VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

AUTOMATICKÝ POSUV PRO DIGITÁLNÍ ZAPISOVAČ DAT

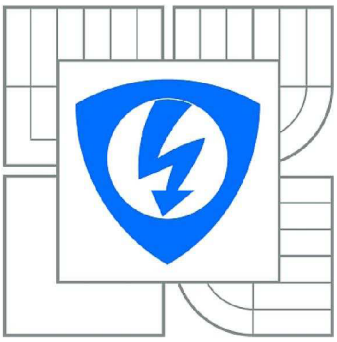
AUTOMATIC SHIFT FOR DIGITAL DATA RECORDER

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. Václav Šuta
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Diblík
SUPERVISOR

BRNO 2011



VYSOKÉ UCENÍ TECHNICKÉ V BRNE

Fakulta elektrotechniky a komunikacních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdelovací technika

Student: Bc. Václav Šuta **ID:** 77747

Rocník: 2 **Akademický rok:** 2010/2011

NÁZEV TÉMATU:

Automatický posuv pro digitální zapisovac dat

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možností konstrukce pocítacem ovládaného posuvu pro digitální zapisovač.

Navrhnete obvodovou strukturu kontroloru posuvu. Vyhledejte potrebné obvodové prvky a provedte merení jejich charakteristik. Realizujte desky plošných spoju a mechanickou cást zarízení.

Realizujte a oživte laboratorní prípravek. Realizujte software pro ovládáni posuvu pomocí PC. Provedte experimentální overení cinnosti.

DOPORUCENÁ LITERATURA:

[1] MATOUŠEK, D. C pro mikrokontroléry, 2. vydání. Praha: BEN – technická literatura, 2007.
[2] Firemní dokumentace výrobce integrovaných obvodu [online]. Dostupné na: http://www.freescale.cz

Termín zadání: 7.2.2011 **Termín odevzdání:** 20.5.2011

Vedoucí práce: Ing. Jan Diblík

prof. Dr. Ing. Zbynek Raida
Predseda oborové rady

Abstrakt

Diplomové práce Řízení Automatického posuvu pro digitální zapisovač dat se primárně zabývá návrhem a konstrukcí zařízení pro měření položířky laserového svazku. Úvodní část práce je věnována souvisejícím teoretickým základům a slouží pro lepší orientaci v následujcích kapitolách. Druhá část je potom věnována samotné konstrukci, osazení a oživení přípravku.

Abstract

Master theses of Automatic Shift Control of the Digital Data Recorder is primarily concerned to design and construction of equipment used for measuring the half-width of the laser beam. The introductory part is devoted to the related theoretical basics and serves for better orientation in the following chapters. The second part is devoted to construction, completion and basic setting of product.

Klíčová slova

laser, gaussovský svazek, automatický posuv, řízení motoru, L293, ATmega8

Keywords

laser, gaussian beem, automatic shift, motoru control, L293, ATmega8

ŠUTA, V. *Automatický posuv pro digitální zapisovač dat.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 53 s. Vedoucí diplomové práce Ing. Jan Diblík.

Prohlášení

Jako autor diplomové práce na téma Řízení Automatického posuvu pro digitální zapisovač dat" dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Janu Diblíkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....
podpis autora

Obsah:

1 Úvod

V zadání diplomové práce je uvedeno „*Řízení Automatického posuvu pro digitální zapisovač dat*“, jedná se o součást zařízení pro měření položířky laserového svazku. Tato práce se postupně zabývá stručnou teorií laseru, měřící metodou použitou pro měření položířky laserového svazku, návrhem konstrukce a nakonec samotnou konstrukcí a realizací potřebného mechanického a programového vybavení pro měření položířky laserového svazku.

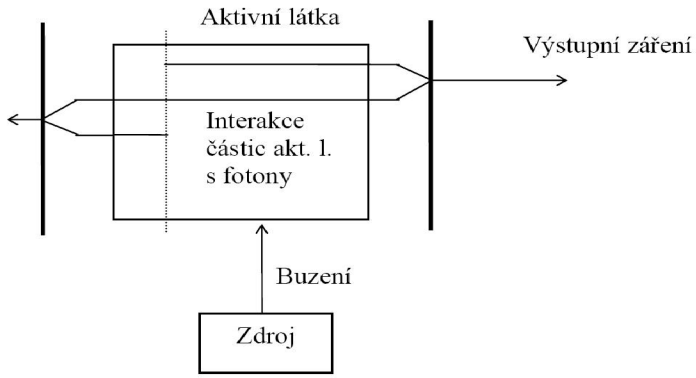
2 Teorie

Tato kapitola je věnována stručnému popisu Laseru, principu generování svazku, jednotlivým typům laserových svazků, měřící metodě pro měření pološirky laserového svazku a eventuálním možnostem realizace.

2.1 Laser

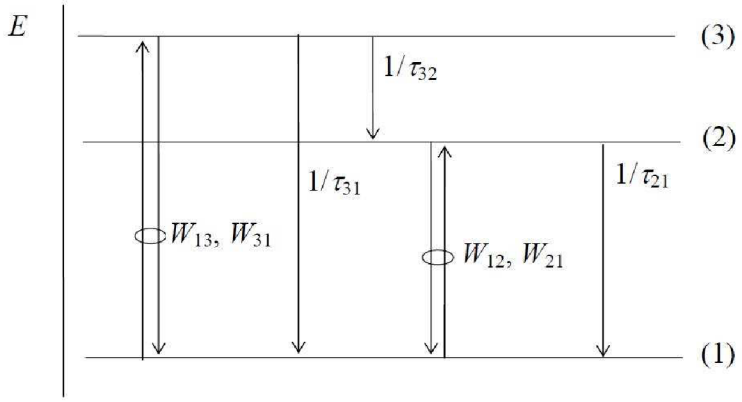
Laser je optický zdroj elektromagnetického záření, tj. světla. Samotné slovo LASER je zkratkou anglického výrazu „Light Amplification by Stimulated Emission of Radiation," což v překladu znamená zesilování světla stimulovanou emisí záření.

Laser je tvořen rezonátorem, aktivním prostředím a zdrojem energie. Viz obrázek níže:



Obr. 1: Rezonátor

Obecný princip laseru lze vysvětlit na rubínovém laseru. Zdrojem energie je v tomto případě výbojka, která do aktivního prostředí dodává energii ve formě záblesků, jenž energeticky vybudí elektrony aktivního prostředí ze základní energetické hladiny do vyšší energetické hladiny. Návratem do nižší energetické hladiny poté dojde k tzv. excitaci. Tímto způsobem je do vyšších energetických stavů vybuzeno velké množství elektronů aktivního prostředí a vzniká tak inverzní obsazení stavů. Tedy na vyšších energetických hladinách je více elektronů než na nižších.



Obr. 2: Rozložení energetických hladin

Kde W jsou energie a τ je doba života částice na dané hladině.

K inverznímu obsazení byly elektrony vybuzeny běžným světlem (tzn. světlo nebylo monochromatické). Při opětovném přestupu jednoho elektronu na nižší energetickou hladinu dojde k vyzáření (emisi) kvanta energie ve formě fotonu o příslušné vlnové délce a fázi. Tento foton následně prochází aktivním prostředím a interaguje s dalšími elektrony (sráží je na nižší energetickou hladinu) inverzního obsazení, čímž spouštějí tzv. stimulovanou emisi fotonů. Pokud foton prochází aktivním prostředím, fotony, které jsou vyzářeny sraženými elektrony, mají stejnou vlnovou délku a fází. Vzniká tedy monochromatické světlo.

Aktivní část laseru je umístěna uvnitř rezonátoru, tvořeného, v případě rubínového laseru, zrcadly. Díky zrcadlům dochází k odrazu svazku fotonů a jeho opětovnému průchodu prostředím, což dále podporuje stimulovanou emisi, a tím dochází k exponenciálnímu zesilování toku fotonů. Výsledný světelný svazek pak opouští rezonátor průchodem přes polopropustné zrcadlo rezonátoru.

Na rubínovém laseru byl vysvětlen obecný princip fungování laseru. Jeho konstrukce však může být provedena různými způsoby. Například jako zrcadla se používají obvykle dielektrická zrcadla, někdy se používá leštěný kov (zlato), v některých případech má dostatečnou odrazivost samotné rozhraní aktivního prostředí se vzduchem (laserová dioda). Zrcadla v rezonátoru nemusí být rovinná, naopak je v řadě případů výhodné použít kulová. Konkávní i konvexní zrcadla ovlivňují stabilitu záření v rezonátoru, tato stabilita je závislá na poloměrech křivosti zrcadel a jejich vzájemné vzdálenosti.

Existují určité lasery s dostatečně velkým ziskem v aktivním prostředí, které rezonátor nepotřebují a pracují superradiačně – záření tedy stačí jediný průchod k získání dostatečné intenzity. Patří mezi ně např. dusíkový nebo měděný laser.

Důležitým prvkem aktivního prostředí je jeho materiál, protože jde o látku obsahující oddělené kvantové energetické hladiny elektronů. Pokud zvolíme plyn nebo směs plynů, hovoříme o plynových laserech, pokud je zvolen monokrystal (kde hladiny vznikají dopováním prvkem s jinou vazností), jde o lasery pevnolátkové. Nejčastěji se však setkáme s tzv. diodovými lasery, kde je aktivní látkou polovodič s PN přechodem.

2.2 Typy laserových svazků

Laserový svazek nemusí mít běžně vygenerované gaussovské rozložení, existuje více typů rozložení.

2.2.1 Gaussovský svazek

Gaussova funkce popisuje rozložení intenzity průřezu Gaussova svazku, který je rozbíhavý a nemá kruhový průřez. Většina laserů generuje paprsek s Gaussovým rozložením, v takovém případě tvrdíme, že laser pracuje v základním příčném módu. Pokud projde Gaussův svazek čočkou, jedná se stále o svazek s gaussovským rozložením, ale už je charakterizován jinými parametry. Gaussův svazek je dobře popsán matematickým modelem.

2.2.2 Besselův svazek

Rozložení intenzity Besselova svazku je popsáno besselovou funkcí. Besselův svazek je nedifrakční a nerozbíhavý, což je základní rozdíl proti běžnému světlu. Vytvořit dokonalý Besselův svazek je nemožné, adekvátní se vytváří modelováním gaussovského svazku.

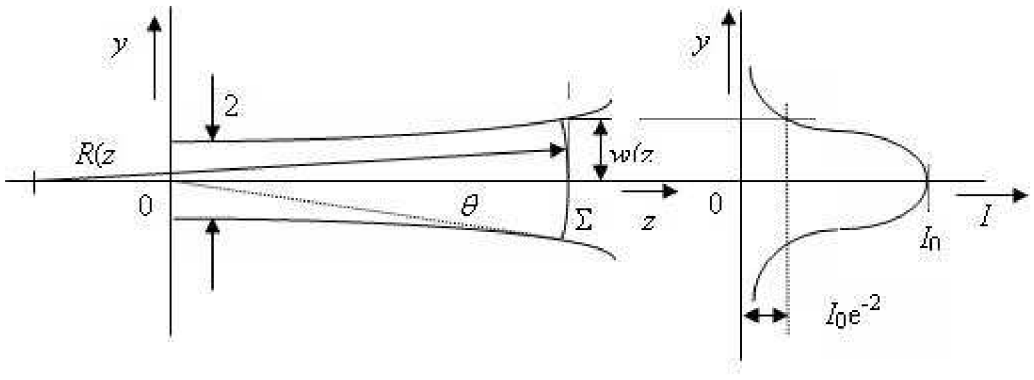
2.2.3 Tophat

Tophat je laserový svazek s téměř stejnou hustotou energie v celém kruhovém průřezu paprskem. Obvykle je vytvářen pomocí difrakčních optických prvků z Gaussova paprsku. Tyto typy paprsků se často používají v průmyslu, např. pro laserové vrtání otvorů do desek s plošnými spoji, či v laserových systémech s velmi vysokým výkonem, kde se pro zesilování používá soustava optických zesilovačů.

2.3 Měřící metoda

Výše uvedené informace tvoří teoretický základ, po kterém je možné se věnovat měřící metodě, jež je v této aplikaci používána.

Úloha, pro kterou bude zařízení používáno, měří svazek s gaussovským rozložením intenzity.



Obr. 3: Laserový svazek s gaussovským rozložením

Na obrázku výše je znázorněn profil symetrického Gaussova svazku s položířkou svazku $w(z)$, poloměrem křivosti $R(z)$, úhlem divergence θ a Rayleighovu vzdáleností z_0. Kde:

$$w(z) = w_0\left[1+\left(\frac{z}{z_0}\right)^2\right]^{1/2} \qquad R(z) = z\left[1+\left(\frac{z_0}{z}\right)^2\right] \qquad \theta = \frac{\lambda}{\pi w_0} \qquad z_0 = \frac{\pi w_0^2}{\lambda} \tag{1}$$

Za okraje svazku je považována taková vzdálenost w(z) od osy svazku, kde intenzita záření poklesne na hodnotu $I_0 e^{-2}$ vůči hodnotě intenzity I_0 na ose svazku.

Pokud je $z >> z_0$ je intenzita záření vyjádřena vztahem:

$$I_n \propto z_n^{-2} \exp\left[-2\left(\frac{\pi w_0}{\lambda}\right)^2\left(\frac{x_n}{z_n}\right)^2\right] \tag{2}$$

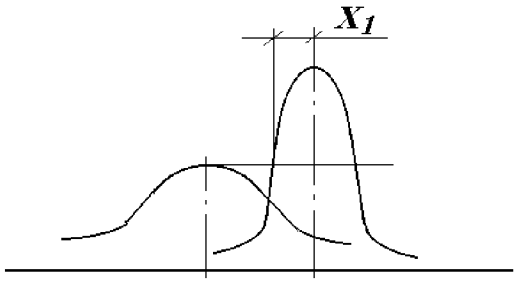
Měření je prováděno pro dvě hodnoty z, tedy z_1 a z_2. Jsou-li zvoleny hodnoty x_1, x_2 tak, aby platilo $I_1 = I_2$, dostaneme po dosazení:

$$1 = \frac{z_2^2}{z_1^2}\exp\left\{-2\left(\frac{\pi w_0}{\lambda}\right)^2\left[\left(\frac{x_1}{z_1}\right)^2 - \left(\frac{x_2}{z_2}\right)^2\right]\right\} \tag{3}$$

Další úprava vede ke zjednodušení. Je-li zvolena podmínka $I_1 = I_2$ tak, aby současně platilo $x_2 = 0$, potom platí:

$$w_0 = \frac{\lambda}{\pi}\left(\frac{z_1}{x_1}\right)\sqrt{\ln\frac{z_2}{z_1}} \tag{4}$$

Jedinou neznámou v rovnici zůstalo x_1, to odečteme z naměřených hodnot.

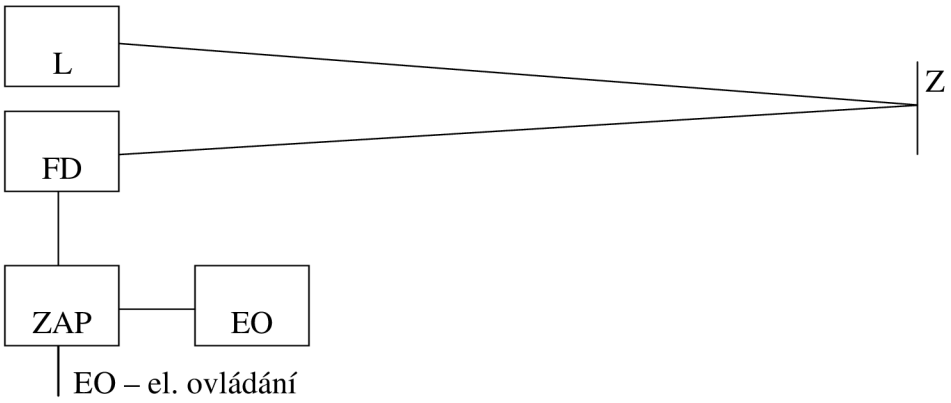


Obr. 4: Intenzity gaussovského svazku

2.4 Automatický posuv

Na základě uvedených informací o laseru, typech svazků, způsobech měření a počítání pološířky laserového svazku je možné se věnovat uspořádání, jež je pro tuto úlohu používáno a zařízení, které bude na jeho základě sestaveno.

Na obrázku je blokově naznačeno uspořádání pracoviště.



Obr. 5: Uspořádání pracoviště

Jako detektor intenzity je použita fotodioda (FD), která se přes průřez paprsku posouvá pomocí mikrometrického šroubu, což je právě část úlohy, kterou je nezbytné nahradit. V současnosti je nastavení posuvu a zápisu naměřených dat komplikované, čemuž chceme předejít automatizací a ovládáním pomocí počítače.

2.5 Možnosti realizace

2.5.1 Varianta s mikrometrickým šroubem

Existuje více možností realizace posuvu. Jednou z variant je použití mikrometrického šroubu připevněného na krokový motor. Tato varianta nebude při konstrukci použita, ale pro úplnost přehledu je vhodné ji uvést.

Při návrhu varianty s mikrometrickým šroubem bude postupováno následovně. Bude vybrán šroub s vhodným stoupáním, např. série High Resolution Lead-screws (stoupání 0,64 mm na závit, průměr 4,76 mm), dále motor s přiměřeným výkonem, např. malý bipolární krokový motor NEMA 8 (proud jednou fází je 0,6 A, krok má 1,8 stupně).

Pokud má šroub na 360° stoupání 0,64 mm, stoupání na minimální krok lze spočítat. Minimální krok ve stupních je 1,8°, lze tedy spočítat kolik kroků je třeba na celou obrátku a následně minimální vzdálenost posuvu.

$$\frac{360}{1{,}8} = 200 \quad \text{a} \quad \frac{0{,}64mm}{200} = 3{,}2\mu m\,.$$

Je-li minimální krok 3,2μm, lze tvrdit, že se jedná o dostatečně malou vzdálenost pro posuv a měření. Pokud je po každém kroku provedeno měření, bude dat pro zpracování příliš, taková přesnost je tedy považována za zbytečnou.

2.5.2 Varianta s kolejnicí

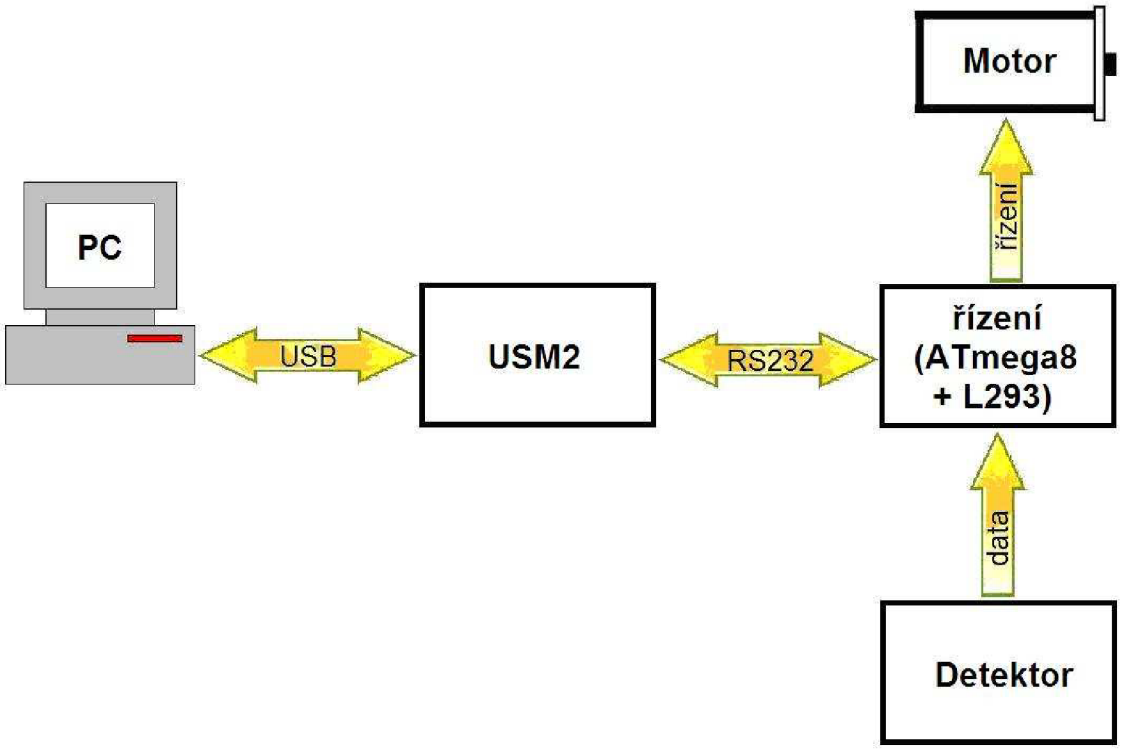
Nevhodnému řešení s mikrometrickým šroubem se lze vyhnout použitím již hotového krokového motoru s kolejnicí. Tento je možné obstarat v běžné inkoustové tiskárně.

V kapitole *2.3 Měřící metoda* se počítá se dvěma vzdálenostmi z, kde se vzdálenost z_1 pohybuje kolem 5,5 metrů a vzdálenost z_2 přibližně 12,14 metrů. Pro tyto vzdálenosti je šířka gaussovského svazku 3 cm a 10 cm, není tedy třeba posuv navrhovat pro větší šířky svazků.

Bližšímu rozboru návrhu varianty, kde bude použit krokový motor z tiskárny i s kolejnicí, bude věnována kapitola *4. Návrh zařízení*.

3 Návrh programu

Předběžný návrh zařízení předpokládá, že se projekt bude dělit na dvě základní části: programovou a mechanickou.



Obr. 6: Blokové schéma

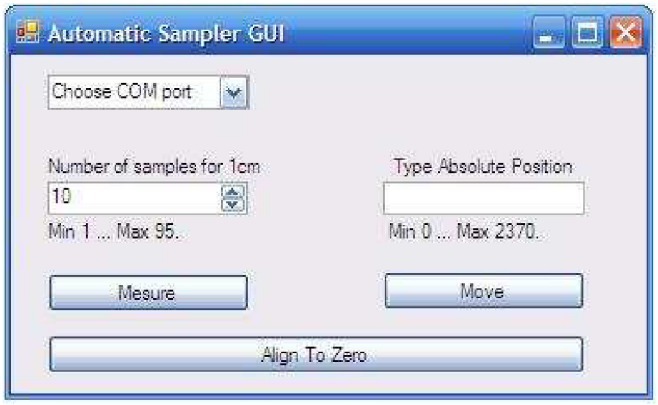
Do programové části spadá ovládací grafické uživatelské rozhraní (dále jen GUI) v počítači a program v mikroprocesoru. Do mechanické části patří torzo tiskárny, motor, snímače polohy, desky plošných spojů a kabeláž.

Na blokovém schématu výše jsou zobrazeny základní bloky návrhu. Počítač (dále jen PC) jako ovládací prvek je spojen s mikroprocesorem komunikační linkou. Protože většina PC dnes už nedisponuje sériovým portem, je potřeba vřadit nějaký konvertor, ku příkladu USM2. GUI poté používá virtuální sériový port.

Mikroprocesor ATmega8 ovládá díky jednotce USART sériovou komunikaci, ovšem nikoli na správných napěťových úrovních, je tedy nutné před procesor vřadit obvod MAX232, který převod napěťových úrovní zajišťuje.

3.1 Programové vybavení

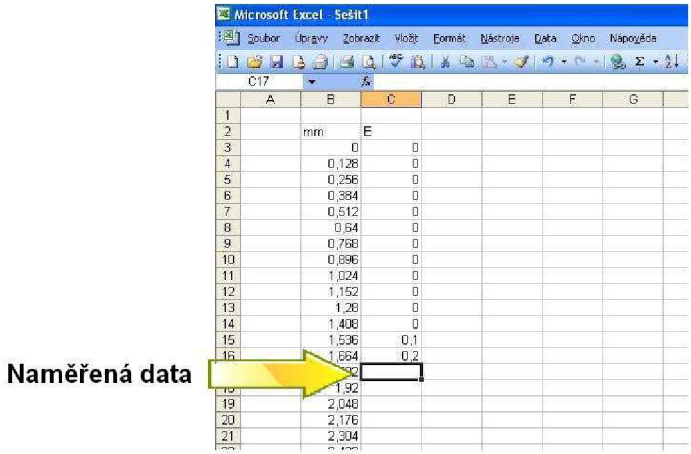
Celé zařízení je ovládáno z PC. GUI podporuje dva základní módy: měřící a posouvací. Tyto dva módy jsou doplněny o posuv na nulovou hodnotu (kalibrace).



Obr. 7: Ovládací rozhraní.

3.1.1 Měřící mód

Nejprve je nutné nastavit všechny vstupní parametry, poté stiskem tlačítka „*Mesure*“ zahájit samotné měření. Nastavené vstupní parametry jsou následně pomocí přenosové linky odeslány do mikroprocesoru, ten přijede na první pozici měření a podle parametrů z ovládací konzole začne měření provádět. Nato se detektor posune o stanovený počet kroků na další určenou pozici a změřenou hodnotu odešle za pomoci přenosové linky zpět do PC. Program paralelně běžící v PC tato data zachytává a ukládá do souboru v PC.



Obr. 8: Data importována do souboru v PC

3.1.2 Posouvací mód

V porovnání s měřícím módem neodebírá posouvací mód žádné vzorky z detektoru ani neposílá zpět žádné potvrzení o skončení přesunu. Poté, co je v grafickém rozhraní nastavena absolutní pozice a dojde ke stisknutí tlačítka „*Move*," je odeslána do mikroprocesoru instrukce kam má být vozíček s detektorem posunut.

3.1.3 Funkce „Align to zero"

Vozíček s detektorem přejede na začátek kolejnice a nastaví registr aktuální pozice na nulu. Začátkem je myšlen pravý okraj při pohledu zezadu.

Tuto funkci je vhodné použít vždy před začátkem každého měření.

3.1.4 Instrukce

GUI používá pro komunikaci pěti-bytové instrukce. První byte představuje jaký mód nebo funkce se bude provádět, následující čtyři byty představují měřítko. To znamená, po kolika krocích má byt odečten vzorek v měřícím módu a na jakou pozici má být detektor posunut při posuvném módu.

Obr. 9: Příklad instrukce

Pro posuv na pozici je používána instrukce „**w**xxxx", měření „**m**xxxx" a pro funkci Align to Zero „**a**yyyy." Za *x* dosadíme hodnotu, na kterou si přejeme se posunout popř. po kolika krocích vzorkovat a za *y* cokoli – procesor hodnotu znaků *y* ignoruje, ale je potřeba vždy odesílat celou pěti-bytovou instrukci, aby byl dodržen její rámec.

3.1.5 Jiné možnosti ovládání

Jako ovládací program lze použít prostředí určené pro řízení a ovládání laboratorních experimentů LabVIEW.

LabVIEW je moderní programovací vývojové prostředí, jež obsahuje mnoho nástrojů pro analýzu měřených dat a je určenou k vytváření programu ve formě blokových diagramů. Měřená data LabVIEW přímá přes sériový a paralelní port a s využitím měřicích karet a různých měřicích přístrojů. LabVIEW nabízí i síťovou komunikaci pomocí TCP/IP a spoustu dalších možností. Obsahuje propracované uživatelské rozhraní, kde lze s využitím

připravených prvků (např. tlačítek, snímačů atd.) připravit měřicí pracoviště ovládané počítačem přesně podle potřeb konkrétního projektu.

4 Návrh zařízení

Návrh zařízení se v průběhu konstrukce reálného zařízení mění podle aktuálních potřeb (doladění a oživení zařízení).

4.1 Výběr tiskárny

Kritéria pro výběr tiskárny byla jednoznačná a prostá: inkoustová nebo jehličková tiskárna, nejnižší cena. Tyto dva typy tiskáren využívají pro tisk posuvnou hlavici, což bylo důvodem jejich výběru.

Finální volbu výrazně usnadnil fakt, že mi byla použitá inkoustová tiskárna HP 930C přenechána bez jakýchkoli finančních požadavků stran původního majitele.

Obr. 10: Inkoustová tiskárna HP 930C

Na uvedeném obrázku č. 10 je ilustrační fotografie tiskárny HP 930C. Pro tento projekt ze stroje potřebujeme pouze kolejnici a motor.

4.2 Motor

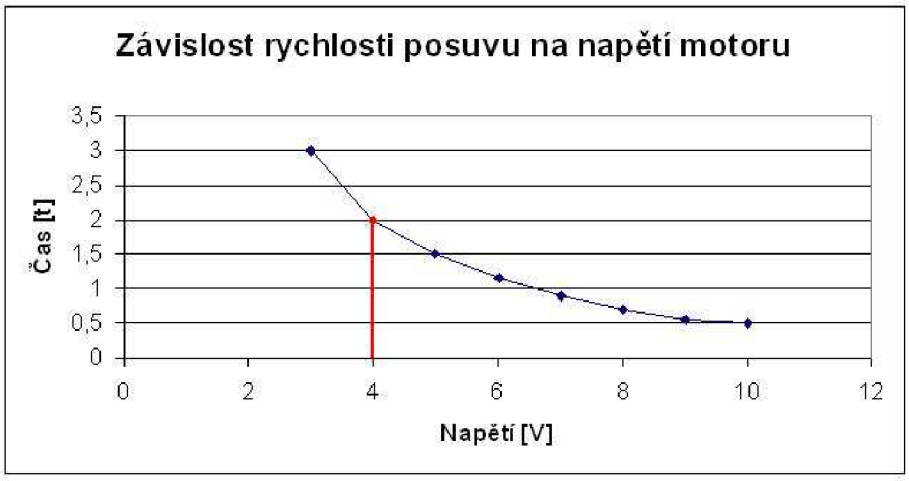
Vzhledem k teoretickému návrhu nastala první odlišnost ihned po rozebrání krytu tiskárny: na motoru byly pouze dvě svorky a podle označení se jednalo o motor stejnosměrný

Parametry motoru:

Technické parametry	
Maximální napětí	24 V
Pracovní napětí	15 V
Maximální proud	1,6 A
Maximální výkon	8,27 W
Minimální napětí	3 V
Odpor	11,6 Ω
Napětí při 1000 otáčkách	6,4 V
Moment na hřídeli	6,08 N*cm/A
Maximální moment	9,4 N*cm
Motor naprázdno	
Proud	0,1 A
Otáčky	3540 rpm

Tab. 1: Parametry motoru

Pro zvolení vhodné rychlosti posuvu vozíčku, je třeba vědět, jak rychle se bude pohybovat v závislosti na napětí na motoru. V dílně bylo změřeno, jakému napětí odpovídá který čas. Při vynesení naměřených dat do níže uvedeného grafu, bylo zjištěno, že jde o exponenciální závislost. Navzdory faktu, že motor pracuje do napětí 24V, stačilo měřit pouze do 10V, protože pro vyšší napětí už byl posuv tak rychlý, že jej nebylo možné měřit.



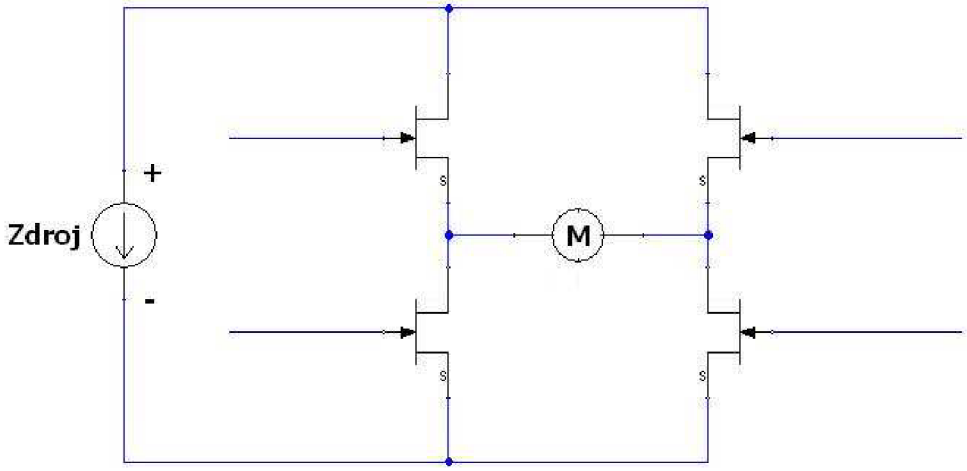
Obr. 11: Závislost rychlosti posuvu motoru na středním napětí na motoru.

Rychlost posuvu vozíčku závisí na napětí na motoru, aby toto nebylo nutné snižovat pomocí regulačního obvodu, použijeme pulzně šířkovou modulaci (dále jen pod angl. zkratkou PWM). Směr otáčení motoru lze řídit polaritou.

4.2.1 Řízení motoru

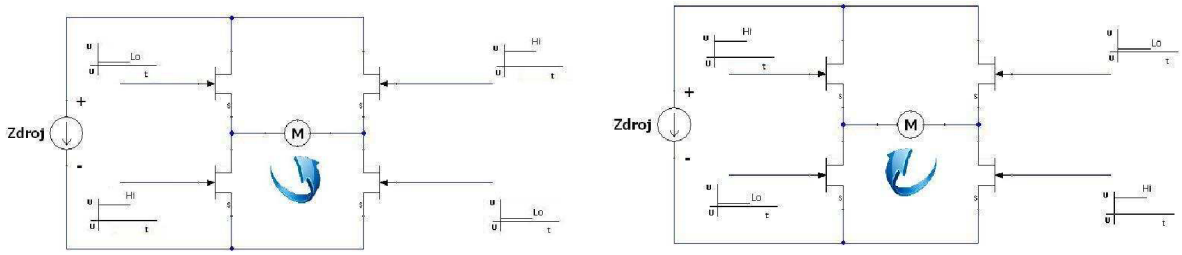
4.2.1.1 Řízení motoru pomocí tranzistorů

Aby bylo možno řídit stejnosměrný motor, je třeba jej podle toho zapojit. Schéma je naznačeno na obrázku 12.



Obr. 12: Principální znázornění zapojení motoru.

Na obrázku 12 jsou znázorněny 4 tranzistory zapojené do „*H - můstku*“, v příčce je zapojen motor. Toto zapojení umožňuje měnit polaritu na svorkách motoru a tedy měnit i směr jeho otáčení resp. směr posuvu.



Obr. 13a, b: Nastavení hradel tranzistorů.

Na obrázku 13a a 13b je předvedeno, jakým způsobem se mění řídící signály, podle toho se motor otáčí doprava nebo doleva. Pokud přivedeme na řídící radlo (gate) všech spínacích tranzistorů úroveň LO (nízká úroveň), motor se samovolně zastaví, nikoli však okamžitě.

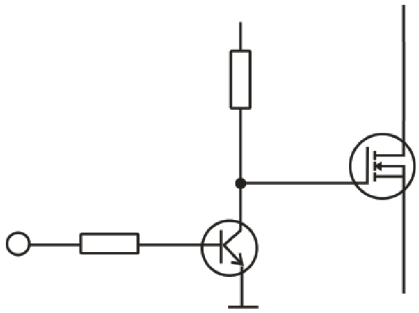
Pokud je motor zapnutý, může odebírat až 1,6A, což mu výstupní piny procesoru nemohou dodat, z tohoto důvodu jsou do obvodu vřazeny výše znázorněné spínací tranzistory. Zvolen byl tedy spínací tranzistor, který se řídí napětím, aby proudově nezatěžoval piny

procesoru a s dostatečnou proudovou rezervou, aby se zbytečně nezahříval. Z katalogu byl vzhledem ke stanoveným podmínkám vybrán tranzistor IRF530 typu FET, v pouzdře TO220.

Typ	MOS-N
Max. napětí	100V
Max. proud	14A
Odpor kanálu	0.16R
Max. výkon	88W
Pouzdro	TO220

Tab. 2: Parametry tranzistoru.

Po zvážení parametrů tranzistoru IRF530 je zřejmé, že dostačuje pro svůj účel, ale nelze jej řídit přímo pomocí výstupních pinů procesoru. Z tohoto důvodu je nezbytné vybranému FET tranzistoru předřadit jiný tranzistor, který je možné ovládat procesorem. Na obrázku 14 je schematický naznačeno jak toho lze dosáhnout.

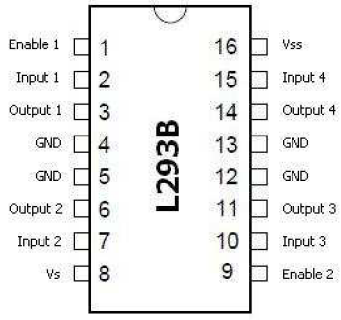


Obr. 14: Princip spínání hradla.

Tento jednoduchý spínací stupeň je závislý na parametrech FETu. Aby byl obvod kompletní, je nutné k FETu ještě paralelně připojit diodu – motor je induktivní zátěž a mohl by tranzistor zničit.

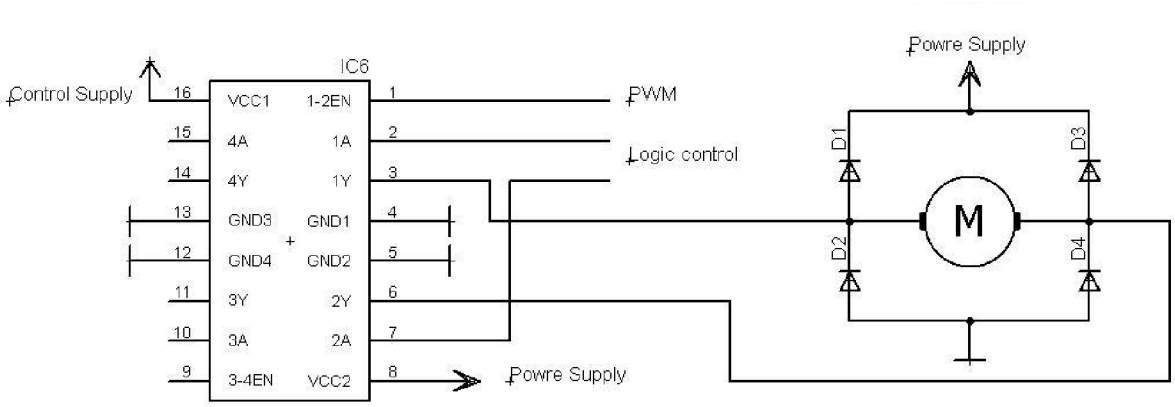
4.2.1.2 Řízení motoru pomocí obvodu L293

Obvod L293B je určen k řízení bipolárního krokového motoru nebo stejnosměrného motoru s řízením směru otáčení.



Obr. 15: Rozložení pinů L293B.

H-můstek je při řízení motoru plně nahrazen obvodem L293, jež je nadto vybaven pinem „*Enable*“. Je-li na zmíněný pin Enable přivedeno PWM, lze jím řídit otáčky motoru. Dále obvod využívá dvojího napájení: pracovní (do 36V) a logické (do 7V).

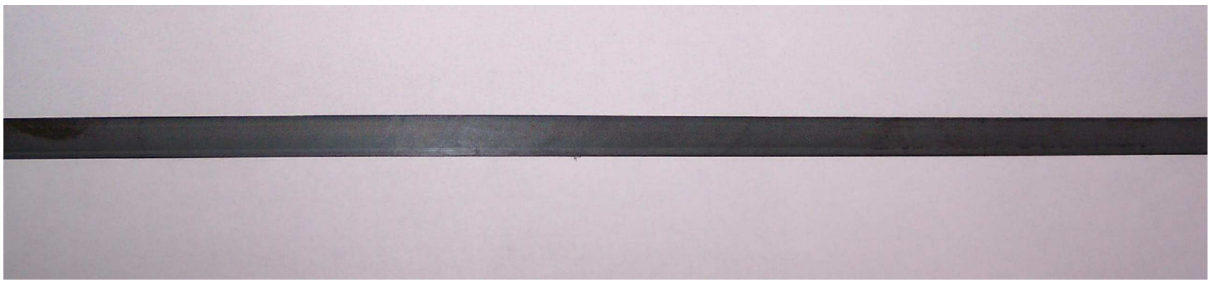


Obr. 16: Zapojení L293B pro řízení ss motoru.

4.3 Určení polohy

Obvodem L293 byl vozíček rozpohybován na obě strany, pomocí PWM je ovládána rychlost. Namísto krokového motoru je však v práci užit motor stejnosměrný, což vyvolává otázku, jakým způsobem bude určována poloha při měření?

Aby bylo možné v tiskárně určit polohu vozíčku, byl mezi koncovými body kolejnice napnut plastový proužek s jemným dělením. To bylo provedeno natisknutím tenkých čar na plastový proužek.



Obr. 17: Plastový proužek s jemným čárováním.

Na vozíčku byla umístěna optická závora, která sledovala pravidelné přerušování paprsku způsobované výše zmíněným plastovým proužkem. Tímto způsobem byla určována poloha detektoru.

Při pohybu vozíčku je možné sledovat signál z optické závory a vyhodnocovat jej v procesoru. Výstup si lze představit jako posloupnost po sobě jdoucích logických 1 a 0. V případě, že se vozíček dostane na konec kolejnice, se objeví pouze vysoká úroveň (1), což je způsobeno průhledným koncem plastového proužku. Ilustrace této optické závory není k dispozici, neboť byla během testů v laboratoři zničena ještě před pořízením příslušné fotodokumentace. Ve stejném okamžiku došlo též k přetržení plastového pásku, z toho důvodu je pro určování polohy třeba použít jiný způsob.

Jedním z navržených způsobů řešení je připojit na osu motoru výceotáčkový potenciometr a určovat polohu absolutně. Procesor ATmega8 má k dispozici osmi kanálový deseti bitový AD převodník, což dává 1024 hodnot na 290 mm a tedy:

$$d = \frac{l}{n} = \frac{290mm}{1024} = 0{,}283mm$$

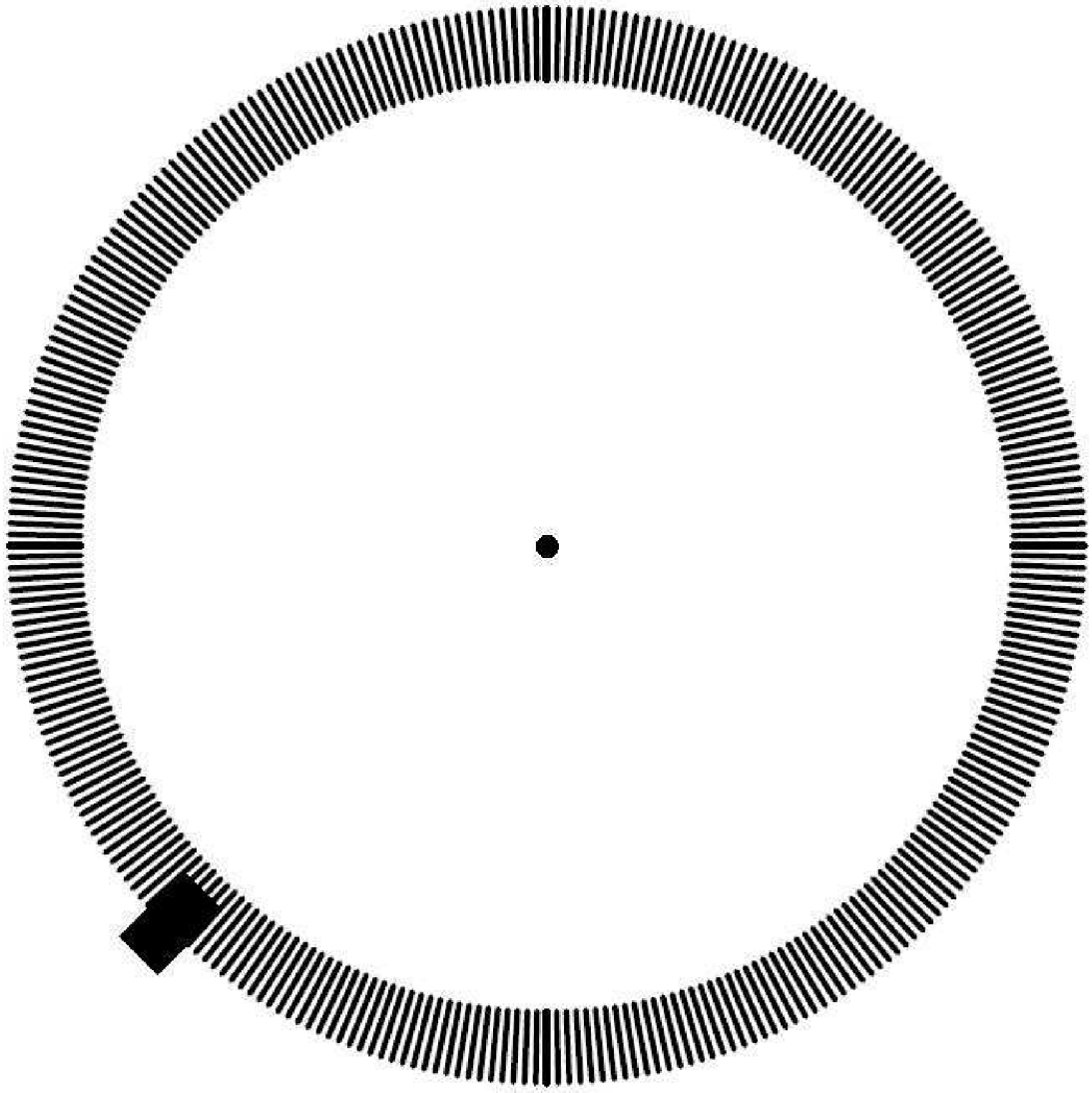
kde d je jeden krok, l je celá dráha a n rozsah převodníku. Tedy jeden krok d je 0,283 mm.

Vhodný výceotáčkový potenciometr je PM 534 100K, který je deseti otáčkový. Osa motoru se na celé dráze otočí devětkrát, potenciometru tedy zůstane ještě rezerva. Pro analogově-digitální převod (dále jen AD) lze použít vnitřní referenci 5V nebo externí napěťovou referenci, např. obvod LT1021. LT1021 je přesný referenční obvod na principu zenerových diod a dosahuje přesnosti 0,2%.

Tento způsob určování polohy je finančně velice nákladný (vysoká cena napěťové reference a výceotáčkového potenciometru). Současně by bylo potřeba provádět dva AD převody, jeden pro určení polohy a druhý pro zjištění intenzity laserového paprsku. Uvedený problém by se případně dal vyřešit připojením externího rychlého AD převodníku.

Pro určení polohy vozíčku je vhodné použít metodu bezkontaktní, protože potenciometr by častým používáním degradoval. Bezkontaktní metodou může být použití indukčního snímače, jehož nevýhoda je obdobná jako v případě potenciometru – pořizovací náklady se pohybují v řádech tisíců korun. Současně vyžaduje velkou konstrukční přesnost.

Posledním navrhovaným řešením je užití optické závory a kódového kolečka. Přesností a rozlišením kódového kolečka je dána i přesnost a četnost možných měření.



Obr. 18: Kódové kolečko (naznačena poloha optické závory).

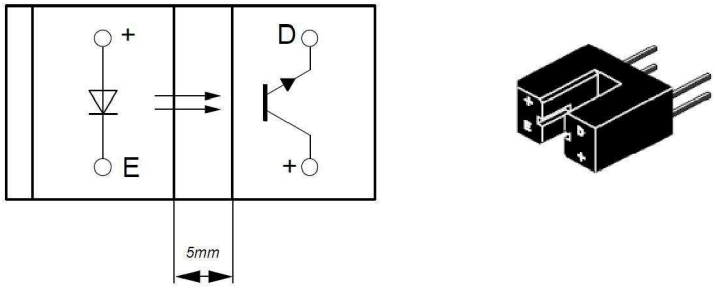
Na kolečku je 300 čar. Na 290 mm napočítáme 2780 přerušení optické závory a tedy:

$$d = \frac{l}{n} = \frac{290mm}{2780} = 0{,}104mm$$

jeden krok je tedy 0,104 mm.

Při použití rozlišení 300 bodů na jednu otáčku získáme téměř třikrát lepší rozlišení než u konstrukce s potenciometrem.

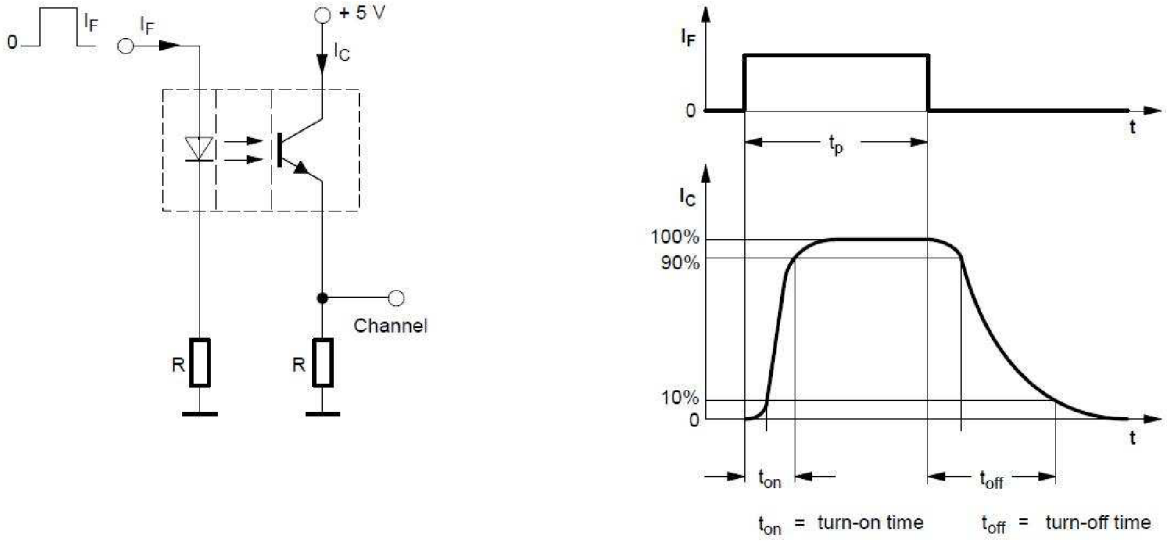
Jako optická závora je použit obvod TCST1103. Jde o zařízení, kde jsou zdroj a detektor signálu umístěny na stejné ose. Vlnová délka zdroje záření je 950nm, nachází se tedy mimo viditelné spektrum, díky čemuž není rušena okolním světelným pozadím.



Obr. 19: Vnitřní struktura a pouzdro TCST1103.

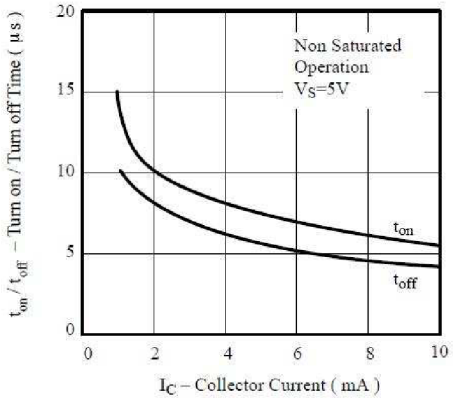
Na obrázku 19 je nalevo zobrazena vnitřní struktura TCST1103 a napravo pouzdro TCST1103.

Zdroj je třeba budit takovým proudem, aby neprosvítil stínící části kódového kolečka. Proto byl zdroji vřazen do série odpor 10KΩ, tím je proud zdrojem omezen na 5mA. Pro rychlost reakce detektoru je rozhodující proud jim protékající.



Obr. 20: Zpoždění TCST1103.

Na obrázku 20 je vidět zpoždění t_{on} a t_{off}. Jeho velikost lze ovlivnit proudem I_c.



Obr. 21: Závislost Zpoždění na kolektorovém proudu.

Proud *Ic* je určen podle grafu na obrázku 21. Proud Ic = 6mA. Odpor v sérii s detektorem je tedy:

$$R = 33\Omega$$

Odpor 33Ω není v sérii a tak byl vybrán nejbližší dostupný. Tedy 39Ω.

Mezi optickou závoru a procesor byl zařazen RC článek a Schmittův klopný obvod, a to z důvodu úpravy signálu a ochrany proti zákmitům.

Jeden naměřený puls představuje vzdálenost:

$$d = \frac{l}{n} = \frac{290mm}{2780} = 0{,}104mm$$

kde *n* je počet čar a *l* délka posuvu.

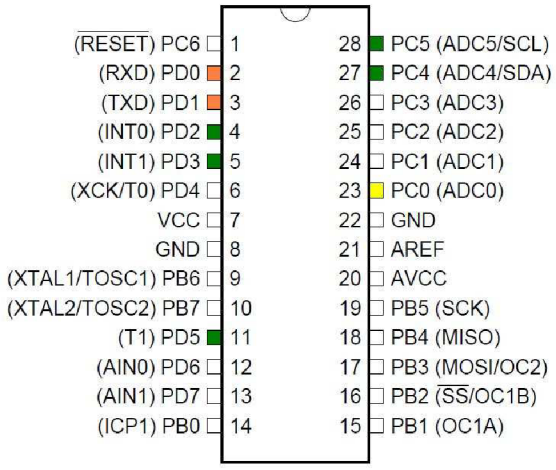
Nezávisle na pohybu vozíčku představuje každý puls 0,104mm. Signál z čidla, které snímá intenzitu laserového záření, je možné vzorkovat s tímto nebo menším rozlišením. Konkrétní přesnost je záležitostí programu.

4.4 Řídící jednotka

V závěru předchozí kapitoly byl zmíněn procesor, což je nejdůležitější součást zařízení (bude řídit motor a zpracovávat informace z detektoru).

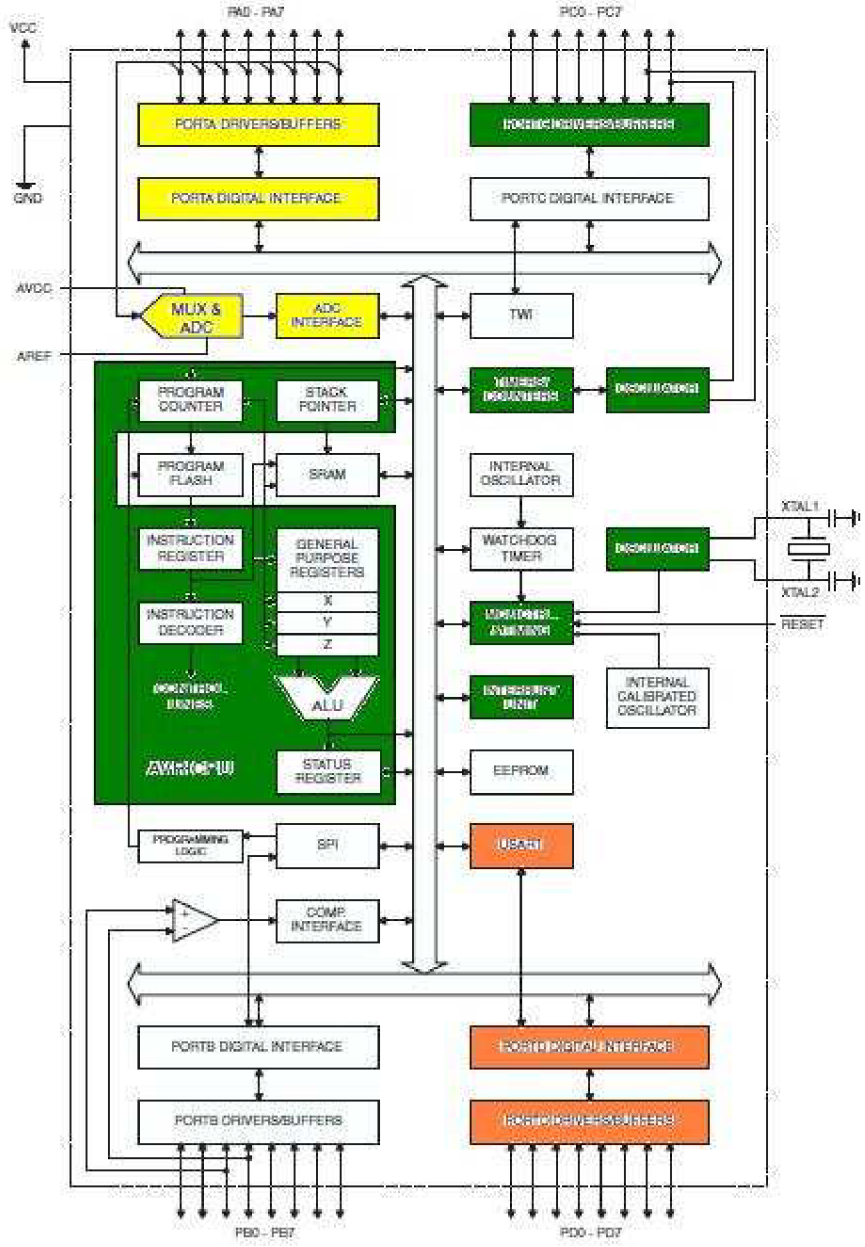
Ze zkušenosti byl zvolen procesor od výrobce ATMEL z řady MEGA, konkrétně ATmega8. Následující seznam popisuje vlastnosti a schopnosti tohoto procesoru, tučně je pak pro snadnější orientaci zvýrazněno to, co je od něj pro danou práci vyžadováno.

ATmega8 je nízkonapěťový 8 bitový Mikroprocesor, který využívá RISC architekturu.
- 131 Instrukcí
- **32 8-mi bitových pracovních Registrů**
- **16K Bytů programové Flash paměti**
- 512 Bytů EEPROM
- 1K Byte Interní SRAM
- Programovatelný zámek pro zabezpečení programu mikrokontroleru
- Programování Flash, EEPROM, zámků a řídících registrů přes JTAG rozhraní
- **Dva 8-bitové časovače / čítače** s oddělenými děličkami a porovnávacími režimy
- **Jeden 16-bit čítač / časovač s oddělenou předděličkou**, porovnávací mód
- Čítač reálného času se samostatným oscilátorem
- Čtyři **PWM** kanály
- 8-kanálový**, 10-bitový AD převodník**
- 2 diferenční kanály s programovatelným zesílením na 1x, 10x nebo 200x
- Bytově - orientované Sériové rozhraní
- **Programovatelný USART**
- Master / Slave SPI Sériové rozhraní
- Interní kalibrovaný RC oscilátor
- **Vnější a vnitřní zdroje přerušení**
- Šest režimů spánku: Idle, ADC redukce šumu, Power-save, Power-down, Standby a Extended Standby
- Provozní napětí: - 2.7 - 5.5V pro Atmega8L
  - 4.5 - 5.5V pro ATMEGA8
- Rychlostní módy: - +0 - 8 MHz pro Atmega8L
  - 0 - 16 MHz pro ATMEGA8



Obr. 22: Rozložení pinu na ATmega8

Dále je v blokovém schématu naznačeno, které periferie budou využity. V obrázku 22 jsou zvýrazněny barevně.



Obr. 23: Blokové schéma ATmega8.

Barevně je odlišeno, které periferie budou pro jednotlivé úkoly programu nezbytné.

- Žlutá barva vyznačuje, jaké části mikrokontroleru budou potřeba pro vzorkování signálu, tedy vzorkování intenzity laserového svazku.
- Zeleně je vyznačeno, co bude potřeba pro řízení motoru.
- Červeně je vyznačena část, která nastavuje komunikaci celého zařízení s PC.

4.4.1 Čítač kroků (Timer/Counter1)

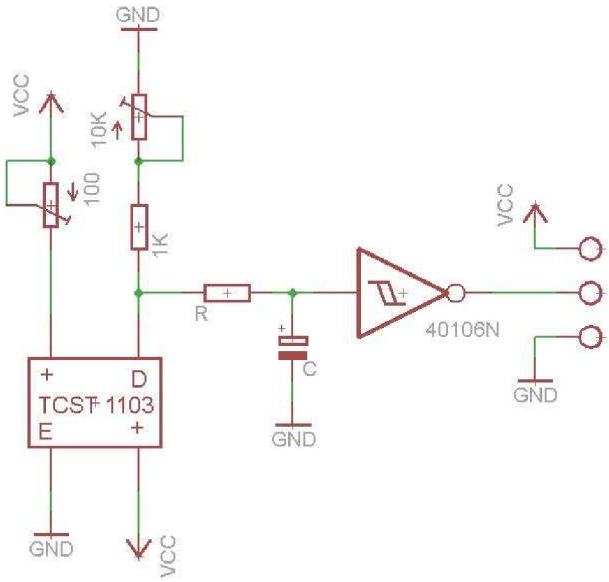
Pro čítání kroků na celé šířce měření by osmi bitový čítač nestačil, proto je potřeba použít šestnácti bitový čítač.

V tomto případě je pro čítání užit šestnácti bitový Timer/Counter1. Pro nastavení tohoto čítače stačí jeden registr TCCR1B, v něm musí být nastaveno na hodnotu 1 pouze 2 bity: CS12, CS11. Tímto způsobem byl nastaven zdroj hodinového signálu na externí, pin T1(PD5). Čítač bude reagovat na sestupnou hranu.

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

Obr. 24: Registr TCCR1B [4].

Vstupem čítače je optická závora.



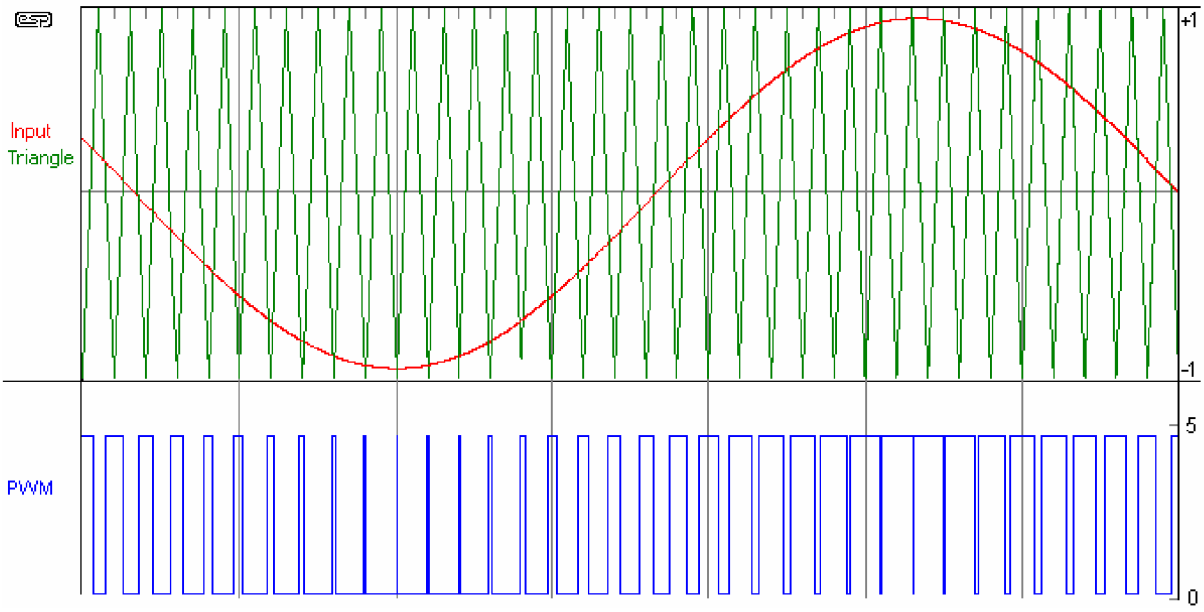
Obr. 25.: Schéma obvodu pro čítání.

4.4.2 PWM pro řízení motoru

PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu, informace je přenášena pomocí střídy. Díky této vlastnosti je pulsně šířková modulace často využívána ve výkonové elektronice pro řízení velikosti napětí nebo proudu do zátěže. Právě této vlastnosti využívají PWM regulátory.

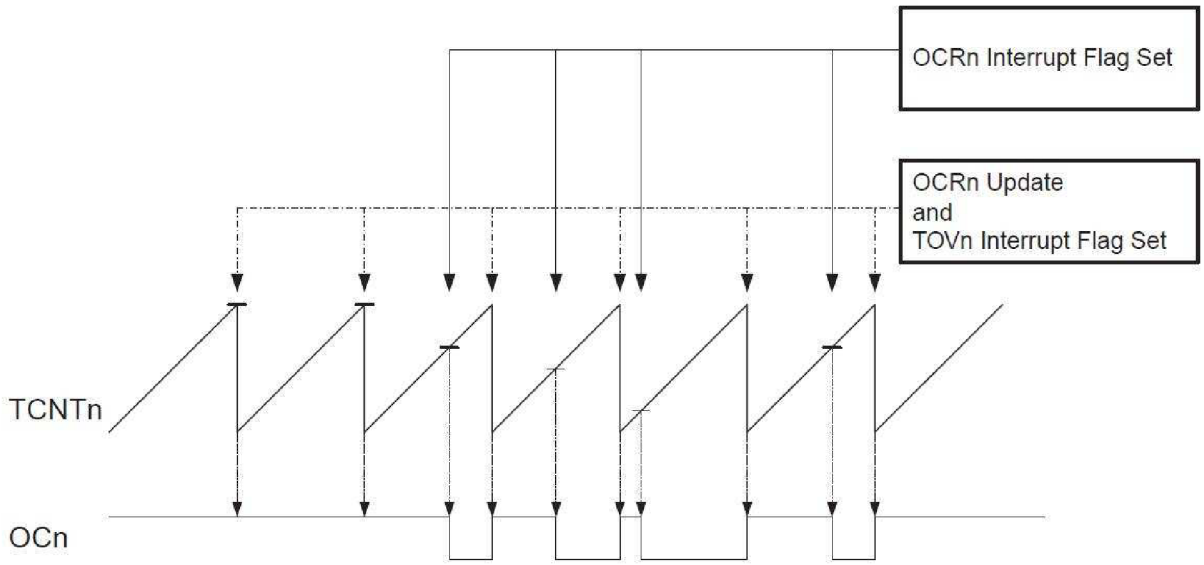
PWM regulátory se používají pro regulaci otáček stejnosměrných motorů. Jedná se o regulaci využívající změny šířky proudového impulzu do motoru, čímž se liší od spojité regulace proudu, kde nedochází jen ke snižování proudu ale i napětí. Při pulzní regulaci zůstává proud i napětí stejné, ale mění se aktivní doba, kdy proud prochází motorem. Motor, jenž je takto regulovaný má větší sílu i při nižších otáčkách, jeho nejdůležitější vlastností ale je, že tento druh regulace je prakticky bezztrátový (spínací prvek přechází do plně otevřeného stavu velice rychle). Když je spínací prvek otevřen, má velmi malý odpor a ztráta je tak

zanedbatelná, pokud je uzavřen, neteče jím proud, a tak nemají ztráty kde vznikat. Výraznější ztráty vznikají jen při spínání a jsou úměrné frekvenci, na které PWM pracuje.



Obr. 26: Modulace PWM.

Na obrázku 26 lze pozorovat zjednodušený princip modulace PWM. Hradlo se překlopí pokaždé, když modulační signál protíná trojúhelníkový modulovaný signál. Frekvence modulovaného signálu určuje frekvenci PWM a tím i velikost ztrát.



Obr. 27: Generování "Fast PWM" v mikroprocesoru [4].

Na obrázku 27 je vidět, že v mikrokontroleru dochází ke generaci PWM trochu jiným způsobem – využívá k ní periferii Timer/Counter. Čítač čítá od nuly do maxima, v okamžiku

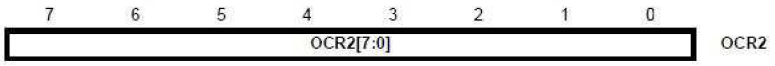
kdy je hodnota v čítači rovna přednastavené hodnotě, dojde k překlopení výstupního hradla. V okamžiku kdy čítač dosáhne maxima, se výstupní hradlo překlopí zpět.

V tomto případě je pro generování PWM použit Timer/Counter2. Pro nastavení tohoto čítače na PWM stačí 2 resp. 3 registry.

V registru TCCR2 je třeba nastavit na hodnotu 1 pouze 2 bity: WGM21 a WGM20. Tím bude zapnut Fast PWM Mode.

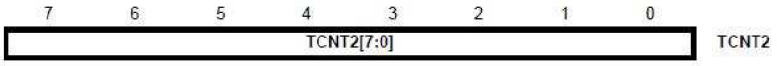
Obr. 28: Registr TCCR2 [4].

Registr TCCR2 obsahuje zadanou hodnotu, kterou stále porovnává s registrem TCNT2 a v případě shody dojde k překlopení výstupního hradla (v okamžiku kdy TCNT2 dosáhne maxima se výstupní hradlo překlopí zpět). Hodnotou v registru OCR2 je regulována střída. Jedná se o osmi bitový čítač, a proto hodnota 0 znamená střídu 0% a hodnota 255 střídu 100%.



Obr. 29: Registr OCR2 [4].

Registr TCNT2 obsahuje aktuální hodnotu čítače. Při přetečení se vynuluje.



Obr. 30: Registr TCNT2 [4].

4.4.3 AD převodník (Analog to Digital Converter)

Procesor ATmega8 je vybaven deseti bitovým AD převodníkem, který disponuje osmi kanálovým multipexerem. Pro napěťovou referenci je na výběr buď externí referenční obvod, vnitřní reference na úrovni napájecího napětí nebo vnitřní reference na úrovni 2,56V. I když číslo 2,56 působí nezvykle, je dělitelné 256. Tato volba je vhodná při využití jen osm bitů AD převodníku.

V tomto jsou pro nastavení AD převodníku použity 2 resp. 3 registry.

V registru ADMUX musí být nastaveny dva bity: REFS1 na 0 a REFS0 na 1. Tím je nastavena vnitřní napěťová reference na 5V napájecího napětí. Hodnota MUX3..0 je nulová, protože je použit jen jeden kanál: ADC0.

Obr. 31: Registr ADMUX [4].

V registru ADCSRA je třeba nastavit dva bity: ADEN na 1, tím je zapnut AD převodník, a ADSC na 1, čímž bude spuštěn každý jednotlivý AD převod v Single Conversion módu.

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA

Obr. 32: Registr ADCSRA [4].

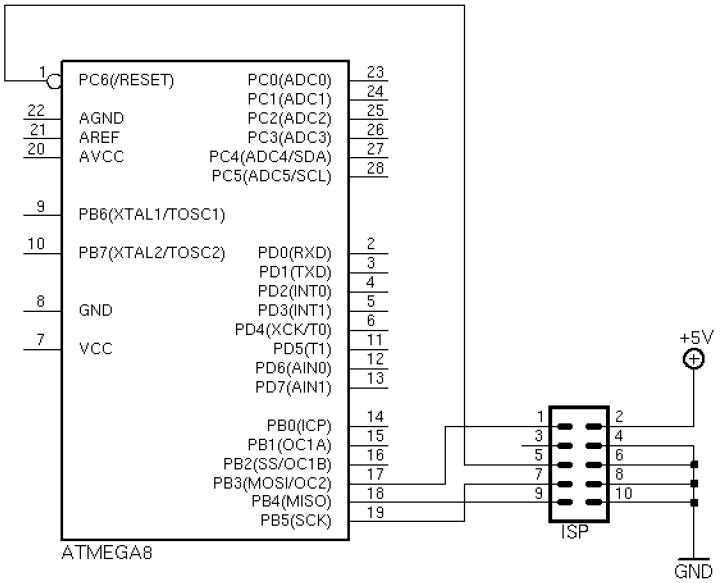
Registr ADC obsahuje výsledek AD převodu. Pokud je v registru ADMUX nastaven bit ADLAR na hodnotu 0, je výsledek zarovnán doprava.

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

Obr. 33: Registr ADC [4].

4.4.4 ISP rozhraní

ISP je zkratka anglického termínu „In-System Programming“, což znamená, že FLASH (a EEPROM) paměť lze programovat bez odpájení nebo vyjmutí procesoru z patice. Při návrhu desky plošných spojů je tedy třeba pamatovat na ISP a přidat deseti pinový konektor MIL10. Pokud není procesor nastaven pro použití s vnitřním RC oscilátorem, je navíc třeba připojit externí zdroj hodinového signálu, např. krystal. Zdroj hodinového signálu je možné konfigurovat pomocí bitů, které jsou nastavovány při programování procesoru. Další možností je použití Boot-Loaderu, což je část programu která umožní programovat procesor přímo. Např.: přímo přes USART nebo I2C sběrnici.



Obr. 34: Rozhraní ISP na konektoru MIL10.

4.4.5 ISR přerušení

ISR přerušení nelze brát jako samostatnou periferii, jde o sadu registrů, kterými je nastavována resp. povolována obsluha přerušení při výskytu specifické události v ostatních periferiích. Tyto události nazýváme vektory přerušení.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
12	0x00B	USART, RXC	USART, Rx Complete

Obr. 35: Tabulka použitých přerušení.

Na obrázku 35 jsou uvedena přerušení používaná programem procesoru. Externí přerušení INT0 a INT1 jsou použita pro zastavení motoru při sepnutí koncových spínačů, Timer1 je použit k řízení motoru, přerušení RXC pro příjem instrukcí.

V programu dochází k obsluze přerušení odskočením z běžícího programu do vektoru přerušení.

```
//V preruseni se prijaty Byte prepise do promene "reciv..."
ISR(USART_RXC_vect)
 {
    recieved_instruction[s] = USART_Receive();
    s++;
    if (s == 2)
        {
         s = 0;
         RX_complete = 1;
         scale = recieved_instruction[1];
         scale = scale - 48;
         USART_Transmit_Text(recieved_instruction);
        }

 }
```

Obr. 36: Obsluha přerušení pro příjem vícebitové instrukce.

4.4.6 Komunikace a nastavení USART

Jednu z úloh, které bude procesor zastávat je komunikace s PC. Pro komunikaci je využita jednotka USART.

Základní dovednosti a vlastnosti této komunikační jednotky jsou plně duplexní provoz, tzn. nezávislé registry pro příjem a odesílání sériových dat, asynchronní nebo synchronní provoz a volitelná přenosová rychlost. USART podporuje sériovou komunikaci v rámcích s 5, 6, 7, 8 nebo 9 bity dat, 1 nebo 2 Stop bity, generování liché a sudé parity. Dále

detekuje přetečení a rámcové chyby. Odfiltrování šumu zabezpečuje detekci falešných start bitů a digitální nízko-pásmový filtr. USART ovládá tři přerušení: vysílání dokončeno, vysílací registr prázdný a příjem dokončen a dvojnásobnou rychlost komunikace, ale pouze pro asynchronní mód.

Z toho všeho je pro práci zajímavá pouze Asynchronní sériová komunikace, přesněji standart pro linku RS232.

Pro správnou funkci jednotky USART je nutné nastavit řídící registry. Tyto jsou řídící a stavový registr a registr symbolové rychlosti. Dále je v jednotce USART obsažen I/O registr UDR, což je datový registr pro příjmané/vysílané slovo.

Řídící a stavové registry UCSRA, UCSRB, UCSRC, UBRRH a UBRRL jsou popsány jen schematicky, v závorkách jsou uvedeny hodnoty, které jsou nastavovány přímo pro tuto aplikaci.

7	6	5	4	3	2	1	0	
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	**UCSRA**

Obr. 37: Rozložení řídících a stavových bitů v registru UCSRA [4].

RXC: USART Receiver Complete
TXC: USART Transmit Complete
UDRE: USART Data Register Empty
FE: Frame Error
DOR: Data OverRun
PE: Parity Error
U2X: Double USART Transmission Speed - dvojnásobná přenosová rychlost (jen u asynchronní)
MPCM: Multi-processor Communication Mode - všechny příchozí rámce, které neobsahují adresu, budou ignorovány (data/adresa v rámci je signalizována 1. stop bitem, nebo 9. datovým bitem 0)

7	6	5	4	3	2	1	0	
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	**UCSRB**

Obr. 38: Rozložení řídících a stavových bitů v registru UCSRB [4].

RXCIE: RX Complete Interrupt Enable - povoluje přerušení při ukončení příjmu
TXCIE: TX Complete Interrupt Enable - povoluje přerušení při ukončení vysílání
UDRIE: USART Data Register Empty Interrupt Enable – povoluje přerušení, pokud je datový registr prázdný
RXEN: Receiver Enable - povolení používání USART přijímače (pin RxD (PD0) na pouzdru)
TXEN: Transmitter Enable - povolení používání USART vysílače (pin TxD (PD1) na pouzdru)
UCSZ2: Charakter Size - v kombinaci s UCSZ1:0 (viz dále) nastavuje počet datových bitů v přenášeném rámci
RXB8: Receive Data Bit 8 - devátý bit při devítibitovém přijatém datovém slově (musí být čteno dříve než nižších 8 bitů z UDR)
TXB8: Transmit Data Bit 8 - devátý bit při devítibitovém vysílání

7	6	5	4	3	2	1	0	
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC

Obr. 39: Rozložení řídících a stavových bitů v registru UCSRC [4].

URSEL: Register Select - výběr přístupu k registru UCSRC nebo UBRRH (sdílí stejnou paměť)
UMSEL: USART Mode Select - 0/1: asynchronní/synchronní komunikace
UPM1:0: Parity Mode
USBS: Stop Bit Select - výběr počtu stop bitů pro vysílač; 0: 1 stop bit, 1: dva stop bity
UCSZ1:0: Character Size - společně s UCSZ2 (UCSRB registr) definuje délku datového slova
UCPOL: Clock Polarity - pouze pro synchronní komunikaci

UPM1:0	*Parita*
00	Bez parity
01	Rezervováno
10	Sudá parita
11	Lichá parita

UCSZ2:0	*Délka slova*
000	5 bitů
001	6 bitů
010	7 bitů
011	8 bitů
111	9 bitů

Obr. 40: Hodnoty bitů UPM a UCSZ [4].

URSEL	-	-	-	UBRR[11:8]	UBRRH
UBRR[7:0]					UBRRL

Obr. 41: Řídících a stavových bitů v registru UBRRH a UBRRL [4].

UBRR: USART Baud Rate Register Clock Polarity - dvanácti bitová hodnota určující symbolovou rychlost:

$$UBRR = \frac{\text{frekvence mikrokontroléru}}{(16 \cdot \text{symbolová rychlost})} - 1$$

Jak vypadá nastavení těchto registrů v praxi, je možné ukázat na části zdrojového kódu. Součástí inicializace jednotky USART je i nastavení baudové rychlosti (je parametrem funkce). Je nutné nastavit povoleni přijmu, vysílaní dat a přerušeni při dokončeni přijmu (toho bude využito při zahájení posuvu). A samozřejmě musíme nastavit datový rámec.

```
void USART_Init( unsigned int baudvalue )
 {
    /* Set baud rate */
    int baud;
    baud=((F_CPU/8)/(2*baudvalue))-1;
    UBRRH = (unsigned char)(baud>>8);
    UBRRL = (unsigned char)baud;
    /* Enable Receiver and Transmitter */
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
    /* Set frame format: 1stop bit, 8data, bez parity */
    UCSRC = (1<<URSEL)|(0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0)
 }
```

Obr. 42: Funkce USART_Init.

Nyní je jednotka USART inicializována a může tedy začít komunikovat. Aby byl příjem a odesílání dat zjednodušen, je užitečné navrhnout funkce. Zdrojem informací pro tyto funkce byl [4].

```
unsigned char USART_Receive( void )
    {
    /* Wait for data to be received */
    while ( !(UCSRA & (1<<RXC)) );
    /* Get and return received data from buffer */
    return UDR;
    }
```

Obr. 43: Funkce USART_Recieve.

```
void USART_Transmit( char data )
    {
    /* Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) );
    /* Put data into buffer, sends the data */
    UDR = data;
    }
```

Obr. 44: Funkce USART_Transmit.

Pro účely práce se hodí si funkce trochu upravit a doladit, pro případ, že budeme posílat text nebo číslo. Je vidět, že funkce na sebe postupně navazují.

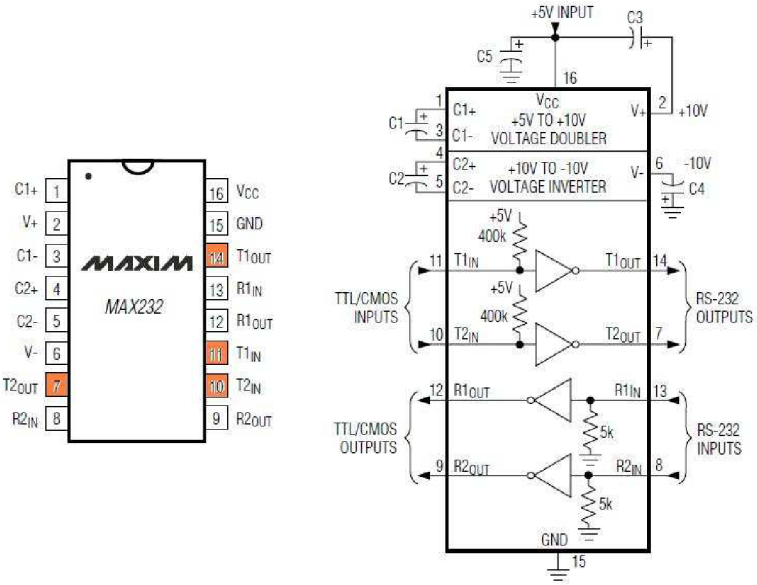
```
void USART_Transmit( char data )
    {
    /* Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) );
    /* Put data into buffer, sends the data */
    UDR = data;
    }

void USART_Transmit_Text( char textdata[30] )
    {
    int i = 0;
    /* Wait for empty transmit buffer */
    while ( !( UCSRA & (1<<UDRE)) );
    /* Put data into buffer, sends the data */
    while(textdata[i] != '\0')
        {
        USART_Transmit(textdata[i]);
        i++;
        }
    }

void USART_Transmit_Integer( int i )
    {
    char intdata[32];
    /* Conver Interg TO Ascii */
    itoa(i, intdata, 10);
    /* Sends the data */
    USART_Transmit_Text(intdata);
    }
```

Obr. 45: Funkce pro posílání textu a čísel.

V tuto chvíli už umí mikrokontroler komunikovat, pro testování na straně PC prozatím bude stačit např. program Terminal. Pro finální verzi je přichystán jednoduchý program s grafickým rozhraním, připraveným přímo pro tuto aplikaci.

Standart RS232 pracuje na napětích +15 a -15V, a proto je nutné zařadit obvod MAX232, aby převedl signál na úrovně TTL.

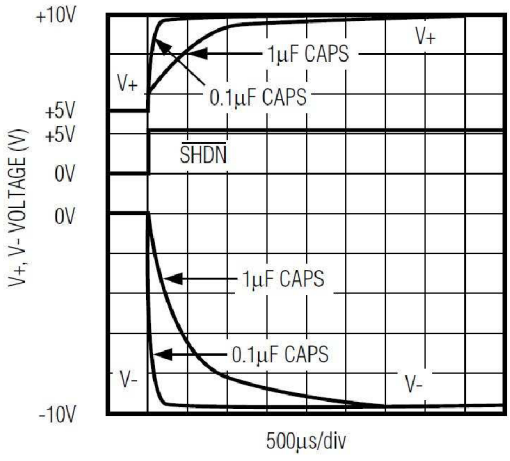
Obvod MAX232 obsahuje dvě vnitřní nábojová čerpadla, které přeměňují +5 V na ± 10V (bez zatížení komunikačními obvody) pro RS-232 standart. První část obvodu (násobič) používá kondenzátor C1 a C3 pro zvýšení napětí na dvojnásobek (z +5V na +10V). Zjednodušeně lze říci, že kondenzátory jsou paralelně nabity a sériově vybity. Druhá část obvodu (invertor) používá kondenzátor C2 a C4 k inverzi na -10 V(z +10V na -10V).



Obr. 46: Bloková struktura MAX232 [5].

Na obrázku 46 je znázorněn obvod MAX232 a jeho vnitřní bloková struktura. Barevně označené piny jsou použity pro komunikaci.

Je-li použit obvod MAX232 od firmy MAXIM, jsou kondenzátory C1 až C5 pouze 0,1uF, Konkurenční výrobky potřebují pro správnou funkci kondenzátory s větší kapacitou.



Obr. 47: Napěťové úrovně MAX232 [5].

Standart RS232 pracuje s logickými úrovněmi +15V a -15V. Obvod MAX232, pokud je zatížen, je v ideálním případě schopen dosáhnout napěťových úrovní pouze +8,5V a -9,5V. I tyto hodnoty jsou dostačující pro bezproblémovou komunikaci.

5 Obvodová realizace

Na následujících schématech je vidět, jak lze navrhnout desky pro řídící obvody. Na těchto deskách se nachází napájecí obvody, komunikační obvody (obvod MAX232) snímací a vzorkovací obvody a samotný procesor. Snímač se zesilovačem se nachází na samostatné desce plošného spoje, aby nedocházelo k rušení.

Zapojení samotného procesoru vyplívá z potřeb projektu. Procesor musí být připojen dvěma piny (2 PD0(RXD) a 3 PD1(TXD)) k obvodu MAX232 kvůli komunikaci. Další dva piny jsou použity pro připojení koncových spínačů (4 PD2(INT0) a 5 PD3(INT1) resp. 6 PD4(T0)). AD převodník je připojen na pin 23 PC0(ADC0). Pro řízení motoru jsou použity piny 28 PC5(ADC5/SCL), 27 PC4(ADC4/SDA) a 17 PB3(MOSI/OC2). Pro sledování stavu posuvu 11 PD5(T1).

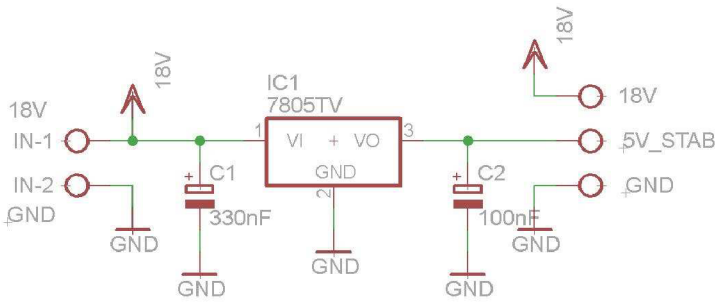
Deska je doplněna konektorem pro programování pomocí SPI, což souvisí s vývojem desky.

5.1 Napájení

Napájení bylo řešeno původním zdrojem tiskárny a napěťovou stabilizací na požadovaná napětí. Původní zdroj však nebyl vhodně zkonstruován, a jeho vlivem došlo ke zničení celé práce. Při následné opravě byl zdroj nahrazen spínaným síťovým adaptérem s pevným napětím 24V a proudovou limitací do 1A.

Obr. 48: Zdroj 24V/1000mA (ilustrační foto).

Na obrázku 48 je vidět nový 24V zdroj. Nezatížený dává 24V stejnosměrného napětí.



Obr. 49: Schéma úpravy napětí.

Na obrázku 49 je znázorněno zapojení běžného napěťového stabilizátoru LM7805. V konečné fázi však nebudou napájecí obvody na samostatné desce plošného spoje.

Obr. 50: Návrh DPS a rozložení součástek na samostatné napájení.

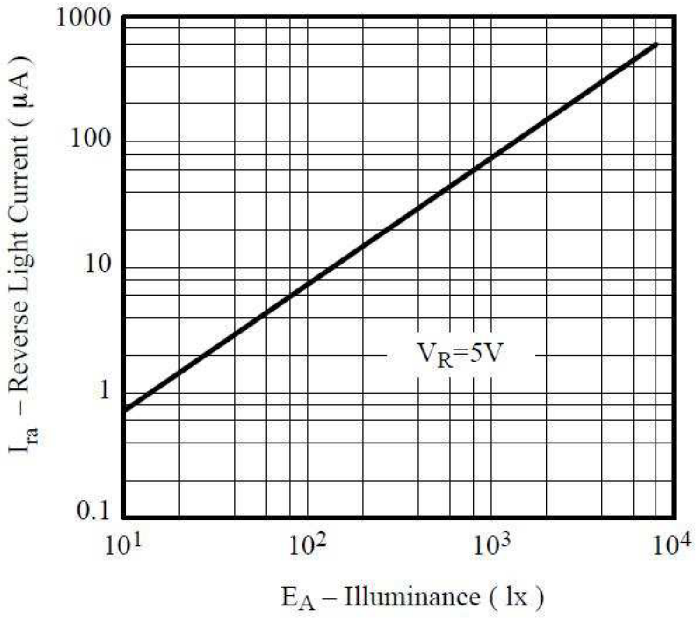
Na obrázku 50 je vidět návrh DPS a zapojení běžného napěťového stabilizátoru LM7805, pokud by bylo potřeba umístit napájecí obvody na samostatnou desku plošného spoje.

5.2 Snímač intenzity světla.

Pro snímání intenzity laserového paprsku byla použita fotodioda v režimu proudového zdroje. Z katalogu byla vybrána BPW34.

BPW34 je rychlá a vysoce citlivá PIN fotodioda v miniaturním plochém plastovém pouzdře, vzhledem ke své čiré epoxidové konstrukci je citlivá na viditelné a infračervené světlo (záření). Důsledkem velké aktivní oblasti je dioda velmi citlivá a má široký přijímací úhel.

Fotodioda pracuje v proudovém režimu, a to nakrátko (U = 0V), kdy je závislost fotoproudu na intenzitě dopadajícího záření prakticky lineární.



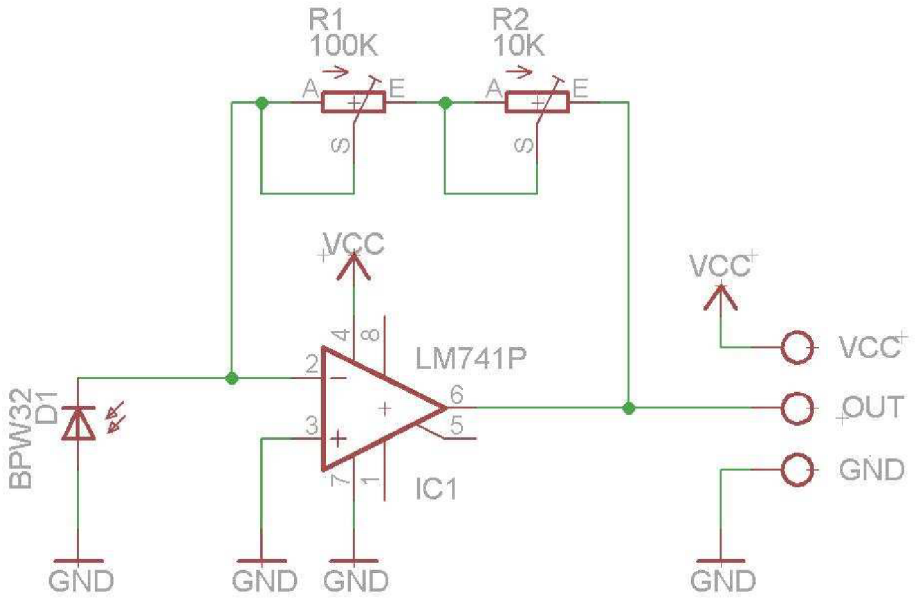
Obr. 51: Závislost závěrného proudu na osvitu diody[6].

Fotodioda je zkratována na zem invertujícím vstupem operačního zesilovače. Operační zesilovač nastavuje přes zápornou zpětnou vazbu mezi vstupy nulové rozdílové napětí. V tomto případě je dosažena tzv. virtuální nula na invertujícím vstupu. Veškerý fotoproud I_R protéká zpětnovazebním rezistorem, protože vstupní odpor operačního zesilovače je velký

(1012 Ω). Aby byla na invertujícím vstupu virtuální nula, nastaví operační zesilovač na svém výstupu napětí:

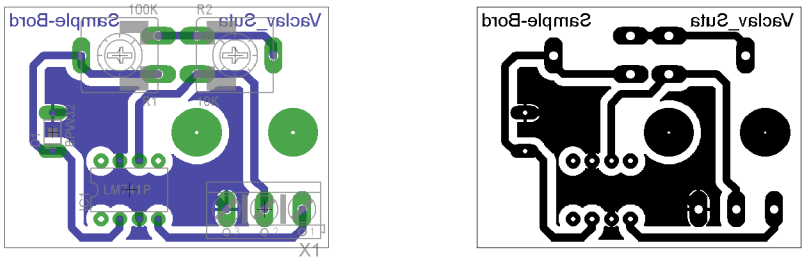
$$U_{výst} = R_F \cdot I_F$$

Výstupní napětí operačního zesilovače je proto přímo úměrné fotoproudu I_R. Konstantou úměrnosti je hodnota odporu zpětnovazebního rezistoru R_F. Fotoproud vždy vytéká z anody ven, proto je výstupní napětí v zapojení kladné.



Obr. 52:Schéma zapojení snímače intenzity laseru.

Na obrázku 52 je vidět schéma zapojení snímače intenzity profilu laserového svazku. Původní odpor 100KΩ je nahrazen dvěma trimry 100KΩ pro hrubé doladění a 10KΩ pro jemné doladění zesílení operačního zesilovače.



Obr. 53: Návrh DPS a rozložení součástek na snímači intenzity.

Tato konstrukce snímače ale nerespektuje přirozené světelné pozadí. Aby bylo možné tuto nepřesnost odstranit, je potřeba zkonstruovat dva takové snímače. Jeden použít jako detektor a druhý jako snímač přirozeného světelného pozadí. Jejich výstupní signály je třeba od sebe odečítat pomocí dalšího operačního zesilovače a teprve poté vzorkovat AD-převodníkem. Vyhnout se třetímu operačnímu zesilovači je možné vzorkováním obou signálů a následným matematickým odečtem až v procesoru.

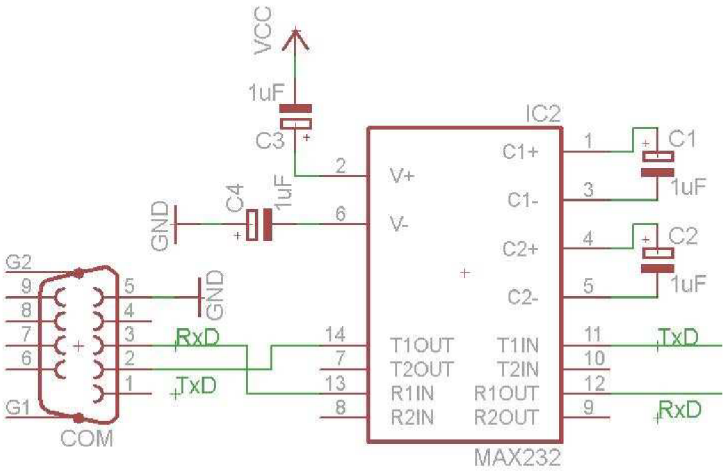
5.3 Komunikace po RS232

Komunikace probíhá ve standartu RS232, pomocí obvodu MAX232 a periferie USART (obvod ATmega8). Kompletní nastavení probíhá pomocí funkce USART_Init().

```
void USART_Init( unsigned int baudvalue )
 {
    /* Set baud rate */
    int baud;
    baud=((F_CPU/8)/(2*baudvalue))-1;
    UBRRH = (unsigned char)(baud>>8);
    UBRRL = (unsigned char)baud;
    /* Enable Receiver and Transmitter */
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
    /* Set frame format: 1stop bit, 8data, bez parity */
    UCSRC = (1<<URSEL)|(0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0)
 }
```

Obr. 54: Ukázka inicializace jednotky USART [4].

Baudovou rychlost je nastavena na 19200Bd. Bity RXEN a TXEN rovny 1 povolují příjem a vysílání, tzn. aktivují periferii. Pomocí přerušení při dokončení příjmu probíhá obsluha přijatých bytů, toto přerušení povoluje bit RXCIE. Je nutné nastavit také formát datového rámce: bit USBS roven 0 nastavuje jeden stopbit. Bity UCSZ2..0 nastavují kolik datových bytů bude přenášeno. Je nastaveno 8 datových bitů.



Obr. 55: Zapojení obvodu MAX232.

5.4 Optický inkrementální snímač polohy

Optickým inkrementálním snímačem polohy je myšleno kódové kolečko a optická závora.

Přesností a rozlišením kódového kolečka je dána i přesnost a četnost možných měření. Na kolečku je 300 čar. Na 290 mm napočítáme 2780 přerušení optické závory a tedy:

$$d = \frac{l}{n} = \frac{290mm}{2780} = 0{,}104mm$$

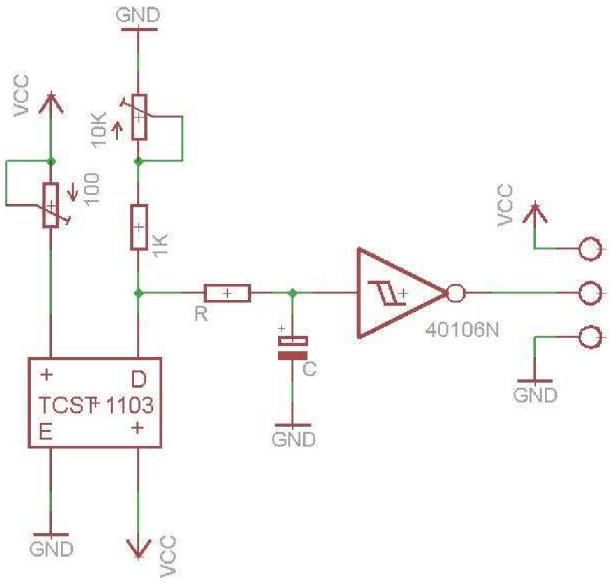
Tedy jeden krok je 0,104 mm.

Jako optická závora je použit obvod TCST1103. Vlnová délka zdroje záření je 950nm, nacház se tedy mimo viditelné spektrum a není rušena okolním světelným pozadím.

Zdroj je třeba budit takovým proudem, aby neprosvítil stínící části kódového kolečka. Oproti teorii však zdroji nevřadíme do série odpor 10KΩ, ale odpor 1KΩ a 10KΩ trimr. Tím je proud zdrojem omezen na 4,5mA a je zde možnost regulace pokud bude změněno kódové kolečko. Pro rychlost reakce detektoru je rozhodující proud jim protékající, jeho velikost lze ovlivnit proudem I_c. Proud I_c určíme pole katalogového listu. Proud I_c = 6mA. Odpor v sérii s detektorem je tedy:

$$R = 33\Omega$$

Odpor 33Ω není v sérii, proto byl zvolen nejbližší dostupný. Tedy 39Ω, popř. jej lze nahradit trimrem 100Ω.



Obr. 56: Schéma zapojení optické závory.

Mezi optickou závoru a procesor je zařazen RC článek a Schmittův klopný obvod pro úpravu signálu a ochranu proti zákmitům.

Časovou konstantu RC článku a jeho hodnoty lze určit podle vztahu:

$$\tau = \frac{1}{počočetPupsů} = \frac{1}{2780} = 359{,}7\,\mu s\,.$$

Hodnota kondenzátoru C je volena odhadem. Kondenzátor C je volen s hodnotou 100nF. Tedy:

$$\tau = R \cdot C$$

$$\frac{\tau}{C} = R$$

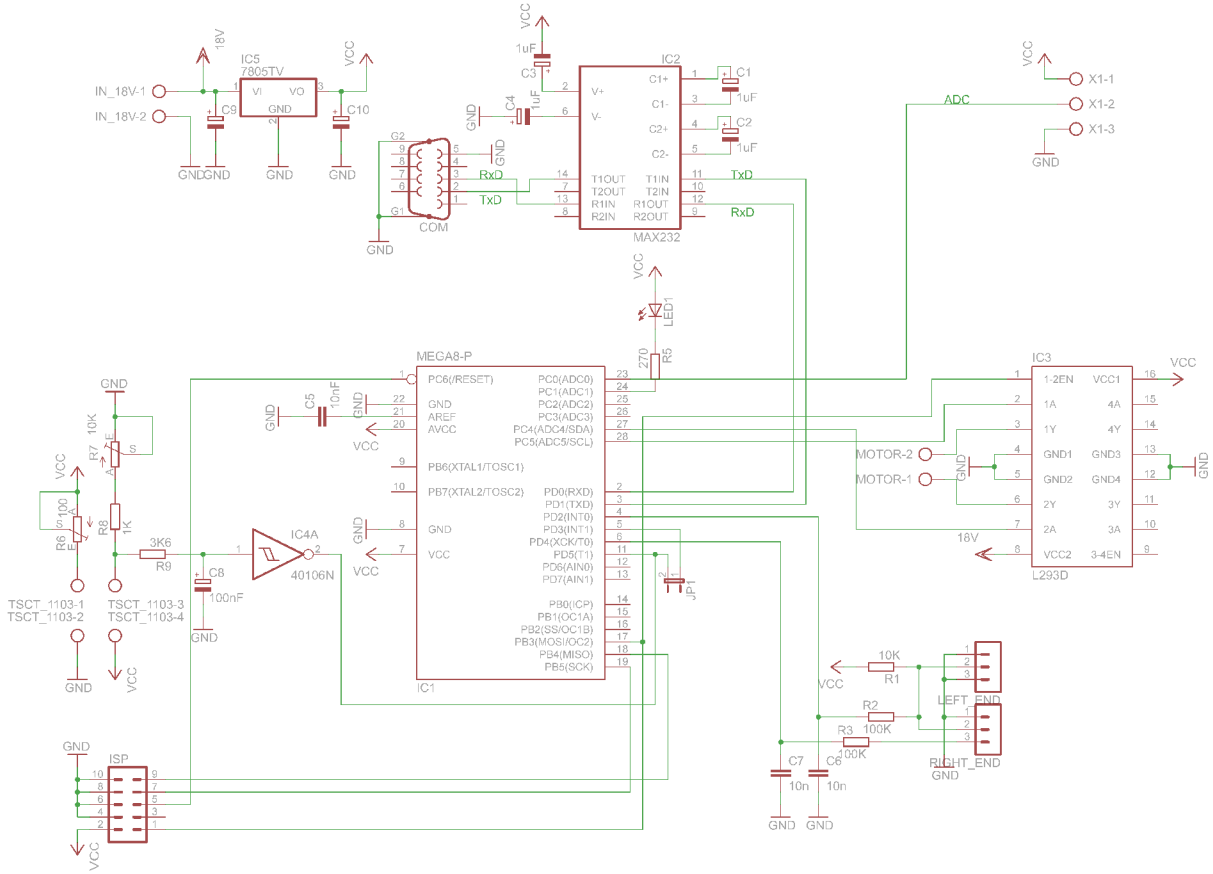
$$\frac{\tau}{C} = \frac{359{,}7\,\mu s}{100nF} = R = 3597\Omega\,.$$

Výsledná hodnota se v řadě nenachází, proto je třeba vybrat podobnou hodnotu např.: 3,3KΩ nebo 3,6KΩ.

Obr. 57: Návrh DPS a rozložení součástek na optické závoře.

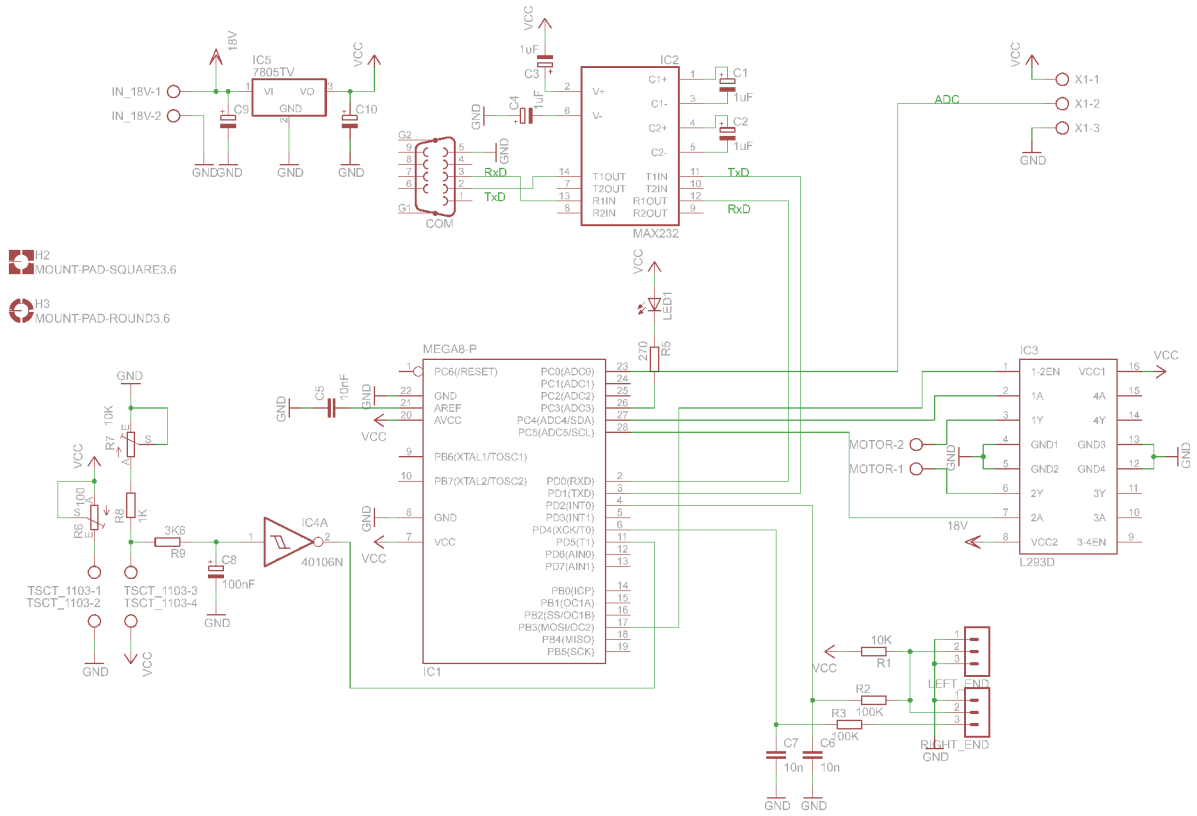
5.5 Řídící obvod

Řídícím obvodem je především myšlena deska plošného spoje osazena mikroprocesorem a podpůrnými obvody. Na následujícím schématu je vidět jak lze navrhnout řídící obvod. Na tomto schématu se nachází napájecí obvody, komunikační obvody (obvod MAX232) snímací a vzorkovací obvody a samotný procesor. Snímač se zesilovačem se nachází na samostatné desce plošného spoje, aby nedocházelo k rušení



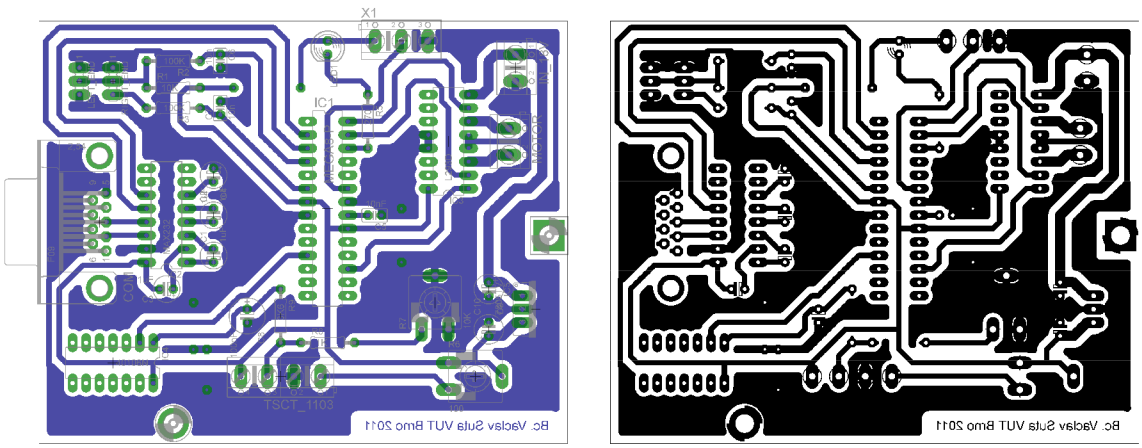
Obr. 58: Schéma řídící desky v1.0.

Pro testování bylo schéma lehce upraveno. Bylo odstraněno rozhraní ISP a jumper J1, dále byly zaměněny piny 26 a 27 na mikroprocesoru.



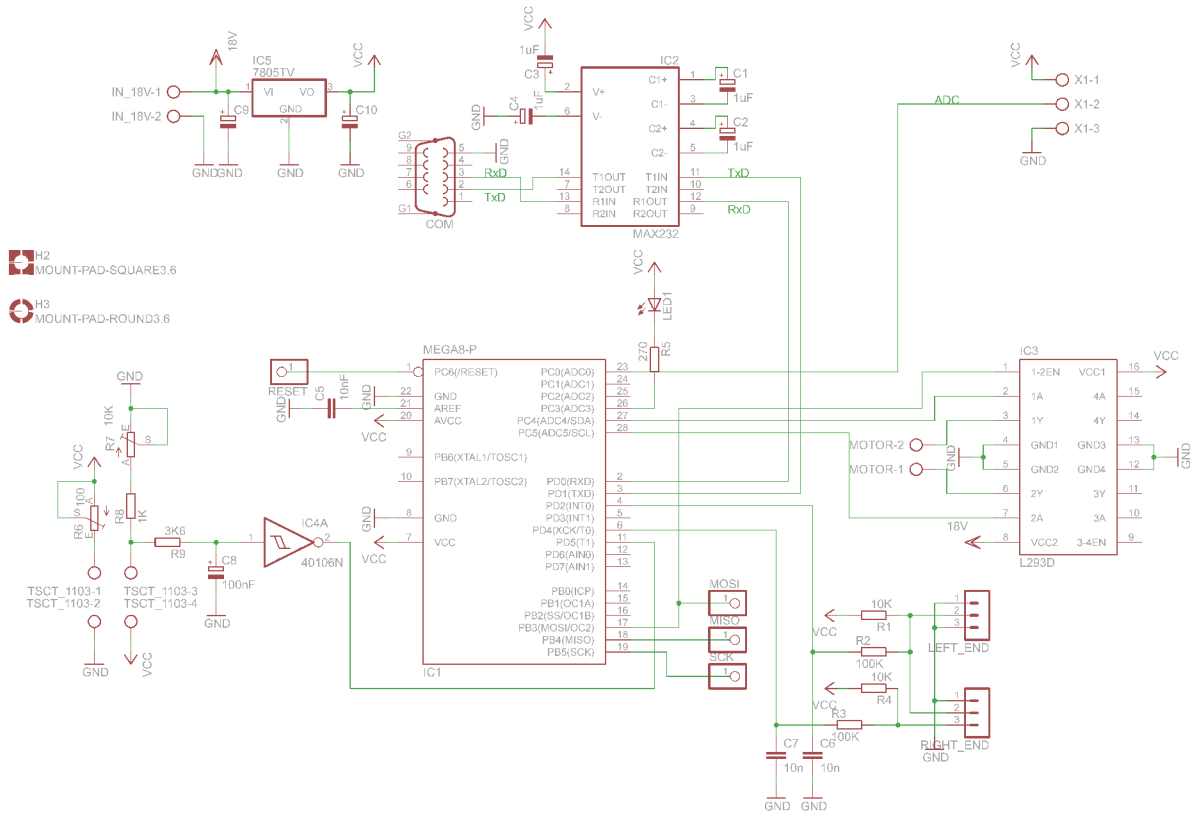
Obr. 59: Schéma řídící desky v1.2.

Z upraveného schématu je generována deska plošného spoje.



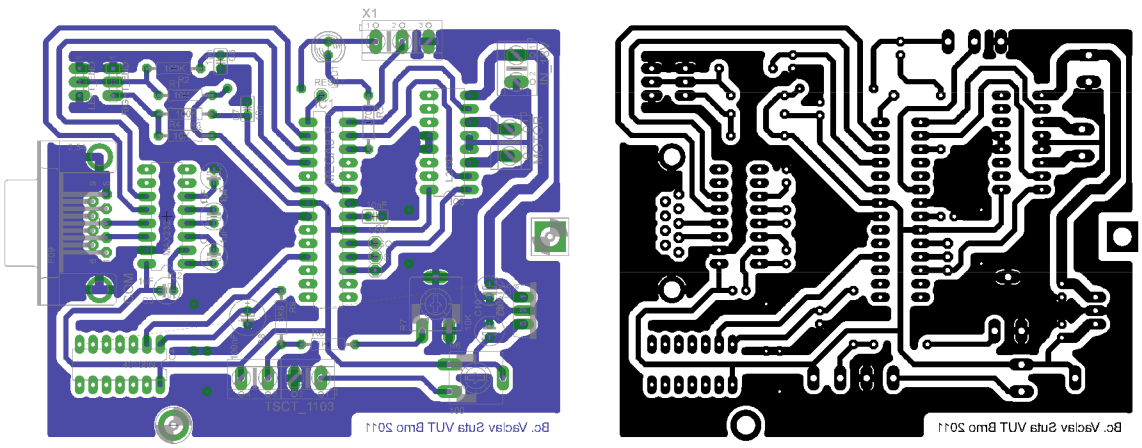
Obr. 60: Návrh DPS a rozložení součástek na řídící desce v1.2.

Závěrečné schéma je doplněno o piny pro ISP, zdvihový odpor R4 a je uzemněn konektor COM 9. Ve schématu není zakreslen odpor 10K mezi + 5V a pinem 1 procesoru ATmega8.



Obr. 61: Schéma řídící desky v1.3.

Ze schématu je generována finální deska plošného spoje.



Obr. 62: Návrh DPS a rozložení součástek na řídící desce v1.3.

5.6 Osazení a oživení řídící desky a detektoru

Při osazení a oživení řídící desky je vhodné postupovat následně.

1. Osadit DPS paticemi pro integrované obvody a zbylými součástkami a zapájet je. Připájet zem na pin 8 integrovaného obvodu 4. (chyba EAGLu)
2. Připojit a otestovat napájení. Změřit jestli je napětí za stabilizátorem 7805 opravdu +5V.
3. Pro tuto aplikaci je v procesoru ATmega8 používán vnitřní takt 8MHz. Je tedy nutné při programování procesoru nezapomenout nastavit bity (Fuses) v dolním Bytu (LowByte) na hodnotu „E4“.
4. Namontovat DPS na zařízení a připojit detektor.
5. Pro ověření komunikace osadit obvody MAX232 a ATmega8 (obvody osazujeme vždy s odpojeným napájením). Pro ověření lze použít:

```
ISR(USART_RXC_vect)
    {
    /* Recived character is send back to PC */
    recieved_instruction[s] = USART_Receive();
    USART_Transmit_Text(recieved_instruction);
    USART_Transmit('\n');
    }
```

Obr. 63: Přijatý znak je odeslán zpět do PC.

Kompletní zdrojový kód je uveden v příloze pod názvem: „Verification_of_Comunication.“
Při ověřování je potřeba zapojit veškerou kabeláž, např. napájení a komunikační kabely.

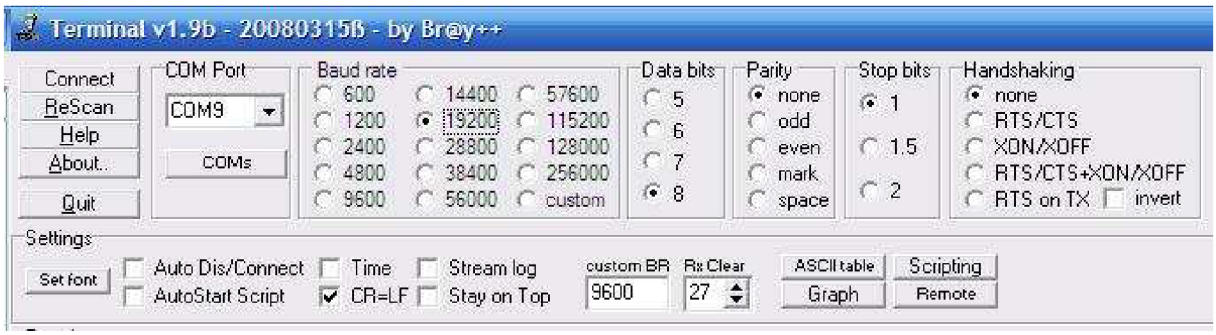
6. Pokud komunikace pracuje bez potíží (tzn. správně napájené patice i kondenzátory pro MAX232) a program se nezasekává (dioda stále bliká), lze nahrát celý zdrojový kód. Ten je umístěn v příloze pod názvem „Soft_v1_6“. Dále je potřeba osadit zbylé integrované obvody.
   Při použití programování ISP je potřeba zajistit společnou zem pro zařízení i programátor a odpojit motor (PWM pro řízení motoru používá stejný pin jako MOSI).
7. Je-li program v1.6 nahraný, napájení zapojené a dioda bliká, je zařízení připraveno pro posuv.
8. Aby bylo možné správně měřit, je potřeba naladit detektor na správné zesílení. To se provádí trimry R1 a R2 (250KΩ a 10KΩ) přímo na detektoru. Zesílení musí být nastaveno tak, aby se detektor nedostával do saturace.
9. V tuto chvíli je zařízení připraveno k měření.

5.7 Oživení ovládání

Pro ovládání bylo navrženo GUI. Bohužel zde stále dochází k chybám, a proto je ovládání celého zařízení prozatím prováděno pomocí programu Terminál.

Aby komunikace probíhala správně, je na rozdíl od GUI potřeba nastavit správný komunikační rámec a baudovou rychlost.

Jak už bylo napsáno výše, je pro přenos jednoho symbolu použito osmi bitů bez parity s jedním stop bitem. „*Handshaking* “ není použit. Mikroprocesor má v jednotce USART nastavenu baudovou rychlost na 19200, proto je důležité i tady nastavit tu samou hodnotu.



Obr. 64: Správné nastavení programu Terminál.

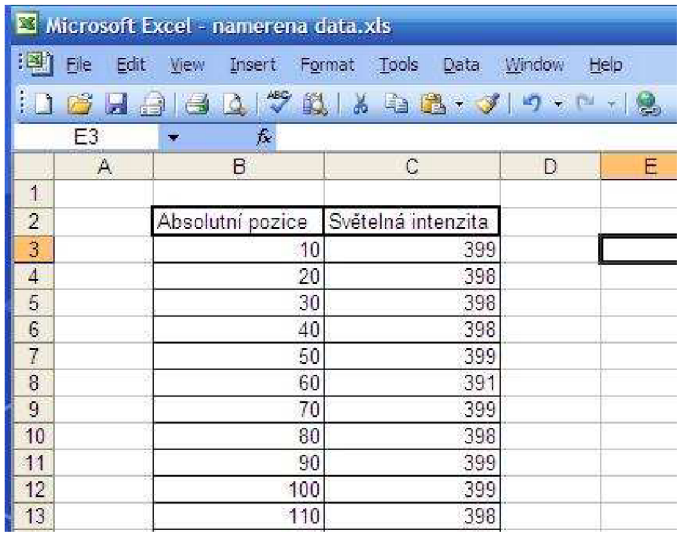
GUI používá pro komunikaci pěti-bytové instrukce, a proto je program Terminál musí používat také. První byte představuje jaký mód nebo funkce se bude provádět a následující čtyři byty představují měřítko. Tzn. v měřícím módu po kolika krocích má být odečten vzorek a při posuvném módu na jakou pozici má být detektor posunut.

Pro posuv na pozici je používána instrukce „**w**xxxx.“ Příkladem instrukce „w1234“ nastaví detektor na Absolutní pozici 1234.

Obr. 65: Příklad instrukce pro posuv.

Při zadávání Align to Zero „**a**yyyy“ dosadíme za hodnotu y cokoliv, procesor hodnotu těchto znaků ignoruje ale je třeba **vždy odesílat celou pěti-bytovou instrukci**, neboť musí být dodržen její rámec. Před každým měřením je vhodné kalibrovat přístroj a nastavit čítač kroků na nulu příkazem „a0000“.

Významnější je instrukce pro měření „**m**xxxx“. Za x se dosadí hodnotu po kolika krocích vzorkovat. Např. instrukce „m0010“ nastaví procesor do měřícího módu a začne měřit, tzn. najede na absolutní pozici 10 a odebere vzorek z detektoru a pošle jej do PC, následně najede na pozici 20 atd…

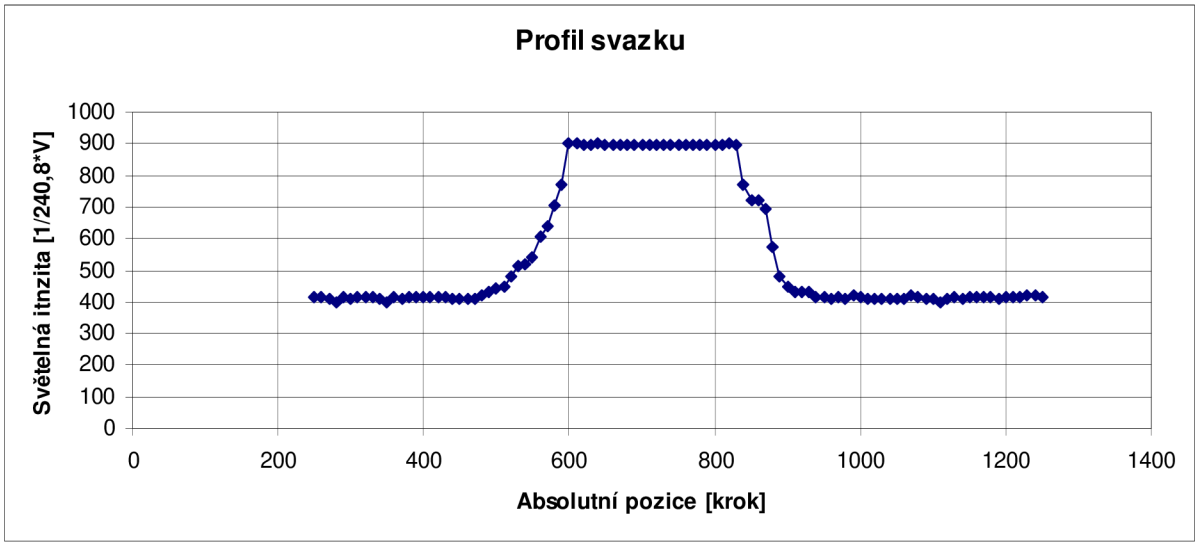


Obr. 66: Data importovaná do tabulkového editoru.

Naměřená data lze importovat do tabulkového editoru a zobrazit si je jako graf.

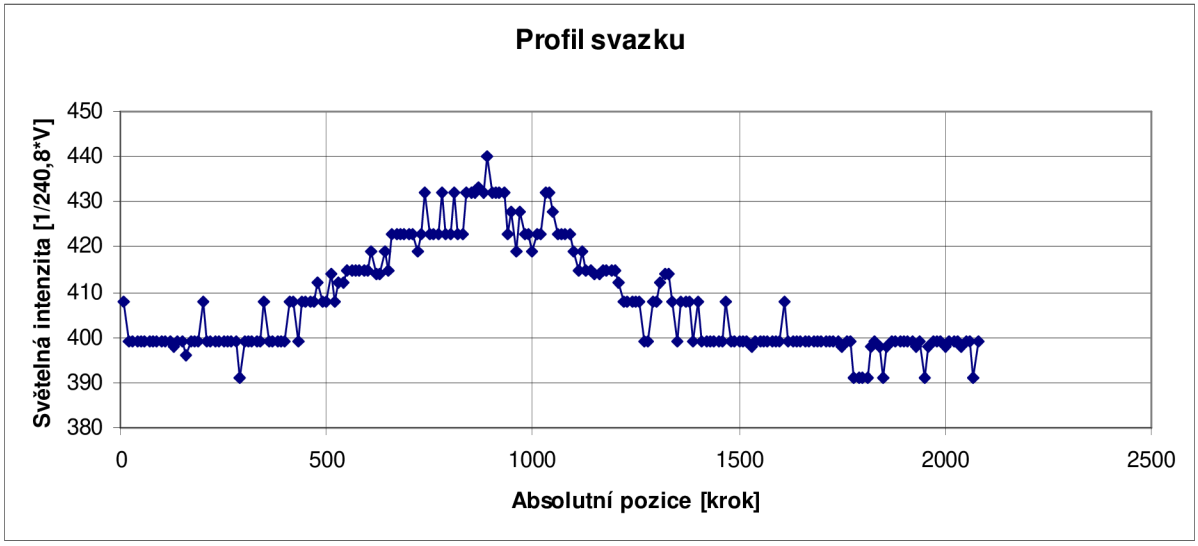
5.8 Kontrolní měření

Měření byla prováděna hlavně kvůli nastavení detektoru. V případě, že zesílení bylo příliš vysoké, docházelo k saturaci detektoru.



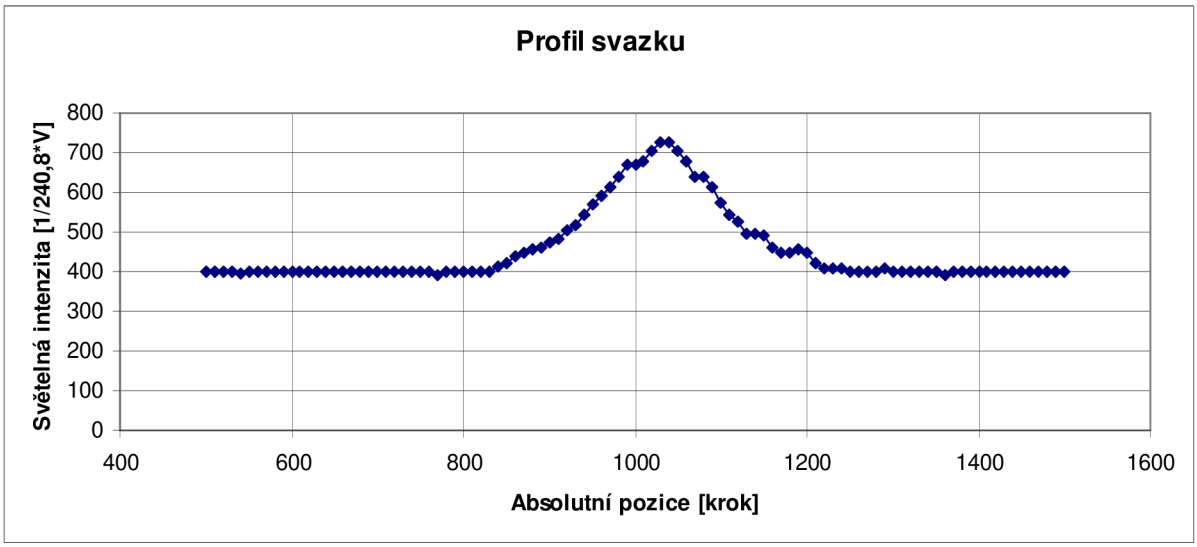
Graf 1: Závislost světelné intenzity na pozici (detektor v saturaci).

V opačném případě, kdy je zesílení příliš nízké, ztrácíme rozlišení, které je AD-převodník v mikroprocesoru schopen nabídnout.



Graf 2: Závislost světelné intenzity na pozici (malé zesílení).

Ideální nastavení detektoru a jeho zesílení určují dva trimry, jeden pro hrubé nastavení (250KΩ) a druhý pro jemné (10KΩ). Jejich hodnota pro tento stav je nastavena na 42,6KΩ.



Graf 3: Závislost světelné intenzity na pozici (R1+R2 = 42,6KΩ).

Na grafech 1, 2, 3 je vidět stálé minimum na hodnotě přibližně 400. Tato hodnota není pevná nebo neměnná, pouze představuje přirozené světelné pozadí při měření. Při změně osvětlení v laboratoři se bude tato hodnota měnit. Jak lze této chybě předejít je popsáno v podkapitole: *5.2 Snímač intenzity světla*.

Na finální měření, kde by byla simulována úloha, nedošlo z časových důvodů. Zařízení dostalo pár dní před dokončením zásah síťového napětí a kompletně shořelo. Bylo nutné jej postavit znovu.

6 Závěr

V úvodu diplomové práce „*Řízení Automatického posuvu pro digitální zapisovač dat*“ bylo objasněno několik základních teorií a informací, počítaje stručnou teorii laseru, či měřící metodu, jež byla použita k měření pološířky laserového svazku. Uvedené informace slouží pro lepší orientaci v následujících kapitolách a k pochopení zvoleného řešení.

V kapitole *3. Návrh programu* a všech jejích podkapitolách bylo navrženo počítačové ovládací grafické rozhraní pro snadnou a rychlou obsluhu při měření. Bohužel zde stále dochází k chybám, z toho důvodu je ovládání celého zařízení prozatím prováděno pomocí programu Terminál. Jak se zařízením komunikovat je podrobně popsáno v podkapitole *5.7. Oživení ovládání.*

Úkolem práce bylo uvedené zařízení vyrobit a zprovoznit. Tento proces se neobejde bez návrhu a zjištění parametrů jednotlivých součástí, čemuž se věnuje kapitola *4. Návrh zařízení*. Podrobně je zde popsáno, jakým způsobem lze řídit stejnosměrný motor (metoda, směr i rychlost), jakým způsobem určovat polohu a jaké periferie a metody používá řídící obvod.

Oddíl *5. Obvodová realizace* se zabývá návrhy výroby jednotlivých bloků: Napájení, Detektor, komunikační obvody a Řídící obvod. Jsou zde uvedeny návrhy desek plošných spojů a rozložení součástek pro každý jednotlivý funkční blok i pro celkové výsledné zapojení. Závěrem je pozornost věnována osazení a oživení měřícího zařízení, jak ve stručných krocích proběhala. Je zde podrobně popsán způsob ovládání a získávání dat, přičemž uvedená kontrolní měření byla prováděna z důvodu/za účelem nastavení detektoru.

Cílem diplomové práce bylo sestavit a naprogramovat měřící zařízení pro účel automatického měření pološířky laserového svazku. Na základě uvedených dat je zřejmé, že cíl byl bezmála splněn, výjimku tvoří komunikace s ovládacím rozhraním, jež se nepodařilo doladit. V tuto chvíli je tedy potřeba používat řešení v podobě programu Terminál.

Další krokem ke zdokonalení sestaveného zařízení je doladit komunikaci s ovládacím rozhraním a plně automatizovat měření.

7 Použitá literatura

[1] http://elektrika.cz
[2] Matoušek David:.USB prakticky, Praha, 2006. BEN, ISBN 80-7300-103-9
[3] http://www.servo-drive.com
[4] Katalogový list k Atmega8
[5] Katalogový list k MAX232
[6] Katalogový list k BWR34

8 Seznam příloh

Příloha 1 Verification_of_Comunication
Příloha 2 Soft v1_6

9 Přílohy

Příloha 1
Verification_of_Comunication

```
#include <avr/io.h>
#include <avr/iom8.h>
#define F_CPU 8000000                //Internal clock source Frequency 8MHz
#define BAUDRATE 19200               //BaudRate 19200
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

char recieved_instruction[6];

unsigned char USART_Receive( void )
        {
        /* Wait for data to be received */
        while ( !(UCSRA & (1<<RXC)) );
        /* Get and return received data from buffer */
        return UDR;
        }

void USART_Init( unsigned int baudvalue )
        {
        /* Set baud rate */
        int baud;
        baud=((F_CPU/8)/(2*baudvalue))-1;
        UBRRH = (unsigned char)(baud>>8);
        UBRRL = (unsigned char)baud;
        /* Enable Receiver and Transmitter and interupt when recive is completed*/
        UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
        /* Set frame format: 1stop bit, 8data, without parity check */
        UCSRC =
(1<<URSEL)|(0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0)|(0<<UMSEL)|(0<<UPM
1)|(0<<UPM0)|(1<<UCPOL);
        }

void USART_Transmit( char data )
        {
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );
        /* Put data into buffer, sends the data */
        UDR = data;
        }

ISR(USART_RXC_vect)
```

```
        {
        /* Recived character is send back to PC */
        recieved_instruction[s] = USART_Receive();
        USART_Transmit_Text(recieved_instruction);
        USART_Transmit('\n');
        }

int main(void)
        {
        /* Port C init(control LED output) */
        DDRC=0b00111000;

        /* Usart init, Set RXD and TXD pin */
        USART_Init(BAUDRATE); //Set Baud rate to 19200
        DDRD=0b00000010;

        /* Enable Global Iterrupt Flag */
        sei();


        while(1)
                {
                _delay_ms(200);
                PORTC |=(1<<PC3);
                _delay_ms(200);
                PORTC &=~(1<<PC3);
                }

}
```

Příloha 2
Soft v1_6

```
#include <avr/io.h>
#include <avr/iom8.h>
#define F_CPU 8000000                //Internal clock source Frequency 8MHz
#define BAUDRATE 19200               //BaudRate 19200
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int scale = 7;
int s = 0;
char recieved_instruction[6];;
volatile unsigned char move = 0;
volatile unsigned char right_border = 0;
volatile unsigned char left_border = 0;
volatile unsigned int temp = 0;
volatile unsigned int position = 0;
volatile unsigned int newposition = 0;

unsigned char USART_Receive( void )
        {
        /* Wait for data to be received */
        while ( !(UCSRA & (1<<RXC)) );
        /* Get and return received data from buffer */
        return UDR;
        }

void USART_Init( unsigned int baudvalue )
        {
        /* Set baud rate */
        int baud;
        baud=((F_CPU/8)/(2*baudvalue))-1;
        UBRRH = (unsigned char)(baud>>8);
        UBRRL = (unsigned char)baud;
        /* Enable Receiver and Transmitter and interupt when recive is completed*/
        UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
        /* Set frame format: 1stop bit, 8data, without parity check */
        UCSRC =
(1<<URSEL)|(0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0)|(0<<UMSEL)|(0<<UPM
1)|(0<<UPM0)|(1<<UCPOL);
        }

void USART_Transmit( char data )
        {
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );
```

```
	/* Put data into buffer, sends the data */
	UDR = data;
	}

void USART_Transmit_Text( char textdata[30] )
	{
	int i = 0;
	/* Wait for empty transmit buffer */
	while ( !( UCSRA & (1<<UDRE)) );
	/* Put data into buffer, sends the data */
	while(textdata[i] != '\0')
		{
		USART_Transmit(textdata[i]);
		i++;
		}
	}

void USART_Transmit_Integer( int i )
	{
	char intdata[32];
	itoa(i, intdata, 10);
	USART_Transmit_Text(intdata);
	}

void TIM16_WriteICR1( unsigned int i )
	{
	unsigned char sreg;
	/* Save Global Interrupt Flag */
	sreg = SREG;
	/* Disable interrupts */
	cli();
	/* Set ICR1 to i */
	ICR1 = i;
	/* Restore Global Interrupt Flag */
	SREG = sreg;
	sei();
	}

void align_to_zero(void)
	{
	TIM16_WriteICR1(2730);
	/* Wait for external iterrupt(right border) */
	while(right_border == 0)
		{
		PORTC |=(1<<PC5);
		PORTC &=~(1<<PC4);
		_delay_ms(50);
		PORTC |=(1<<PC4);
		PORTC |=(1<<PC5);
		_delay_ms(3);
```

```
                }
        TIM16_WriteICR1(2);
        right_border = 0;
        /* Reset of general position counter */
        position = 0;
        TCNT1 = 0;
        }

void move_to(unsigned int i)
        {
        newposition = i;
        /* Move left */
        if(newposition > position ) //Pro posuv vlevo
                        {
                        /* Set border(maximal value) */
                        if(newposition > 2730) newposition = 2730;
                        temp = newposition - position;
                        TIM16_WriteICR1(temp);
                        position = newposition;
                        move = 0;
                        while(move == 0)
                                {
                                PORTC |=(1<<PC4);
                                PORTC &=~(1<<PC5);
                                _delay_ms(50);
                                PORTC |=(1<<PC4);
                                PORTC |=(1<<PC5);
                                _delay_ms(3);
                                }
                        PORTC |=(1<<PC4);
                        PORTC |=(1<<PC5);
                        TCNT1 =       0;
                        }
        /* Move right */
        if(position > newposition ) // Pro posuv vpravo
                        {
                        /* Set border(minimal value) */
                        if(newposition < 0) newposition = 0;
                        temp = position - newposition;
                        TIM16_WriteICR1(temp);
                        position = newposition;
                        move = 0;
                        while(move == 0)
                                {
                                PORTC |=(1<<PC5);
                                PORTC &=~(1<<PC4);
                                _delay_ms(50);
                                PORTC |=(1<<PC4);
                                PORTC |=(1<<PC5);
                                _delay_ms(3);
```

```
                        }
                    PORTC |=(1<<PC4);
                    PORTC |=(1<<PC5);
                    TCNT1 = 0;
                    }
        }

void mesure(unsigned int i)
        {
        /* Set mesure scale */
        TIM16_WriteICR1(i);
        int a = 0;
        while(left_border == 0)
                {
                a = a + i;
                move_to(a);
                /* Get and return AD_value */
                int AD_value = 0;
                int AD_temp = 0;
                ADCSRA |=(1<<ADSC);
                while (ADCSRA & 0x40);
                _delay_ms(10);
                AD_temp = ADCL;
                AD_value = ADCH;
                AD_value = (AD_value<<8);
                AD_value |=AD_temp;
                /* Transmit position and related AD_value */
                USART_Transmit_Integer(position);
                USART_Transmit(';');
                USART_Transmit_Integer(AD_value);
                USART_Transmit('\n');
                }
        PORTC |=(1<<PC4);
        PORTC |=(1<<PC5);
        left_border = 0;
        }

ISR(USART_RXC_vect)
        {
        int rx_temp;
        /* Recive 5 character instruction */
        recieved_instruction[s] = USART_Receive();
        s++;
        if (s == 5)
                {
                int a = 10000;
                scale = 0;
                s = 0;
                for(int i = 1; i < 6; i++)
                        {a = a/10;
```

```
                        rx_temp = recieved_instruction[i];
                        rx_temp = rx_temp - 48;
                        rx_temp = rx_temp * a;
                        scale = scale + rx_temp;
                        }
                //USART_Transmit(recieved_instruction[0]);
                //USART_Transmit_Integer(scale);
                USART_Transmit_Text(recieved_instruction);
                USART_Transmit('\n');
                }

        }

ISR(INT0_vect)
        {
        if(bit_is_clear(PIND, 4))
                {
                /* Stop motor on left border */
                PORTC |=(1<<PC4);
                PORTC |=(1<<PC5);
                _delay_ms(1000);

                PORTC &=~(1<<PC4);
                PORTC &=~(1<<PC5);
                left_border = 1;
                move++;
                }

        if(bit_is_set(PIND, 4))
                {
                /* Stop motor on right border */
                PORTC |=(1<<PC4);
                PORTC |=(1<<PC5);
                _delay_ms(1000);

                PORTC &=~(1<<PC4);
                PORTC &=~(1<<PC5);
                right_border = 1;
                move++;
                }
         _delay_ms(1);
        }

ISR(TIMER1_CAPT_vect)
        {
        /* Stop motor Timer1 overflow */
        PORTC |=(1<<PC4);
        PORTC |=(1<<PC5);
        TCNT1 = 0;
        move++;
```

```
	//PORTC &=~(1<<PC4);
	//PORTC &=~(1<<PC5);
	}

int main(void)
	{
	/* Port C init(control LED output) */
	DDRC=0b00111000;
	DDRB=0b00001000; //T1 vstupni pin pro citac T1

	/* Usart init, Set RXD and TXD pin */
	USART_Init(BAUDRATE); //nastavuje Baud rate na 19200
	DDRD=0b00000010;

	/* Enable external interrupt INT0 on Faling Edge sensitivity */
	MCUCR |=(1<<ISC01)|(0<<ISC00); // nastavuje reakci External Interupt INT0 na
Faling edge.
	GICR |=(1<<INT0);   //Povoluje External Interupt INT0

	/* Enable PWM mode on Timer2, Set prescaler to 256, Output on pin OC2 */
	TCCR2 |=(1<<WGM20)|(0<<CS22)|(1<<CS21)|(1<<CS20)|(1<<COM21);
	OCR2 = 90;

	/* Enable CTC mode on Timer1, Set External Clock Source on pin T1(Faling edge),
Enable overflow interrupt */
	TCCR1B |=(1<<CS12)|(1<<CS11)|(0<<CS10);
	TCCR1A |=(1<<WGM11)|(0<<WGM10);
	TCCR1B |=(1<<WGM13)|(1<<WGM12);
	ICR1 = scale;
	TIMSK |=(1<<TICIE1)|(0<<OCIE1A)|(0<<OCIE1B);

	/* Set supply voltage like reference, Result adjusted to left, Enable AD converson */
	ADMUX |=(0<<REFS1)|(1<<REFS0);
	ADMUX |=(0<<ADLAR);
	ADCSRA |=(1<<ADEN);


	/* Enable Global Iterrupt Flag */
	sei();

	/* Move to startup position 0 */
/*	if(bit_is_clear(PIND, 4))
			{
			align_to_zero();
			}
 */

	while(1)	//nekonecna smycka ve ktere provadime blikani LED diod, pripadne
prepolovani
		// napeti na jedne LED diode (musime mit vhodnou diodu)
```

```
		{
		_delay_ms(200);
		PORTC |=(1<<PC3);
		_delay_ms(200);
		PORTC &=~(1<<PC3);

		if(s == 0)
			{
			switch(recieved_instruction[0])
				{
				case 'w':
				move_to(scale);
				recieved_instruction[0] = 's';
				break;

				case 'm':
				mesure(scale);
				recieved_instruction[0] = 's';
				break;

				case 'a':
				align_to_zero();
				recieved_instruction[0] = 's';
				break;
				}
			}

		}

}
```