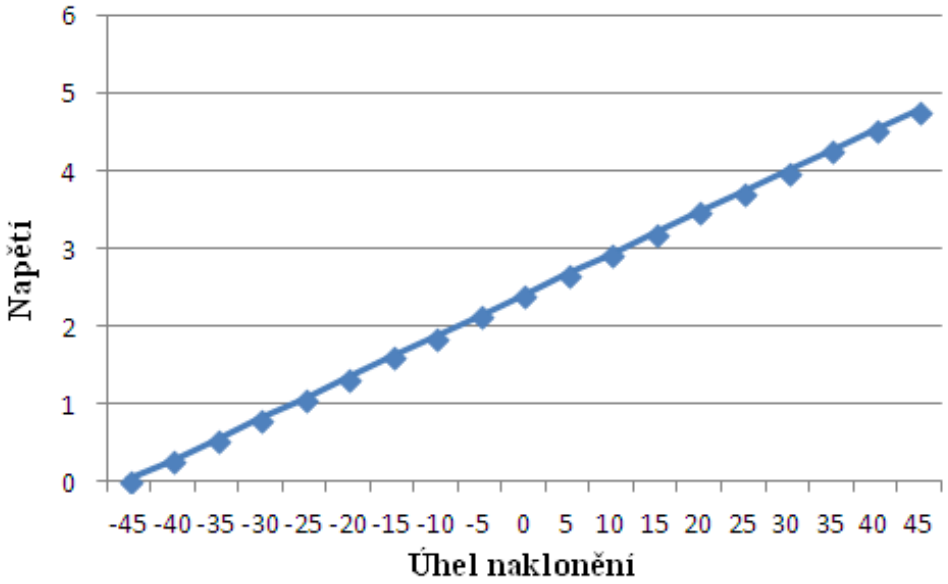
Obrázek 50 - Statická charakteristika

Rovnice regrese: $y = 0.5217 * x - 86.022$

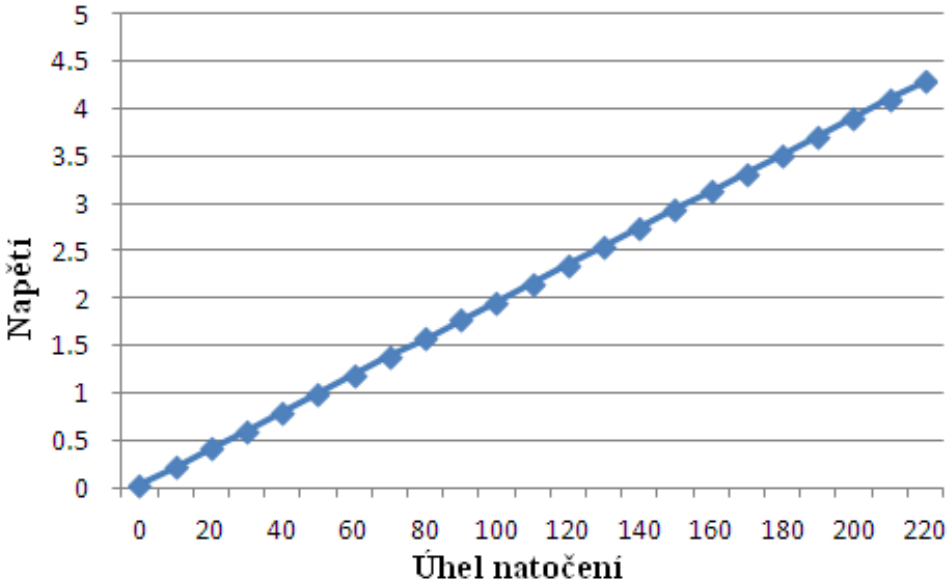
Hodnota spolehlivosti: $R^2 = 0.9883$

3.2.2 Kalibrační charakteristiky polohy

Kalibrační charakteristiky jsou vhodné sestrojit v případě, že je v plánu využívat PC jako řídící systém, nebo v něm jen zpracovávat výstupní informace. Jedná se o závislost úrovně stejnosměrného napětí na úhlu naklonění, respektive natočení. Toto napětí je měřeno na výstupu dolnofrekvenční propusti po převedení PWM signálu generovaného mikroprocesorem ATmega8. Střída PWM signálu je řízena počtem impulzů z optického rotačního enkodéru snímajícího úhel naklonění, respektive natočení.



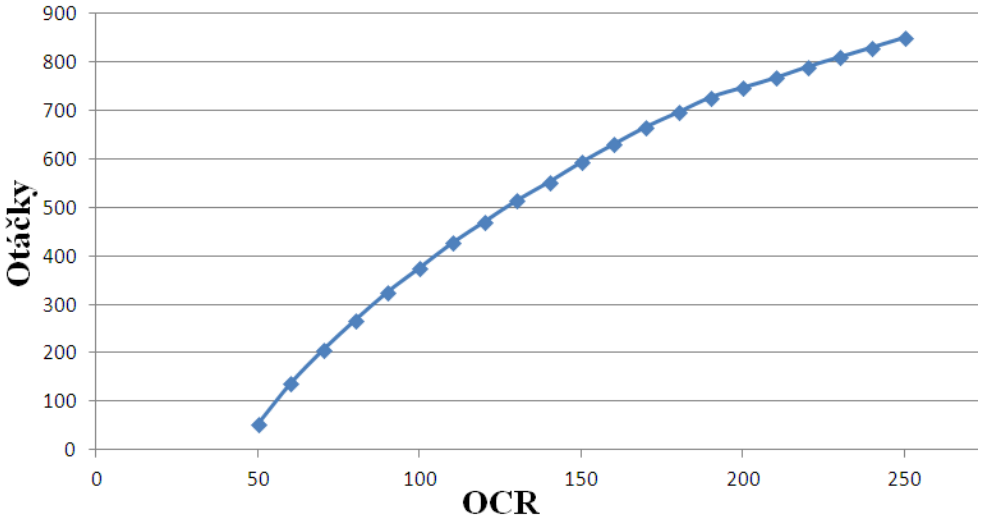
Obrázek 51 - Kalibrační char. - naklonění



Obrázek 52 - Kalibrační char. - natočení

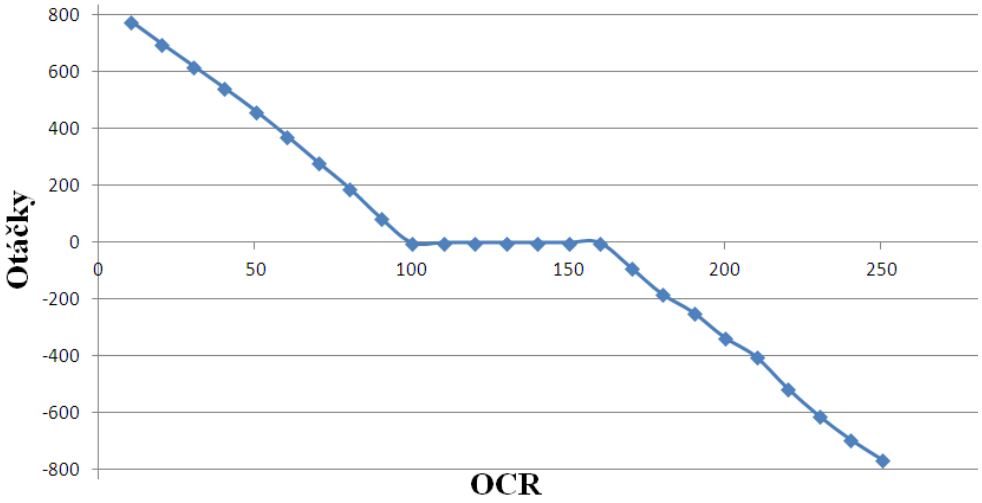
3.2.3 Kalibrační charakteristiky otáček motorů

Závislost otáček motorů na hodnotě OCR pro hlavní motor:



Obrázek 53 - Závislost otáček na OCR - hlavní motor

Pro ocasní motor:



Obrázek 54 - Závislost otáček na OCR - ocasní motor

4 Závěr

Cílem diplomové práce bylo navrhnout a vytvořit samostatnou mikroprocesorovou řídící jednotku, která by mohla být využívána studenty v univerzitních laboratořích pro studijní účely. V první části této práce je stručně popsán dvourotorový laboratorní systém, na němž se demonstruje funkčnost řídící jednotky. Tento systém je disertační prací pana Ing. Libora Havlíčka, Ph.D. v jehož práci [2] se lze dozvědět více informací o konstrukci a funkci.

Na základě teoretických poznatků získaných z dřívějšího studia jsem navrhnul blokové schéma řídící jednotky popsané ve druhé části dokumentace, za nímž následuje popis jednotlivých bloků, jejich schematické zapojení a taktéž jsou zde uvedeny jednotlivé plošné spoje. Veškeré plošné spoje byly pro jednoduchost a nižší náklady vyrobeny na jednostranných deskách s klasickými součástkami. Konec druhé části dokumentace je věnován popisu pouzdra, ovládacích prvků a vlastnímu ovládání. Parametry regulátorů a naměřené charakteristiky jsou uvedeny v části třetí.

V průběhu návrhu a výroby jednotky jsem se potýkal s celou řadou problémů. Časté problémy byly s kvalitou kupovaných nových součástek. Například nefunkční krystaly, rozpadnutí kolébkového spínače po zhruba patnácti sepnutí, nutností přeřezat jak vnější, tak i vnitřní závity u všech distančních sloupků, mizerná kvalita materiálu desek plošných spojů, nefunkčnost mikroprocesoru ATmega32, roztavení izolačních podložek tranzistorů. Dále jsem se potýkal s kolabováním komunikace mezi mikrokontrolérem a programátorem. Tento problém nakonec vyřešilo odebrání vazebních kondenzátorů u krystalů. Kolabování komunikace nenastávalo jen mezi programátorem a mikroprocesorem, ale i mezi oběma mikroprocesory. Důvod se mi nepodařilo zjistit, avšak při dokončování programu tento problém ustal. Nejspíše by se mohlo jednat o nějaký zdroj rušení, neboť celá jednotka není zabezpečena z hlediska EMC.

Řídící jednotka je sestavena z celkem deseti plošných spojů. K tomuto kroku jsem se rozhodl na začátku výroby jednotky, neboť jsem ještě neměl představu o finální podobě pouzdra. Rozdělení na více desek je na jednu stranu praktické, neboť v případě poruchy se dá příslušný plošný spoj vyjmout a přezkoušet mimo jednotku, nebo při úpravě zapojení se dá celý vyměnit. Na druhou stranu je to poměrně nevzhledné, neboť plošné spoje jsou mezi sebou pospojovány distančními sloupky a destičkami vyřezanými z kuprexitu. Navíc při manipulaci s celkem může docházet a i často docházelo k vytrhávání propojovacích vodičů.

Co se tedy týká možností budoucích úprav a vylepšení, tak by se určitě mohlo jednat o navržení zapojení na méně plošných spojů a zvolení lepšího způsobu propojení desek mezi sebou. Jistě i využití SMD součástek by mohlo být prospěšné k celkovému zmenšení, neboť při této realizaci váží jednotka bezmála 1,5 kilogramů a to bez napájecího adaptéru. Dále výměna, nebo celkové upuštění od ovládacích tlačítek, která i přes běžné způsoby ošetření zákmitů stále zakmitávala. Místo výměny by se mohlo ještě zkusit využít například klopných obvodů pro ošetření zákmitů. Po softwarové stránce by se jednalo o možnost zadávání parametrů regulátorů skrze ovládací prvky jednotky a automatickou

kalibraci roztočením ocasního rotoru jedním směrem pro dosažení jak vertikálního tak horizontálního čidla nulujícího úhel. V neposlední řadě by bylo vhodné navrhnout lepší parametry obou regulátorů, aby nedocházelo k rozkmitávání soustavy.

Součástí dokumentace diplomové práce je CD se zdrojovými kódy, schématy zapojení v souborech pro program Eagle, výkresy krabičky a katalogovými listy součástek.

5 Literatura

1. **POWIRSKI, Ireneusz.** *Przetwornica do CAR-AUDIO* [online]. 2004 [cit. 2013-03-31]. Dostupné z: http://sklep.avt.pl/photo/_pdf/AVT2732.pdf
2. **HAVLÍČEK, Libor**. *Modelování a řízení vícerozměrové soustavy* [online]. Pardubice, 2010 [cit. 2013-08-04]. Dostupné z: https://dspace.upce.cz/bitstream/10195/37306/1/HavlicekL_ModelovaniARizeni_2010.pdf. Disertační práce. Univerzita Pardubice.
3. **MCMANIS, Chuck**. H-Bridges: Theory and Practice. *Chuck's Robotics Notebook* [online]. 2006, 23.12.2006 [cit. 2013-08-05]. Dostupné z: http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/index.html
4. **PANDATRON.** Ovládání znakových LCD s řadičem HD44780 – 1. díl. *Pandatron - elektrotechnický magazín* [online]. 2009 [cit. 2013-08-05]. Dostupné z: http://pandatron.cz/?685&ovladani_znakovych_lcd_s_radicem_hd44780_%96_1._dil#comment
5. **BC. ZÁVODSKÝ, Ondrej**. Programujeme AVR v jazyku C - 6. časť. *Svět elektro* [online]. 2012, 2012-03-07 [cit. 2013-08-05]. Dostupné z: http://svetelektro.com/clanky/programujeme-avr-v-jazyku-c-6-cast-459.html
6. **BC. ZÁVODSKÝ, Ondrej**. Programujeme AVR v jazyku C. [online]. 2012, s. 68 [cit. 2013-08-05]. Dostupné z: http://svetelektro.com/Download/avr_kniha.pdf
7. **ATMEL. AVR221**: Discrete PID controller. *Atmel Corporation* [online]. 2006 [cit. 2013-08-10]. Dostupné z: http://www.atmel.com/images/doc2558.pdf
8. **Wikipedie**: Pulsně šířková modulace. [online]. 2013-03-10 [cit. 2013-08-26]. Dostupné z: http://cs.wikipedia.org/wiki/Pulzn%C4%9B_%C5%A1%C3%AD%C5%99kov%C3%A1_modulace
9. **Šulc, B. - Vítečková, M**.:Teorie a praxe návrhu regulačních obvodů. Vydavatelství ČVUT Praha, 2004.
10. **Matoušek David**: Práce s mikrokontroléry ATMEL AVR - 3.díl - edice uP a praxe 2. vydání, BEN - technická literatura, 2006, ISBN 80-7300-209-4.
11. **Matoušek David**: Práce s mikrokontroléry ATMEL ATmega16 - 4.díl - edice uP a praxe 1. vydání, BEN - technická literatura, 2006, ISBN 80-7300-174-8.
12. **Plíva Zdeněk**: Eagle Prakticky, BEN - technická literatura, 2010, ISBN 978-80-7300-252-7.
13. **Záhlava Vít**: Návrh a konstrukce DPS, BEN - technická literatura, 2010, ISBN 978-80-7300-266-4.
14. **Herout Pavel**: Učebnice jazyka C 6. vydání, Kopp, 2010, ISBN 978-80-7323-383-8.
15. **Herout Pavel**: Učebnice jazyka C - 2.díl 2. vydání, Kopp, 2006, ISBN 80-7232-221-4.
16. **Maťátko Jan**: Elektronika 6. vydání, Idea servis, 2005, ISBN 80-85970-49-X.

17. **ATmega32**. Katalogový list obvodu ATmega32 [online]. Dostupné z: http://www.atmel.com/Images/2503s.pdf
18. **ATmega8**. Katalogový list obvodu ATmega32 [online]. Dostupné z: http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf
19. **DEM 20486 SYH-LY**. Katalogový list LCD MODULU [online]. Dostupné z: http://www.e-lab.de/diverse/DEM20486.pdf
20. **LM7805.** Katalogový list lineárního regulátoru [online]. Dostupné z: https://www.sparkfun.com/datasheets/Components/LM7805.pdf
21. **LM2907**. Katalogový list převodníku F/U [online]. Dostupné z: http://www.ti.com/lit/ds/symlink/lm2907-n.pdf
22. **74AC14N**. Katalogový list [online]. Dostupné z: http://www.fairchildsemi.com/ds/74/74AC14.pdf

6 Příloha

6.1 Zdrojový kód mikroprocesoru ATmega32

```
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include "lcd.h"
#include <avr/interrupt.h>
#include <util/delay.h>
#include "uart.h"
#include <math.h>

#define BAUD  9600
#define MYUBRR F_CPU/16/BAUD-1

//globalni promenne
volatile int zUvodu, pocitadlo=0, pocitadlo1 = 0, povol = 0, povol1 = 0, xOCR=160, yOCR=0;
int stavA=0, stavB = 0;
char text[32]; //pole znaku pro vypis na displej

char prijmuto[3];
char prijmuto2[3];
//globalni promenne pro ADprevodnik
float realne;
//PID REGULATOR***************
double r0H = 0.2745;
double TiH = 2.5;
double TdH = 0.475;
double r0O = 0.2836;//0.0876;//1.50815;
double TiO = 9.21*1.5;//4.8;//1.8;
double TdO = 1.75;//1.95;//0.45;
double q0H=0, q1H=0, q2H=0;
double q0O=0, q1O=0, q2O=0;
double eH=0, eH1=0, eH2=0;
double otackyH1=0, otackyH2=0;
double otackyV1=0, otackyV2=0;
double vyslPIDh=0, vyslPIDhstare=0;
double vyslPIDo=0, vyslPIDhstareo=0;

double uhelOCR=0, uhelOCR2=0, uhelOCR1 = 0;
double Y=0, Y1=0, Y2 = 0;
double E1=0, E2 = 0;
//END PID REGULATOR************
//Funkce**************************
int tlacPotvrd();
int tlacPotvrd2();
void primeRizeni();
void neprimeRizeni();
void menu();
void nastavOCR();
void UARTinit(int ubbr); // inicializace seriove linky
void UARTposli(int data); // vysilani po seriove lince
char UARTprijmy(void); //prijem po seriove lince
void ADCinit();
float ADCcti(char kanal);
void PIDh(double uhel);
int PIDv(double uhel);
```

```
//END...Funkce**********
//*************************************************************************
//PID regulatory
//*************************************************************************
int PIDv(double uhel) //zadana uhel nakloneni -  hlavni motor - vertikalni rovina
{                               //prijde uhel natoceni
        double T;
        double noveOCR;
        int vysledne = 0;

        noveOCR = (double)yOCR;         //z enkoderu ZADANA
        //prepocet uhlu na hodnoty OCR
        uhelOCR = 1.8944*uhel+164.84; //REGULOVANA
        //y = 1.8944x + 164.84

        T=0.5;

        //PSD podle pana Duska
        q0H= r0H*(1+(T/(2*TiH))+(TdH/T)); //0.082
        q1H=-r0H*(1-(T/(2*TiH))+((2*TdH)/T));//-0.156
        q2H= r0H*TdH/T;         //0.0741
        vyslPIDh=vyslPIDhstare+(q0H+q1H+q2H)*noveOCR-q0H*uhelOCR-q1H*uhelOCR1-
q2H*uhelOCR2;

        vyslPIDhstare = vyslPIDh;
        uhelOCR1 = uhelOCR;
        uhelOCR2 = uhelOCR1;

        vysledne = (int)vyslPIDh;
        if(vysledne<0)
                vysledne=0;
        else if(vysledne>250)
                vysledne=250;

//ULOZIT DO OCR1A - pro hlavni rotor
        OCR1A = vysledne;
        return q0H;
}
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
void PIDh(double uhelo) //regulovana uhel natoceni - horizontalni rovina - ocasni motor
{                       //uhel 0 - 218
        double To;
        double W = 0;
        int vysledneo = 0;
        double E= 0; //REGULACNI ODCHYLKA

        Y = uhelo;                              //REGULOVANA                    0-220
        W = (double)xOCR;       //z enkoderu ZADANA     0-220
        E = W-Y; //REGULACNI ODCHYLKA - pokud se bude tocit na druhou stranu, tak prohodit Y-W

        To=0.5;

        q0O= r0O*(1+(To/(TiO))+(TdO/To)); //0.082
        q1O=-r0O*(1+((2*TdO)/To));//-0.156
        q2O= r0O*TdO/To;        //0.0741
        vyslPIDo=vyslPIDhstareo+q0O*E+q1O*E1+q2O*E2;

        E1 = E;
```

```
	E2 = E1;

	vyslPIDhstareo = vyslPIDo;
	Y1 = Y;
	Y2 = Y1;

	if(vyslPIDo >= 117)
		vyslPIDo = 117;
	if(vyslPIDo <= -117)
		vyslPIDo = -117;

	vysledneo = 148+(int)vyslPIDo;

	if(vysledneo<0)
		vysledneo=0;
	else if(vysledneo>250)
		vysledneo=250;

//ULOZIT DO OCR1A - pro hlavni rotor
	OCR1B = vysledneo;
}
//*****************************************************************************
//Rotacni enkodery
//*****************************************************************************
ISR (INT0_vect)  //ocasni rotor
{
	if ((PIND & (1<<PD2))==4 && (PINA & (1<<PA5))==32)
	{
		//rotacni enkoder - stisk - pro menu
		pocitadlo=pocitadlo+1;	//inc 16
		if(pocitadlo > 1) pocitadlo = 1;

		//neprime rizeni
		if(povol1 == 1){
			if(xOCR>0){	//odebirat lze do nuly
				xOCR -= 1; //ubrat stridu
			}
		}
		if(povol == 1){
			if(xOCR>0)xOCR -=1;
		}
	}
	if ((PIND & (1<<PD2))==4 && (PINA & (1<<PA5))==0)
	{
		//rotacni enkoder - stisk - pro menu
		pocitadlo=pocitadlo-1;	//dec 8
		if(pocitadlo < 0) pocitadlo = 0;

		//neprime rizeni
		if(povol1 == 1){
			if(xOCR<=220) {		//pridavat lze do 250
				xOCR += 1; //pridej stridu,
			}
		}
		if(povol == 1){
			if(xOCR<=250)xOCR +=1;
		}
	}
}
```

```
ISR (INT1_vect) //hlavni rotor
{
	if ((PIND & (1<<PD3))==8 && (PINA & (1<<PA4))==16)
	{
		if(povol1 == 1 || povol == 1){
			if(yOCR>0) yOCR -= 1; //ubrat stridu
			if(yOCR <= 45) yOCR = 45;
		}
	}
	if ((PIND & (1<<PD3))==8 && (PINA & (1<<PA4))==0)
	{
		if(povol1 == 1 || povol == 1){
			if(yOCR<=250) yOCR += 1; //pridej stridu,
		}
	}
}
//*****************************************************************************
//Hlavni funkce
//*****************************************************************************
int main(){
	//UART----------------------------
	UARTinit(MYUBRR);
	//PWM1----------------------------
	DDRD|= (1<<PD4); //nastaveni PD4/OC1B jako vystupni
	DDRD|= (1<<PD5); //nastaveni PD5/OC1A jako vystupni

	//Rezim 5 fast PWM 8bit  ****
	TCCR1A|=(1<<COM1A1)| (1<<COM1B1)|(1<<WGM10); //OC1A, OC1B
	//Rezim 7 fast PWM 10bit  ****
	//TCCR1A|=(1<<COM1A1)| (1<<COM1B1)|(1<<WGM10)|(1<<WGM11); //OC1A, OC1B

	//preddelicka 256
	TCCR1B = (1<<WGM12) | (1<<CS10);
	//TCCR1B|=(1<<CS12);
	//------------------------------

	//INT0 nabezna hrana na vstupu INT0/ INT1 generuje zadost o preruseni
	MCUCR|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);

	//povoleni preruseni od INT0 a INT1
	GICR |= (1<<INT0) | (1<<INT1);

	//Nastaveni kalibracniho bajtu internihoi RC osc ****
	//OSCCAL = 0xA5;

	//globalni povoleni preruseni
	sei();

	//inicializace Port A - vstup (tlacitka)
	DDRA=0x00;
	PORTA=0xff;

	lcd_init(LCD_DISP_ON); // inicializacia displeja

//Tachodynama*********************************************
	//DDRB|= (1<<PB6); //nastaveni PB6 jako vystupni
	//DDRB|= (1<<PB7); //nastaveni PB7 jako vystupni
//End...Tachodynama*****************************************
//Uvod*****************************************************
```

```
        menu();
//End...Uvod*********************************************
        while(1){
        }
        return 0;
}
//*****************************************************************************
//Nastaveni OCR
//*****************************************************************************
void nastavOCR()
{
        OCR1A = 45;
        OCR1B = 99;
        yOCR = 45;
        xOCR = 0;
}
//*****************************************************************************
//ADprevodnik
//*****************************************************************************
void ADCinit()
{
        ADMUX |= (1<<REFS0);
        ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADATE)| (1 <<ADPS2) | (1 <<ADPS1) | (1
<<ADPS0);
}
float ADCcti(char kanal)
{
        ADC = 0;

        if(kanal == 0){
                ADMUX |= (1<<MUX2)|(1<<MUX1)|(1<<MUX0);//1 na MUX0 je pin ADC7
        }
        if(kanal == 1){
                ADMUX |= (1<<MUX2)|(1<<MUX1);
                ADMUX &= ~(1<<MUX0);//0 na MUX0 je pin ADC6
        }
        ADCSRA|=(1<<ADSC); //start prevodu

   while(!(ADCSRA & (1<<ADIF)));

   ADCSRA|=(1<<ADIF);
   kanal=0;
        ADCSRA &= ~(1<<ADSC); //konec prevodu
   return ADC;
}
//*****************************************************************************
//USART
//*****************************************************************************
void UARTinit(int ubbr) // UART inicializace
{
      UBRRH = (int)(ubbr>>8);
      UBRRL = (int)ubbr;

      UCSRB |=(1<<RXEN)|(1<<TXEN); //vysilac, prijimac
      UCSRB &= ~(1<<RXCIE);
      UCSRC |= (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0);
}
```

```
void UARTposli(int data) //posilani UART
{
	while ( !( UCSRA & (1<<UDRE)) ); //je prazdny buffer?
	UDR=data;
}
char UARTprijmy(void) //prijem UART
{

	while((UCSRA &(1<<RXC)) == 0);
	return UDR;

}
//*****************************************************************************
//Menu
//*****************************************************************************
void primeRizeni()
{
		double pom=0;
		int vyslNaklon = 0;
		int stav;
		int natoceni = 0, nakloneni = 0;
		double naklon = 0, natoc = 0;
		int otackyH = 0, otackyV = 0;
		ADCinit();
		nastavOCR();	//Nastaveni pocatecnich stavu citacu - pro PWM
		lcd_init(LCD_DISP_ON);
		povol1 = 1;
		lcd_gotoxy(0, 0);sprintf(text, " | Y  | W  | E  | OT");lcd_puts(text);

		do
		{
		   lcd_gotoxy(16, 3);sprintf(text, "Zpet");lcd_puts(text);

//UART_____________________________________________________________
			UARTposli('1');
			for(int i=0;i<3;i++)
		{
			   prijmuto[i]=UARTprijmy();
		}
			sscanf(prijmuto, "%d", &natoceni);		//REGULOVANA
			lcd_gotoxy(0, 2);sprintf(text, "H:%d ", natoceni);lcd_puts(text);//natoceni

			UARTposli('2');
			for(int i=0;i<3;i++)
		{
			   prijmuto2[i]=UARTprijmy();
		}
			sscanf(prijmuto2, "%d", &nakloneni);		//REGULOVANA
			nakloneni = nakloneni -45; //aby se dosahlo vypisu -45 0 45
			lcd_gotoxy(0, 1);sprintf(text, "V:%d ", nakloneni);lcd_puts(text);//nakloneni
//endUART__________________________________________________________
//ADC-otacky-motoru_____________________________________________
			otackyH = ADCcti(0);
			otackyH=otackyH-30;
			if(otackyH <= 1)
				otackyH = 0;
			if(otackyH < 10){
				lcd_gotoxy(17, 1);sprintf(text, "%d  ", otackyH);lcd_puts(text); //Hlavni motor
			}
```

```
                if(otackyH < 100 && otackyH >= 10){
                        lcd_gotoxy(17, 1);sprintf(text, "%d ", otackyH);lcd_puts(text); //Hlavni motor
                }
                else{
                        lcd_gotoxy(17, 1);sprintf(text, "%d", otackyH);lcd_puts(text); //Hlavni motor
                }

                otackyV = ADCcti(1);
                otackyV=otackyV-30;
                if(otackyV <= 1)
                        otackyV = 0;
                if(otackyV < 100){
                        lcd_gotoxy(17, 2);sprintf(text, "%d ", otackyV);lcd_puts(text);//Ocasni V
                }
                else{
                        lcd_gotoxy(17, 2);sprintf(text, "%d", otackyV);lcd_puts(text);//Ocasni V
                }
//END ADC_____________________________________________________________
//PID_____________________________________________________________
        //HLAVNI ROTOR//////////////////////////////////
        naklon = (double)nakloneni;
        vyslNaklon = PIDv(naklon); //vertikalni rovina
        //y = 0.5217x - 86.022 //prepocet OCR jednotek pro vypis ve stupnich
        pom = (yOCR*0.5217-86.022);
        if(pom <= -45) pom = -45;
        lcd_gotoxy(6, 1);sprintf(text, " %d ", (int)pom);lcd_puts(text);//ZADANA - z enkoderu
        //lcd_gotoxy(6, 1);sprintf(text, " %d ", ((92*yOCR)/255)-45);lcd_puts(text);//ZADANA
        ////REGULACNI ODCH.
        lcd_gotoxy(12, 1);sprintf(text, "%d " , (int)pom-nakloneni);lcd_puts(text);
        //lcd_gotoxy(13, 1);sprintf(text, "%d " , ((((92*yOCR)/255)-45)-nakloneni));lcd_puts(text);

        //OCASNI ROTOR//////////////////////////////////
        natoc = (double)natoceni; //REGULOVANA
        PIDh(natoc);  //horizontalni rovina - RAGULOVANA
        lcd_gotoxy(6, 2);sprintf(text, " %d  ", (int)(xOCR));lcd_puts(text);//ZADANA - z enkoderu
        //REGULACNI ODCH.
        lcd_gotoxy(12, 2);sprintf(text, "%d ", (int)(xOCR)-natoceni);lcd_puts(text);
//END PID_____________________________________________________________
                stav = tlacPotvrd();
        }while(stav != 0);
        povol1 = 0;
        menu();
}
void neprimeRizeni()
{
        int stav, stav1;
        int natoceni = 0, nakloneni = 0;
        nastavOCR();      //Nastaveni pocatecnich stavu citacu - pro PWM
        lcd_init(LCD_DISP_ON);
        povol = 1;          //povoleni zmen promennych xOCR a yOCR
        //vypis na lcd
        lcd_gotoxy(0, 0);sprintf(text, "   OCR | Uhel");lcd_puts(text);
        lcd_gotoxy(0, 3);sprintf(text, "Ok");lcd_puts(text);
        lcd_gotoxy(16, 3);sprintf(text, "Zpet");lcd_puts(text);

        do{
                lcd_gotoxy(0, 3);sprintf(text, "Ok");lcd_puts(text);
                lcd_gotoxy(0, 1);sprintf(text, "V: %d ", yOCR);lcd_puts(text);
                lcd_gotoxy(0, 2);sprintf(text, "H: %d ", xOCR);lcd_puts(text);
```

```
//UART_____________________________________________________________
                UARTposli('1');  //pro natoceni - ocasni motor
                for(int i=0;i<3;i++){
                   prijmuto[i]=UARTprijmy();
                 }
                sscanf(prijmuto, "%d", &natoceni);          //REGULOVANA
                lcd_gotoxy(9, 2);sprintf(text, "%d   ", natoceni);lcd_puts(text);//natoceni
                //- - - - - - - - - - - - - - - - - - - - - -
                UARTposli('2');  //pro nakloneni - hlavni motor
                for(int i=0;i<3;i++){
                   prijmuto2[i]=UARTprijmy();
                }
                sscanf(prijmuto2, "%d", &nakloneni);        //REGULOVANA
                nakloneni = nakloneni -45; //aby se dosahlo vypisu -45 0 45
                lcd_gotoxy(9, 1);sprintf(text, "%d   ", nakloneni);lcd_puts(text);//nakloneni
//endUART__________________________________________________________
//ADC______________________________________________________________
/*              //vyvod PA6
                otackyH = ADCcti(0);
                otackyH=otackyH-30;
                if(otackyH < 0) otackyH = 0;
                lcd_gotoxy(7, 1);sprintf(text, " %d ", otackyH);lcd_puts(text); //Hlavni motor

                otackyV = ADCcti(1);
                otackyV=otackyV-30;
                if(otackyV < 0) otackyV = 0;
                lcd_gotoxy(7, 2);sprintf(text, " %d ", otackyV);lcd_puts(text);//Ocasni V
*/
//END ADC__________________________________________________________
                stav1 = tlacPotvrd2();
                if(stav1 == 0) //Po stisku OK se nastavene hodnoty zapisi do OCR1A/B a uz se to toci
                {
                        OCR1A = yOCR;
                        OCR1B = xOCR;
                        lcd_gotoxy(0, 3);sprintf(text, "  ");lcd_puts(text);
                }
                stav = tlacPotvrd();
        }while(stav != 0);          //Po stisku Zpet se vrati do menu
        povol = 0;          //zakazani zmen promennych xOCR a yOCR
        xOCR = yOCR = 0;        //vynulovani xOCR a yOCR
        nastavOCR();    //Nastaveni pocatecnich stavu citacu - pro PWM
        menu(); //navrat do menu
}
void menu()
{
        int stav, vyber = 0;
        nastavOCR();    //Nastaveni pocatecnich stavu citacu - pro PWM
        lcd_init(LCD_DISP_ON);

        lcd_gotoxy(1, 0);sprintf(text, "Vitejte v programu ");lcd_puts(text);
        lcd_gotoxy(0, 1);sprintf(text, "====================");lcd_puts(text);
        lcd_gotoxy(1, 2);sprintf(text, "Rizeni s regulaci");lcd_puts(text);
        lcd_gotoxy(1, 3);sprintf(text, "Staticka char. V");lcd_puts(text);
        do{
                if(pocitadlo == 0){
                        lcd_gotoxy(0, 2);sprintf(text, "*");lcd_puts(text);
                        lcd_gotoxy(0, 3);sprintf(text, " ");lcd_puts(text);
                        vyber = 0;
                }else{
```

```
                        lcd_gotoxy(0, 2);sprintf(text, " ");lcd_puts(text);
                        lcd_gotoxy(0, 3);sprintf(text, "*");lcd_puts(text);
                        vyber = 1;
                }

                stav = tlacPotvrd();
        }while(stav != 0);
        if(vyber == 0)
                primeRizeni();
        else
                neprimeRizeni();
}
//*****************************************************************************
//TLACITKA
//*****************************************************************************
int tlacPotvrd()  //pravy enkoder stisk
{
        int A=0, B = 0;
                do{
                        //precist stav pinu
                        A = (PINA & (1<<PA2));
                        //pockat
                        _delay_ms(100);
                        //precist znovu stav pinu
                        B = (PINA & (1<<PA2));
                        //porovnat
                }while(A!=B);
        return A; //stisk = 0, jinak = 4
}
int tlacPotvrd2()  //levy enkoder stisk
{
        int A=0, B = 0;
                do{
                        //precist stav pinu
                        A = (PINA & (1<<PA3));
                        //pockat
                        _delay_ms(100);
                        //precist znovu stav pinu
                        B = (PINA & (1<<PA3));
                        //porovnat
                }while(A!=B);
        return A; //stisk = 0, jinak = 4
}
//*****************************************************************************
```

6.2 Zdrojový kód mikroprocesoru ATmega8

```
#include <avr/io.h>
#define BAUD  9600
#define MYUBRR F_CPU/16/BAUD-1
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>

//GLOBALNI*PROMENNE**************************************************************
volatile int countH = 0, countV = 0;
//END*GLOBALNI*PROMENNE**********************************************************
//IRC****************************************************************************
```

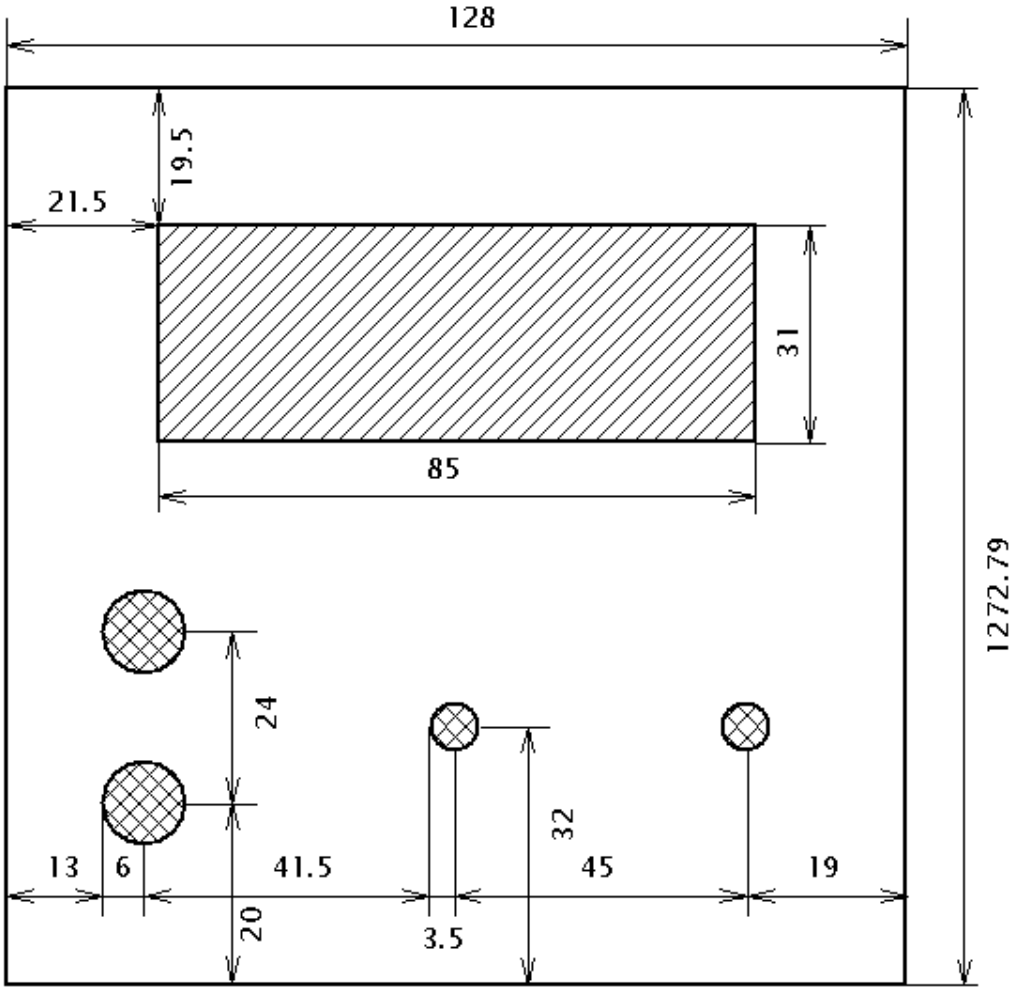
```
ISR (INT0_vect)  //horizontalni smer
{
	if ((PIND & (1<<PD2))==4 && (PINC & (1<<PC5))==32)
		countH += 1;
	if ((PIND & (1<<PD2))==0 && (PINC & (1<<PC5))==0)
		countH += 1;
	if ((PIND & (1<<PD2))==4 && (PINC & (1<<PC5))==0)
		countH -= 1;
	if ((PIND & (1<<PD2))==0 && (PINC & (1<<PC5))==32)
		countH -= 1;
}
ISR (INT1_vect) //vertikalni smer
{
	if ((PIND & (1<<PD3))==8 && (PINC & (1<<PC3))==8)
		countV -= 1;
	if ((PIND & (1<<PD3))==0 && (PINC & (1<<PC3))==0)
		countV -= 1;
	if ((PIND & (1<<PD3))==8 && (PINC & (1<<PC3))==0)
		countV += 1;
	if ((PIND & (1<<PD3))==0 && (PINC & (1<<PC3))==8)
		countV += 1;

}
//END*IRC*********************************************************************************
//UART************************************************************************************
void UARTinit(int ubbr){
   UBRRH = (int)(ubbr>>8);
   UBRRL = (int)ubbr;

   UCSRB =(1<<TXEN)|(1<<RXEN)|(0<<RXCIE);
   UCSRC |= (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0);
}
void UARTposli(int data ) {
   while ( !( UCSRA & (1<<UDRE)));
   UDR=data;
}
char UARTprijmy(void){
  while((UCSRA &(1<<RXC)) == 0);
  return UDR;
}
//END*UART********************************************************************************
//PWM*************************************************************************************

//END*PWM*********************************************************************************
//MAIN************************************************************************************
int main(void){
//PWM
	DDRB|= (1<<PB1); //OC1A jako vystupni
	DDRB|= (1<<PB2); //OC1B jako vystupni

	//Rezim 5 fast PWM 8bit  ****
	TCCR1A|=(1<<COM1A1)| (1<<COM1B1)|(1<<WGM10); //OC1A, OC1B

	TCCR1B = (1<<WGM12) | (1<<CS10);
	//preddelicka 8 při 16MHz (0.5us)
	//TCCR1B = (1<<WGM12) | (1<<CS11);
	OCR1A = 0;
	OCR1B = 0;
	//TCCR1B|=(1<<CS12);
```

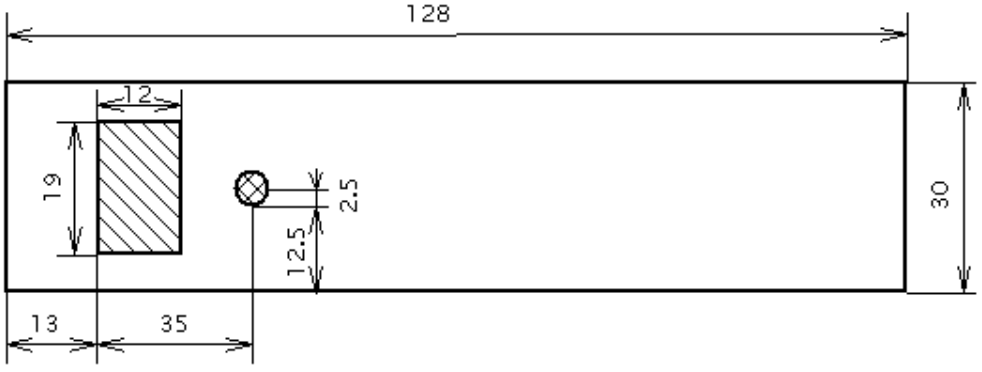
```
        //------------------------------
        //OSCCAL = 0xA5;
//END PWM

        //Preruseni kvuli IRC snimacum naklonu H a natoceni V
        //INT0 zmena log. stavu na vstupu INT0/ INT1 generuje zadost o preruseni
        //MCUCR|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);
        MCUCR|=(1<<ISC00)|(1<<ISC10);

        //povoleni preruseni od INT0 a INT1
        GICR |= (1<<INT0) | (1<<INT1);

        //globalni povoleni preruseni
        sei();

        UARTinit(MYUBRR);
        int i;

        char ctH[3], ctV[3];
        int uhelH=0, uhelV=0;
        int povol1 =0, povol2 = 0;

        while(1)
        {
                //opticke zavory - kalibrace
                if((PINC & (1<<PC4))==0){
                        countH = 0;
                        povol1 = 1;
                }
                if((PINC & (1<<PC2))==0){
                        countV = 0;
                        povol2 = 1;
                }
                //end opticke zavory - kalibrace

                //vynulovani pole
                for(int i = 0; i<3; i++)
                {
                        ctH[i]=0;
                        ctV[i]=0;
                }

                //pretypovani cisel na znaky
                sprintf(ctH, "%d", countH);
                sprintf(ctV, "%d", countV);

                //horizontal
                if(UARTprijmy() == '1'){
                   for(i=0;i<3;i++){
                           UARTposli(ctH[i]);
                   }
                }

                //vertical
                if(UARTprijmy() == '2'){
                   for(i=0;i<3;i++){
                           UARTposli(ctV[i]);
                   }
                }
```

```
                    //uhel - napeti
                    if(povol1 == 1 && povol2 == 1){
                            sscanf(ctH, "%d", &uhelH); //pretypovani na int, uhel ve stupnich
                            OCR1A = (255/218)*uhelH;  //zmena stridy, 255-obsah OCRx, 218-pocet
                            sscanf(ctV, "%d", &uhelV);
                            OCR1B = uhelV*255/93;
                    }
            }
 return 0;
}
```

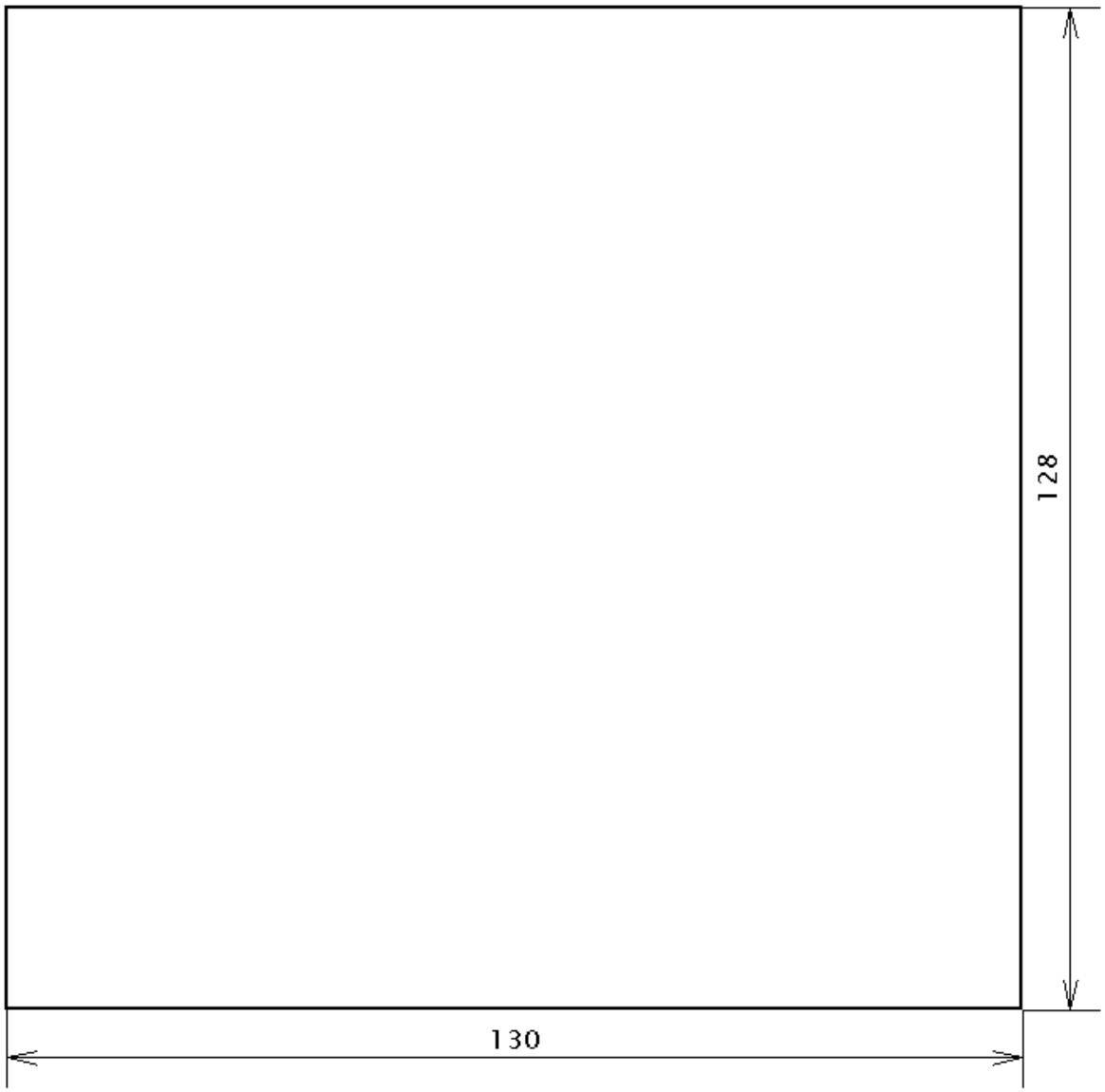
6.3 Výkresy

Čelo

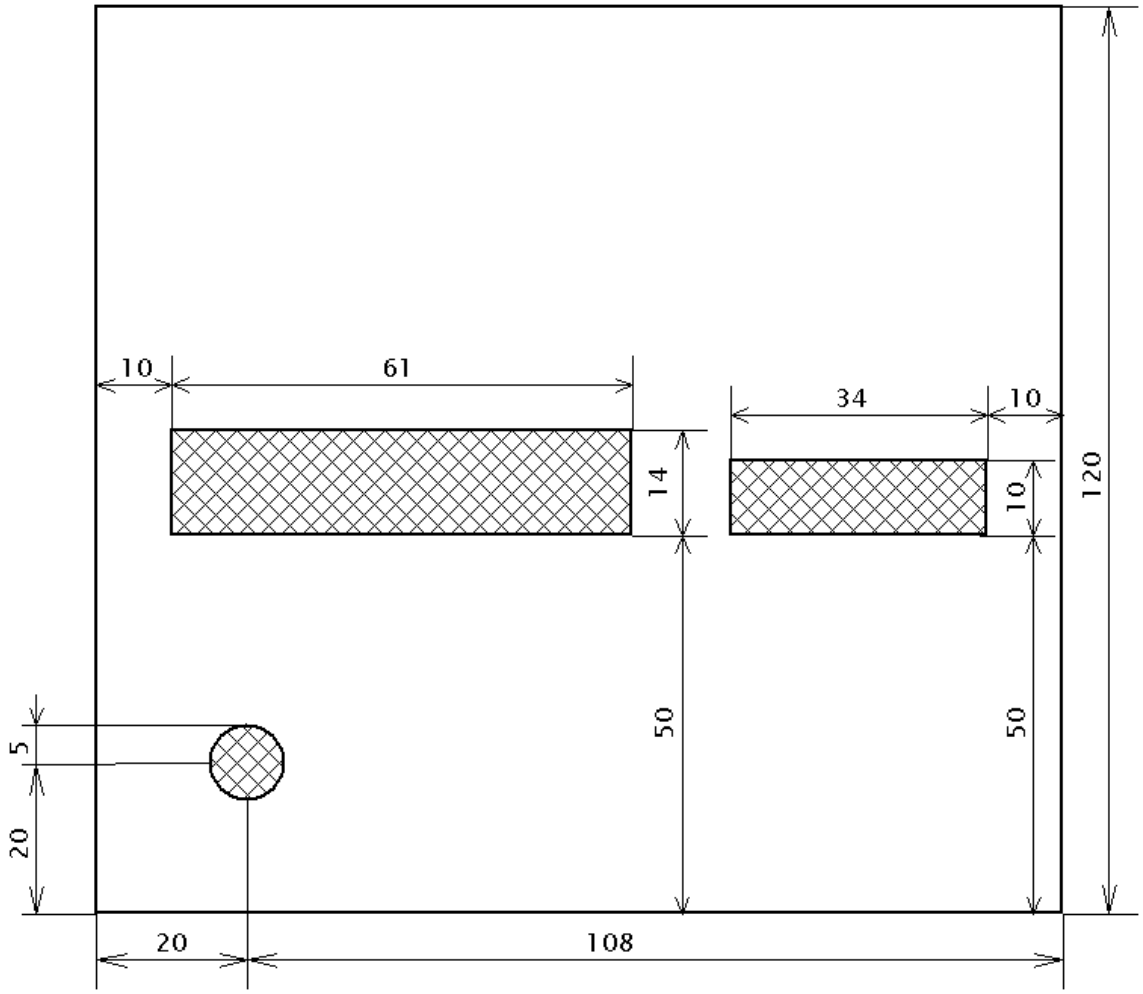




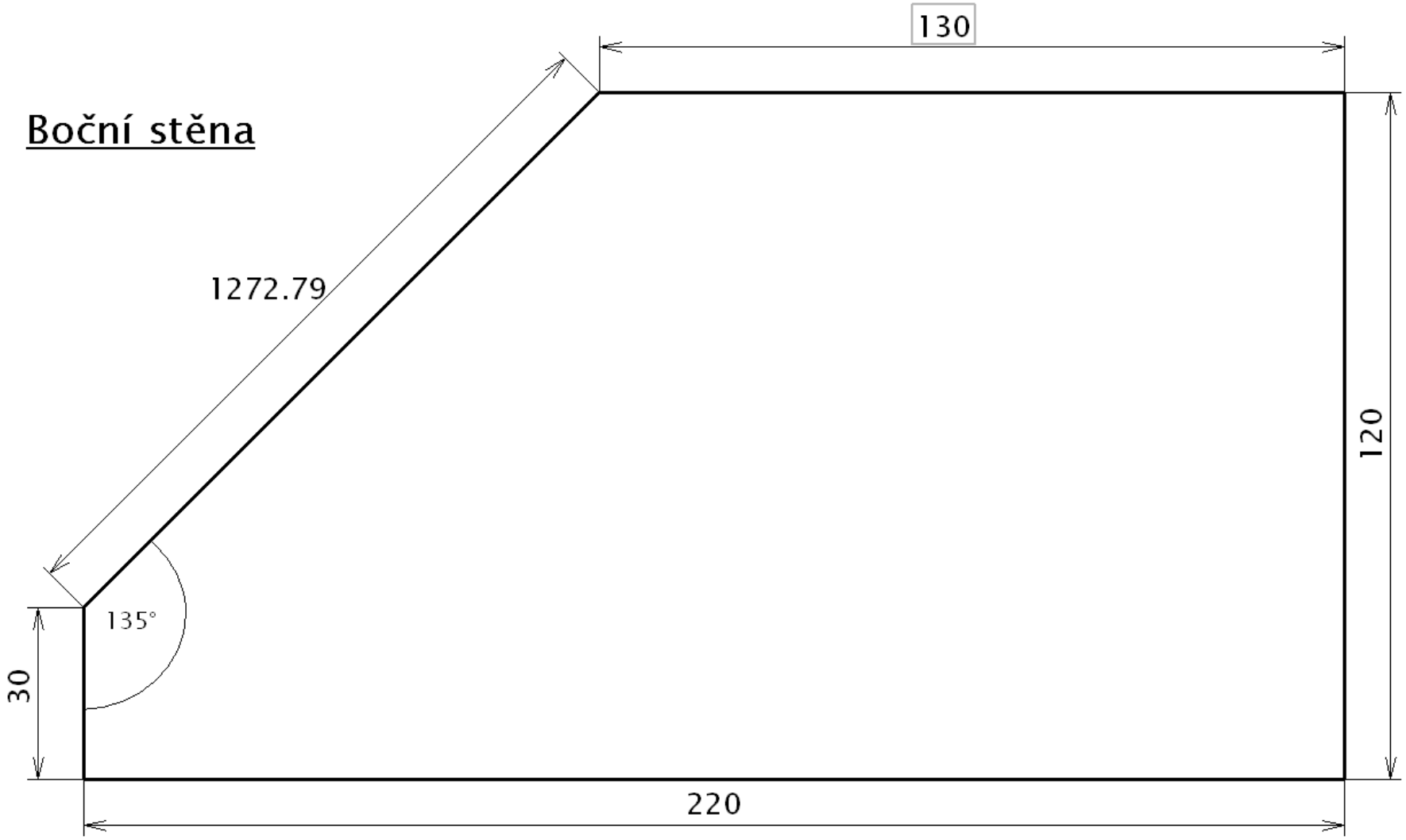
Horní část



Zadní část



Boční stěna



Spodní část Otvory o průměru 3mm