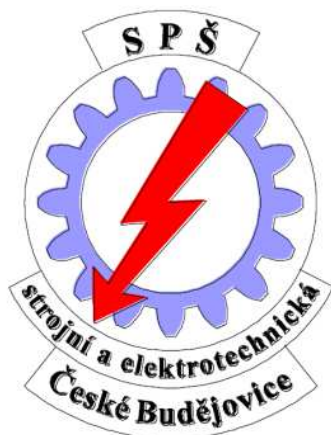


**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA STROJNÍ A ELEKTROTECHNICKÁ  
ČESKÉ BUDĚJOVICE, DUKELSKÁ 13**



**NÁVRH A REALIZACE ROBOTA SLEDUJÍCÍHO ČERNOU ČÁRU  
MATURITNÍ PRÁCE**

Autor práce: **Lukáč Čížek**  
Vedoucí práce: **Bc. Jakub Kákona**  
Oponent: **Ing. Jan Janoud**  
  
Školní rok: **2012/2013**

Prohlašuji, že jsem maturitní práci s názvem: "*Návrh a realizace robota sledujícího černou čáru*" vypracoval samostatně pod vedením Bc. Jakuba Kákony s použitím literatury uvedené na konci mé maturitní práce v seznamu použité literatury.

V Českých Budějovicích dne 26.února 2013

.....  
*Lukáš Čížek*

## **Zadání odborné maturitní práce**

Zadání je vystaveno pro žáka:

Jméno a příjmení: **Lukáš Čížek**  
Datum narození: **31. 1. 1994**  
Žák třídy: **4.EA ve šk. r. 2012 - 2013**

Téma maturitní práce:

### **Návrh a realizace robota sledujícího černou čáru**

Požadované dílčí body práce:

1. návrh a realizace modulu pro LiOn akumulátor, který zajistí jeho ochranu a dobíjení
2. vhodný výběr senzorů pro sledování čáry, návrh modulu a jeho realizace
3. návrh a realizace modulu H-můstků pro řízení motorů
4. návrh a realizace podvozku
5. výběr vhodného MCU pro robota
6. napsání programu pro vybraný procesor

Cíl maturitní práce:

Návrh, konstrukce, implementace programu pro sledování čáry a zpracování dokumentace. Zajištění kompatibility všech součástí a modulů s opensource modulární stavebnicí MLAB

**Rozsah práce:** minimálně 20 stran textu + elektrické schéma

**Termín odevzdání práce:** do 21. dubna 2013 ve dvou vyhotoveních

**Vedoucí práce:** Bc. Jakub Kákona

**Oponent:** Ing. Jan Janoud

**Kritéria hodnocení:** Původnost, technická úroveň návrhu, formální úroveň zprávy, úroveň ústní prezentace při obhajobě práce

V Českých Budějovicích 19. 12. 2013

Ing. Čestmír Tschauer,  
ředitel školy

Chtěl bych poděkovat všem, kteří mi umožnili realizovat tuto práci, zvláště vedoucímu mé maturitní práce Bc. Jakubu Kákonovi, dále pak mému oponentovi Ing. Janu Janoudovi, učiteli Ing. Oldřichu Smutnému, ale také mým přátelům, spolužákům a rodině za inspiraci i jejich trpělivost.

Konstrukce robota byla realizována z příspěvku Střední průmyslové školy strojní a elektrotechnické v Českých Budějovicích a z prostředků firmy Universal Scientific Technologies s.r.o. Za tyto příspěvky mnohokrát děkuji.

### **Abstrakt**

Práce se zabývá rozbořem problémů vznikajících při úkolu sledování černé čáry, jejich řešením a výslednou konstrukcí robota včetně napsání programu. Tato úloha je poměrně častou výukovou metodou sensoriky a autonomního řízení. Lze se s ní také setkat v různých robotických soutěžích po celém světě.

**Klíčová slova:** sledování čáry, sensorika, autonomní řízení, autonomní robot, řádkový senzor

# OBSAH

<b>1.</b>	<b>Úvod</b>	<b>1</b>
<b>2.</b>	<b>Rozbor problémů a jejich řešení</b>	<b>2</b>
2.1.	Napájení	2
2.1.1.	<i>Akumulátory</i>	2
2.1.2.	<i>Stabilizace napájecího napětí</i>	3
2.2.	Buzení motorů	5
2.2.1.	<i>Stejnoseměrné komutátorové motory</i>	5
2.2.2.	<i>H-můstky</i>	6
2.2.3.	<i>Rušení</i>	7
2.3.	Senzory	8
2.3.1.	<i>Řádkový snímač</i>	8
2.3.2.	<i>Odrazový senzor</i>	9
2.3.3.	<i>Dotykový senzor</i>	10
2.4.	Výběr procesoru	11
2.4.1.	<i>Procesor PIC16F887</i>	11
<b>3.</b>	<b>Blokové schéma robota</b>	<b>12</b>
<b>4.</b>	<b>Řídící program</b>	<b>13</b>
4.1.	Vývojový diagram	13
4.2.	Popis programu	14
4.2.1.	<i>Indikace</i>	14
4.2.2.	<i>Řízení motorů</i>	15
4.2.3.	<i>Čtení analogových senzorů</i>	16
4.2.4.	<i>Práce s optickým řádkovým snímačem</i>	16
4.2.5.	<i>Vyhodnocení</i>	20
4.2.6.	<i>Řídící část programu a diagnostika</i>	22
<b>5.</b>	<b>Konstrukce</b>	<b>23</b>
5.1.	Popis mechanické konstrukce	23
5.2.	Fotografie robota	24
<b>6.</b>	<b>Závěr</b>	<b>25</b>
<b>7.</b>	<b>Použitá literatura</b>	<b>26</b>
<b>8.</b>	<b>Obsah příloženého CD</b>	<b>27</b>
<b>9.</b>	<b>Příloha A - Dokumentace navržených modulů</b>	<b>28</b>
9.1.	LION2CELL01A	
9.2.	HBRDGL29801A	
9.3.	LINREG01A	
9.4.	OLSA01A	
9.5.	LEDSENBAR01A	

# 1. ÚVOD

Robotické aplikace se stále více vyskytují v průmyslu, což je důvodem, proč se rozvíjí na středních odborných školách výuka automatizace, robotiky a mikroprocesorové techniky, a tak lze na středních i vysokých školách narazit krom teoretické výuky i na výuku praktickou, což znamená, že je nutné do škol poskytnout potřebné vyučovací pomůcky.

Právě robot, který bude sledovat černou čáru, je typickou výukovou aplikací z oblasti sensoriky a samočinného řízení a proto by mohl dobře posloužit i jako pomůcka pro výuku.

Tato práce si neklade za cíl vytváření nových poznatků z oblasti sensoriky a řízení, ale snaží se o výběr a aplikaci dosud známých způsobů měření a řízení tak, aby bylo možné na základě zkonstruovaného prototypu robota, dokumentace a získaných poznatků sestavit spolehlivé a rychlé roboty pro sledování čáry.

## 2. ROZBOR PROBLÉMŮ A JEJICH ŘEŠENÍ

Základním problémem bylo vyřešit, jaký druh baterie zvolit pro napájení, jak zajistit její ochranu a dobíjení, a jakým způsobem realizovat stabilizaci napětí 5V pro napájení logiky, tedy procesoru a logické části H-můstku a senzorů.

### 2.1. NAPÁJENÍ

#### 2.1.1. AKUMULÁTORY

V dnešní době lze na trhu zakoupit nepřeberné množství akumulátorů elektrické energie různých typů a odlišných provedení. Při výběru vhodného akumulátoru jsem se rozhodoval podle nominálního napětí článku, počtu nabíjecích cyklů článku, kapacity, ale vzhledem k tomu, že se jedná o pojezdného robota, rozhodovala při výběru i hmotnost a geometrické rozměry.

Vzhledem k dostupnosti jsem se nakonec rozhodoval mezi NiMh (Nikl - metal hybridový) a LiOn (Lithium - iontový). Jejich vlastnosti jsou stručně popsány níže.

Výhodnější se nakonec ukázalo použít Li-Ion akumulátor, byť je pro jeho použití potřeba navrhnout ochranné obvody, protože při nesprávném zacházení, jako je např. přebíjení nebo zkrat, může dojít k jeho vznícení a explozi.

#### NiMh akumulátory

NiMh články mají poměrně nízké jmenovité napětí 1,2 V. Na plně nabitém článku je napětí až 1,4 V, na vybitém 1,0 V. Životnost akumulátorů se pohybuje do 1000 nabíjecích cyklů. Na další stránce jsou uvedeny základní vlastnosti těchto akumulátorů

##### Výhody:

- bezpečnost provozu (*oproti Li-Ion akumulátorům*)
- jednoduchost aplikace (*pro většinu aplikací odpadá potřeba obvodů pro kontrolu a ochranu baterie*)

##### Nevýhody:

- malá kapacita (*proto články mají větší rozměry a hmotnost*)
- malé jmenovité napětí (1,2 V)
- samovybíjení (*až 30 % za měsíc*)
- nelze je vybíjet a nabíjet velkými proudy (*může dojít ke zničení akumulátoru*)

#### Li-Ion akumulátory

Li-Ion akumulátory se dnes běžně používají ve spotřební elektronice, zejména v noteboocích a mobilních telefonech. Jmenovité napětí je přibližně 3,7 V a vydrží až 2000 nabíjecích cyklů.

##### Výhody:

- vysoká hustota energie (*mají velkou kapacitu i při malé hmotnosti a rozměrech*)
- netrpí samovybíjením (*pod 5 % za měsíc*)
- nemají paměťový efekt (*způsobuje náhlý pokles napětí při zatížení, pokud akumulátor nebyl před nabitím plně vybit*)



## Nevýhody:

- stárnutí (*baterie pomalu ztrácí kapacitu i bez ohledu, zda je používána či nikoliv*)
- nebezpečí výbuchu nebo vznícení při špatném zacházení (*nutnost ochranných obvodů*)
- nesmí se úplně vybit (*při napětí pod 3 V je velmi těžké je zpět oživit - ztratí kapacitu*)

Pro použití Li-Ion akumulátoru jsem provedl návrh modulu LION2CELL01A, jehož dokumentace je uvedena v kapitole "Dokumentace navržených modulů".

## 2.1.2. STABILIZACE NAPĚTÍ

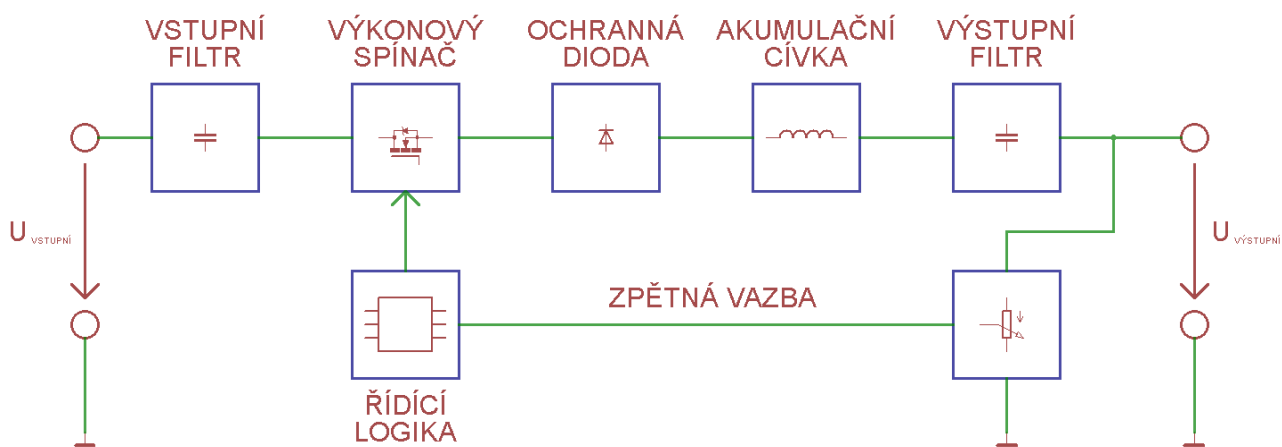
Pro logické části elektroniky je nutné získat odrušené a stabilizované napětí. Jeho hodnota se pohybuje do 5V. Toho můžeme nejlépe dosáhnout dvěma způsoby. Prvním je použití spínaného zdroje, tím druhým je použití lineárního stabilizátoru.

Jako výhodnější varianta se nakonec ukázal lineární stabilizátor, protože jej lze lehce zkonstruovat, nebude generovat rušení a pro malé odběry (na robotovi maximálně 500 mA) se nevyplatí návrh a konstrukce spínaného zdroje.

## Spínaný zdroj

Spínané zdroje využívají cívek a kondenzátorů k akumulaci elektrické energie. Na vysokých frekvencích je připojují pomocí elektronického spínače (nejčastěji tranzistorů MOSFET) a odpojují k napájecímu napětí, čímž se v nich akumuluje energie, která je poté čerpána do zátěže. Elektronický spínač je řízen zpětnou vazbou, která snímá výstupní napětí, čímž se zajistí stabilní výstupní napětí.

Pro robota by bylo nejvýhodnější sestavit snižující měnič bez transformátoru, který by byl sestaven podle blokového schématu, které je níže.



*Blokové schéma spínaného snižujícího měniče bez transformátoru*

**Výhody:**

- malé ztráty
- při velkých výkonech malé rozměry

**Nevýhody:**

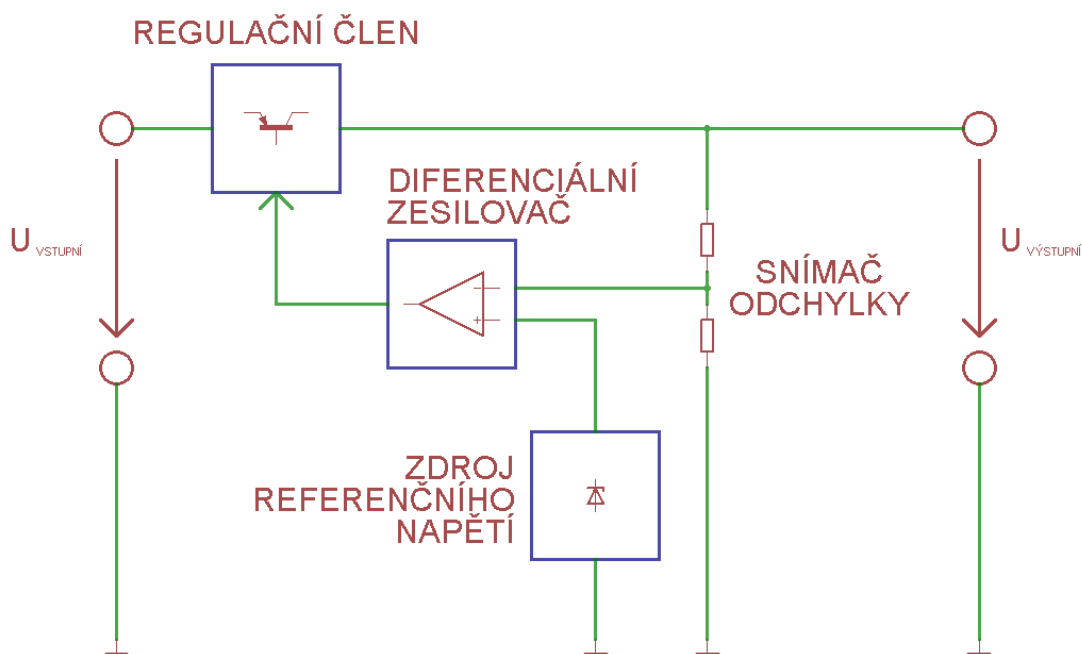
- rušení, které vzniká při spínání
- složité zapojení (oproti lineárním stabilizátorům)

**Lineární stabilizátor**

Lineární stabilizátory pracují na jednoduchém principu. Skládají se z akčního prvku (nejčastěji bipolárního tranzistoru) a z řídicí logiky (diferenciálního zesilovače a zdroje referenčního napětí).

Diferenciální zesilovač sleduje odchylku výstupního napětí a napětí referenčního. Výstup zesilovače budí regulační tranzistor. Dojde-li ke snížení výstupního napětí (např. vlivem zvýšení odběru), zvýší se odchylka a diferenciální zesilovač otevře více tranzistor, čímž se sníží úbytek napětí, který na něm vzniká a výstupní napětí vzroste.

Pevné lineární stabilizátory se dělají např. v třívorkovém provedení s označením 78XX nebo 79XX (XX označuje výstupní napětí). Nastavitelné lineární stabilizátory jsou označovány např. jako LM317.



*Blokové schéma lineárního stabilizátoru*

**Výhody:**

- jednoduchost zapojení
- nevzniká rušení

**Nevýhody:**

- velké ztráty
- pro velké výkony je nutnost chlazení

Pro použití lineárního stabilizátoru jsem navrhl modul LINSTAB01A, jehož dokumentace je uvedena v kapitole "Dokumentace navržených modulů".

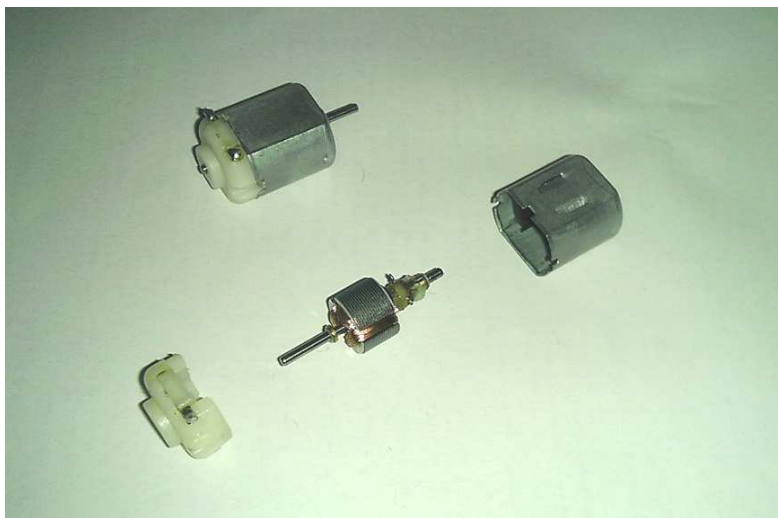
## 2.2. BUZENÍ MOTORŮ

K tomu, aby se robot mohl pohybovat jsem se rozhodl použít modelářské stejnosměrné motory s převodovkou. Vybral jsem je proto, protože je poměrně jednoduché je řídit, ale také je důležitým faktem, že mají nízkou hmotnost, což umožní konstrukci robota menší hmotnosti, který poté může dosahovat vyšší rychlosti.

### 2.2.1. STEJNOSMĚRNÉ KOMUTÁTOROVÉ MOTORY

Stejnoseměrný motor se skládá ze dvou důležitých částí. Tou první je stator a druhou rotor. Na statoru jsou umístěny dva opačně orientované magnety, mezi něž je vložen rotor, na kterém je navinuto několik vinutí (u našeho motoru jsou tři). Velmi důležitou částí rotoru je komutátor, který zajišťuje přepínání cívek při chodu motoru.

Pro zmenšení tření je osička motoru umístěna v bronzových kluzných ložiscích, která jsou namazána vazelínou.



Na obrázku výše lze vidět složený a rozložený motorek, které jsou použity v převodovce robota. Na obrázcích níže je detail rotoru (vlevo) a statoru (vpravo).



### **Funkce stejnosměrného komutátorového motoru:**

Na vinutí rotoru motoru je přivedeno přes komutátor napětí, které způsobí to, že cívkou (představující část vinutí) začne protékat proud, který vytvoří magnetické pole. Poté začne být tato cívka přitahována směrem k magnetu s opačnou polaritou, což způsobí rotační pohyb rotoru. Před momentem, kdy se od magnetu začne cívka oddalovat (vlivem setrvačnosti), přepojí komutátor napětí na další cívku, aby nedocházelo k brzdění pohybu (vlivem magnetické síly, která by zpět přitahovala cívku k magnetu).

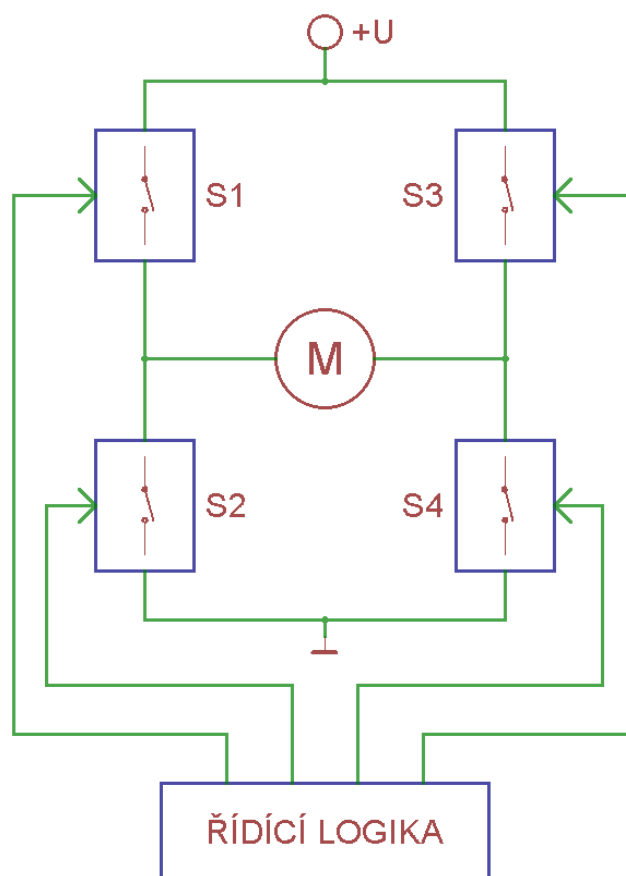
Tento děj se neustále opakuje a to způsobí otáčení hřídelky, na níž je připojena převodovka.

### **2.2.2. H - MŮSTKY**

Vzhledem k tomu, že motory nelze budít přímo z výstupu mikroprocesoru, musíme pro jeho spínání použít minimálně jeden tranzistor. Tímto způsobem však nelze měnit směr otáčení motoru. K tomu slouží tzv. H-můstek.

H-můstek je elektronický obvod, který slouží k přepínání pólů (a tedy i ke změně směru rotace) u stejnosměrných motorů. Dnes se lze setkat s H-můstkou realizovanou pomocí bipolárních nebo unipolárních tranzistorů, popř. může být celý můstek (nebo i více můstků) v jednom integrovaném obvodu.

H můstek si můžeme představit realizovaný pomocí čtyř elektronických spínačů, jak jsem naznačil na obrázku níže.



*Principiální schéma H-můstku*

H-můstky většinou pracují tak, že se spínají vždy dva spínače v jedné diagonále (v našem případě spínače S1 a S4 nebo S2 a S3). Při spínání motoru může dojít k pěti stavům:

1. Motor je zcela odpojen (*žádný ze spínačů není sepnutý*)
2. Motor se točí jedním směrem (*jsou sepnuté spínače S1 a S4*)
3. Motor se točí druhým směrem (*jsou sepnuté spínače S2 a S3*)
4. Motor je brzděn (*spínače S2 a S3*)
  - z motoru je odebírána elektrická energie (*motor je zkratován*)
5. H-Můstek zkratován (*všechny spínače jsou sepnuté*)
  - nežádoucí stav, spínacími tranzistory tečou velké proudy a může dojít ke zničení H-můstku

### **Řízení rychlosti otáčení stejnosměrných komutátorových motorů:**

Řízení rychlosti motorů lze dosáhnout pomocí pulzní šířkové modulace (PWM = pulse width modulation), která mění střídu signálu.

Lze si ji představit tak, že v průběhu jedné periody motor přepneme ze stavu zapnuto na vypnuto a naopak, přičemž se mění poměr mezi dobou, kdy je motor zapnutý a kdy vypnutý. Vzhledem k vysoké frekvenci těchto impulzů se motor točí plynule. Čím delší dobu je motor v jedné periodě sepnut, tím větší jsou otáčky motoru.

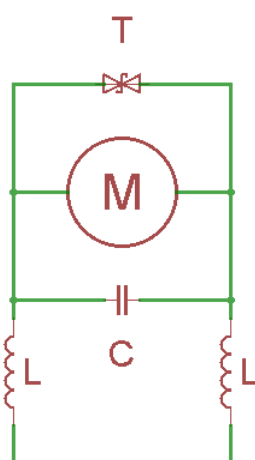
Pro H-můstek jsem navrhl modul HBRDGL29801A, jehož dokumentace je uvedena v kapitole "Dokumentace navržených modulů".

### 2.2.3. RUŠENÍ

Jelikož je motor indukční zátěž, vzniká při chodu motoru rušení, které může mít za následek buď špatnou funkci řídicí jednotky, ovlivnit funkci senzorů a dalších zařízení, ale dokonce může mít za následek i zničení některých prvků.

Při chodu motoru dochází ke vzniku napěťových a proudových špiček, za což může přepínání cívek vinutí komutátorem. Pro jejich odstranění jsem k motoru připojil obousměrný transil, blokovací kondenzátor a dvě cívky.

Transil tvoří přepětovou ochranu, protože při překročení napětí mezi jeho svorkami dojde k jeho sepnutí a zkratování motoru. Blokovací kondenzátor blokuje menší napěťové špičky, na které ještě transil nereaguje. Dvojice cívek zapojených do série s motorem potlačují vznik proudových špiček.



*Odrušení stejnosměrného komutátorového motoru*

### 2.3. SENZORY

Senzory slouží robotovi k tomu, aby získal alespoň základní přehled o okolním prostředí nebo aby sledoval potřebné parametry. Při detekci a měření různých veličin se však setkáváme s problémy, které je potřeba řešit. V našem případě se toto týkalo zejména černé čáry, po níž má robot jezdit.

Pokusy jsem zjistil, že různé materiály různých barev se v infračerveném spektru jeví jinak, než je tomu ve spektru, které vidí člověk. Z tohoto důvodu jsem se rozhodl použít ke sledování čáry optické odrazové senzory, které reagují na viditelné spektrum, místo původní infračervených senzorů..

Toto řešení však přineslo několik komplikací. Je nutné pořídit filtry barev nebo přímo snímací prvky (nejčastěji fototranzistory a fotodiody), které reagují pouze na vhodnou barvu, aby se zamezilo rušení snímání vlivem okolního světla. O řešení jednotlivých senzorů pro snímání čáry se rozepíši v dalších podkapitolách.

Krom detekce čáry je potřeba také zajistit alespoň minimální interakci se člověkem. Z tohoto důvodu jsem na robota přidělal nárazník. K čemu na robotovi je a jak funguje, jsem více rozepsal v podkapitole "*Dotykový senzor*".

**Použité senzory:** optický řádkový snímač  
dva odrazové senzory reagující na modré světlo  
dva dotykové senzory tvořící nárazník

### 2.3.1. ŘÁDKOVÝ SNÍMAČ

Ke snímání čáry je nejlepší mít k dispozici co nejvíce snímacích bodů. Nejprve jsem chtěl použít fototranzistory s filtry světla, ale ukázalo se, že integrovaný AD převodník v procesoru je velmi pomalý, protože by přečtení deseti senzorů trvalo přibližně 100 ms, což je poměrně dlouhá doba a k tomu je potřeba ještě implementovat vyhodnocení a řízení motorů, a tak by nakonec jeden cyklus programu mohl trvat i 110 ms. Chci upozornit, že se jedná o přibližné hodnoty pro procesor PIC16F877 s taktem 8MHz.

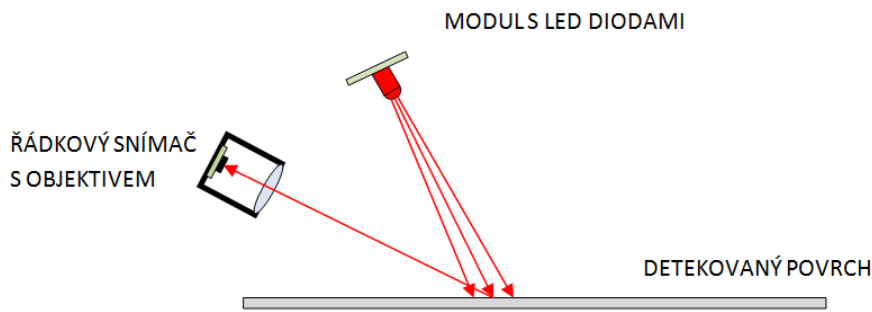
Při prohlížení různých typů senzorů jsem narazil na optické řádkové snímače (linear optical sensor array), který jsem se rozhodl použít. Jedná se v podstatě o zjednodušenou verzi kamery (má méně pixelů a pouze v jedné řadě), kterou jsem již dříve viděl na robotech na sledování čáry používat.

Pro detekci čáry jsem nakonec vybral senzor s označením TSL3301. Jedná se o řádkový senzor, který se skládá ze 102 fotodiód, vyhodnocovací logiky a interního AD převodníku. Senzor pracuje s napájecím napětím 3,0 V - 5,5 V a frekvence hodin může být až 10 MHz. Poté se přečtení celého senzoru včetně zpracování pohybuje v jednotkách ms. Senzor TSL3301 je i s veškerou vyhodnocovací logikou v integrovaném obvodu v pouzdru CL8.

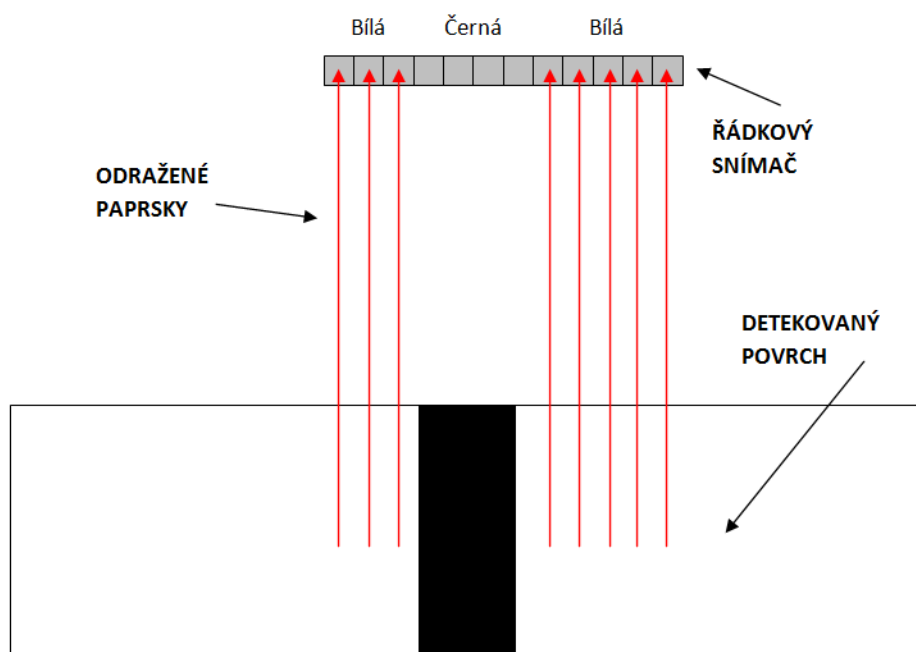
Senzor reaguje na světlo o vlnové délce 300 nm až 1100 nm, ale největší citlivost je na světlo o vlnové délce 700 nm, což odpovídá červenému světlu. Proto byl k tomuto senzoru navržen i modul na osvětlení LEDSENBAR01A. Bohužel se mi nepodařilo sehnat LED diody s vlnovou délkou 700 nm, a tak jsem použil nejbližší hodnotu - 660 nm.

Vzhledem k tomu, že osvětlení snímaného prostoru řeší sám robot, umožňuje to jízdu a správnou funkčnost i za zhoršených optických podmínek, např. v noci při zhasnutém světle.

Při použití tohoto senzoru bylo nutné vyřešit i přenos obrazu na fotodiody. Senzor jsem proto musel opatřit objektivem. K dispozici jsem měl objektiv ze scanneru čárových kódů Genius scanner 4500 A. Bylo nutné zajistit, aby při manipulaci s objektivem a modulem pro senzor TSL3301 nedošlo ke vniku prachu nebo jiných nečistot do prostoru objektivu. Senzor musí být v objektivu také uzavřen tak, aby dovnitř nevnikalo žádné světlo z okolí.



*Princip detekce řádkovým senzorem (pohled z boku)*



*Princip detekce řádkovým senzorem (pohled zepředu)*

Jak je vidět z obrázku výše, chová se řádkový senzor obdobně jako mnoho bodových senzorů v jedné linii. Pro tento senzor byl navržen modul OLSA01A. Jeho dokumentaci spolu s dokumentací k modulu LEDSENBAR01A najdete v kapitole "Dokumentace navržených modulů".

### 2.3.2. ODRAZOVÝ SENZOR

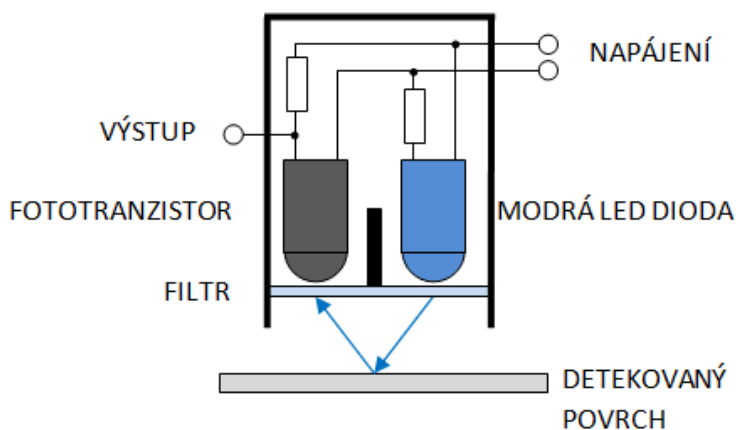
Sice jsem v podkapitole "2.3.1. Řádkový senzor" psal o nevýhodnosti použití integrovaného AD převodníku procesoru v kombinaci s použitím odrazových senzorů s analogovým výstupem.

Nicméně i přes to jsem se rozhodl použít dva odrazové senzory s analogovým výstupem na robotovi. Ty slouží v případě, že robot ztratí na hlavním senzoru - TSL3301 obraz nebo nebude vidět čáru. Jedná se v podstatě o záchranný mechanismus, který robotovi umožní opětovně najít čáru nebo v případě nefunkčnosti optického čárového senzoru, čáru sledovat.



Na robotovi se tedy nacházejí dva záchranné senzory. Pochází z inkoustové tiskárny, kde se používají ke zjištění kvality papíru. K tomu využívají viditelného světla modré barvy. Mechanická konstrukce senzoru je velmi jednoduchá a samotný senzor se skládá pouze z fototranzistoru, odporu, modré LED diody a filtru modrého světla s čočkami. Senzory jsou obaleny černou samolepící páskou, aby se zamezilo pronikání okolního světla k fototranzistoru. Zároveň izolepa slouží jako stínítko pro detekční prostor.

Detekce probíhá tak, že se měří napětí na děliči tvořeným z fototranzistoru a odporu. Mechanické uspořádání a princip detekce je znázorněn na obrázku dole.



*Konstrukce a princip detekce záchranného odrazového senzoru*

### 2.3.3. DOTYKOVÝ SENZOR

Dotykový senzor je na robotovi implementován ze dvou důvodů. Prvním je detekce nízkých překážek. Druhým důvodem je interakce s člověkem, protože při zapnutí robota si lze vybrat režim sledování čáry nebo diagnostiku pro ověření funkčnosti senzorů a motorů.

Jeden programový cyklus spustí diagnostiku, kdy dochází k výpisu všech senzorů na USB a lze pomocí toho zjistit, zda je vše v pořádku a na základě těchto dat odhalit např. vzniklou závadu. Druhý programový cyklus spustí proceduru sledování čáry. O těchto cyklech se podrobněji rozepíše v kapitole "Program".

Dotykový senzor je proveden jako nárazník v přední části robota. Skládá se ze dvou mikrospínačů, z nichž každý je na jedné straně robota, a dvou pull-up rezistorů, které přivádí na senzor napětí 5 V. Spínací tlačítka jsou spojeny s plechovým profilem, který tvoří nárazník. Tato konstrukce umožňuje zjistit, zda se jedná o čelní náraz nebo zda robot pouze o něco zavadil na jedné ze stran.

Pokud spínač není sepnutý (tj. není stisknutý nárazník), je na výstupu senzoru 0 V - logická 0 (potenciál GND). V případě, že dojde k sepnutí, přivede se přes odpor na výstup napětí 5 V - logické 1. Výstup je tedy kompatibilní s logikou TTL i CMOS. Výstup ze senzoru je přímo připojen na digitální vstup mikroprocesoru.

## 2.4. VÝBĚR PROCESORU

Po dokončení návrhu jednotlivých modulů robota bylo potřebné zvolit vhodný procesor. Plně dostačující bylo použití 8 bitového mikrokontroleru. V dnešní době na tomto poli dominuje firma Atmel s procesory ATmega a firma Microchip s procesory PIC.

K dispozici jsem měl procesor PIC16F887 (*modul PIC16F87xTQ4401B*) s USB programátorem PicKit2 (*modul PICPROGUSB02A*) a procesor ATmega 328 (*modul ATmegaTQ3201A*) s převodníkem USB na RS232 (*modul USB232R01B*).

Předem podotýkám, že ani jeden z výše zmíněných modulů v kapitole "2.4. Výběr procesoru" není mým návrhem a pouze jsem použil součásti ze stavebnice MLAB, na jejichž stránkách lze najít kompletní dokumentaci k těmto modulům.

Oba procesory jsou si velmi podobné, jak po stránce paměti, tak rychlosti, ale liší se od sebe architekturou. Nakonec jsem zvolil procesor PIC16F887, protože k němu vlastní kvalitní vývojové prostředí PCW PICC Compiler, jemuž nemůže žádné z freeware prostředí pro ATmega konkurovat.

Sice je jistou alternativou kit Arduino, který lze pomocí modulu procesoru a USB/RS232 převodníku lehce realizovat. Poté by se procesor mohl programovat v jazyce Wiring, který vychází z jazyka C, jeho nevýhodou však je obrovské plýtvání prostředky procesoru (oproti čistému C) a tak i výsledná menší rychlost chodu programu.

### 2.4.1. PROCESOR PIC16F887

Procesor PIC16f887 se doporučuje napájet napětím 3,3 až 5,0 V. Pro taktování lze použít nastavitelný interní oscilátor (v mém případě je nastaven na 8 MHz), ale lze připojit i externí oscilátor s frekvencí až 20 MHz. Cena tohoto procesoru se pohybuje kolem 90 Kč a celý modul vyjde přibližně na 200 Kč.

Procesor má 44 vývodů, z nichž 4 slouží jako napájení a další 4 nejsou použity. Pro uživatele tedy zbývá 36 vývodů. Jak tomu bývá u mikrokontrolerů, je řada těchto vývodů použita pro několik různých funkcí (mikrokontrolery v pouzdru mají krom procesoru i další pomocné obvody, jako je např. AD převodník, časovač a komparátor).

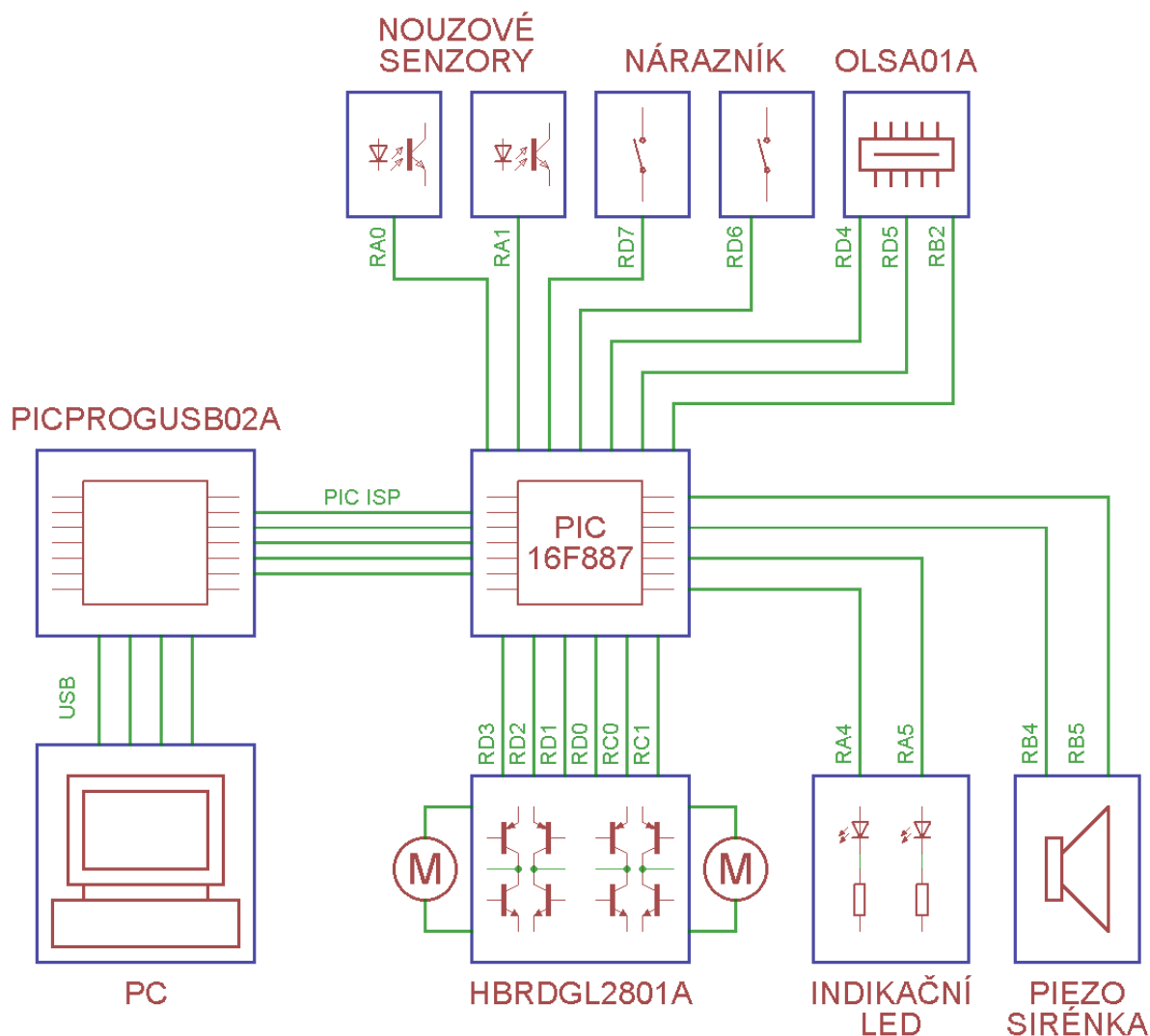
Procesor je založen na harvardské architektuře, která předpokládá oddělené prostory paměti pro program a data. Proto se paměť skládá z paměti typu FLASH (112 KB), EEPROM (256B) a RAM (368B), z něhož paměť FLASH a EEPROM jsou datové oblasti pro uložení programu a nelze k nim přistupovat přímo, ale pouze pomocí speciálních registrů, což slouží jako ochrana proti nechtěnému přepsání programu.

PIC16F887 patří mezi procesory architektury RISC, což znamená, že mají redukovanou instrukční sadu. Návrh této instrukční sady poté počítá s jednoduchými strojovými instrukcemi (např. neexistuje strojová instrukce pro násobení nebo dělení), které jsou však implementovány hardwarově, což značně zvyšuje rychlost provádění těchto instrukcí.

Program pro tento procesor byl napsán ve vývojovém prostředí PIC C Compiler od firmy PCW. Program je v jazyce C. Podrobnější popis a rozbor samotného programu je v kapitole "Program". Po zkompilování programu se program do procesoru nahrál přes programátor PICPROGUSB02A a softwaru od firmy Microchip PicKit2.

### 3. BLOKOVÉ SCHÉMA ROBOTA

Níže lze vidět blokové schéma robota. Je zda zakresleno pouze propojení jednotlivých součástí s procesorem, není zde naznačeno napájení. Logické části obvodů jsou napájeny stabilizovaným zdrojem 5V, k jehož získání slouží modul LINREG01A. Silové části jsou napájeny přímo z baterii (zde se jedná o H-můstek HBRDGL29801A a vstupní napájení modulu stabilizovaného zdroje LINREG01A).



*Blokové schéma robota*

Jako analogové vstupy jsou využity pouze vývody RA0 a RA1. Počítač se k robotovi připojuje pouze při nahrávání programu do procesoru. Za tímto účelem je také nutné propojit PICPROGUSB20A s procesorem pomocí PIC ISP (využívá signály PGC, PGD, VDD, GND a MCLR).

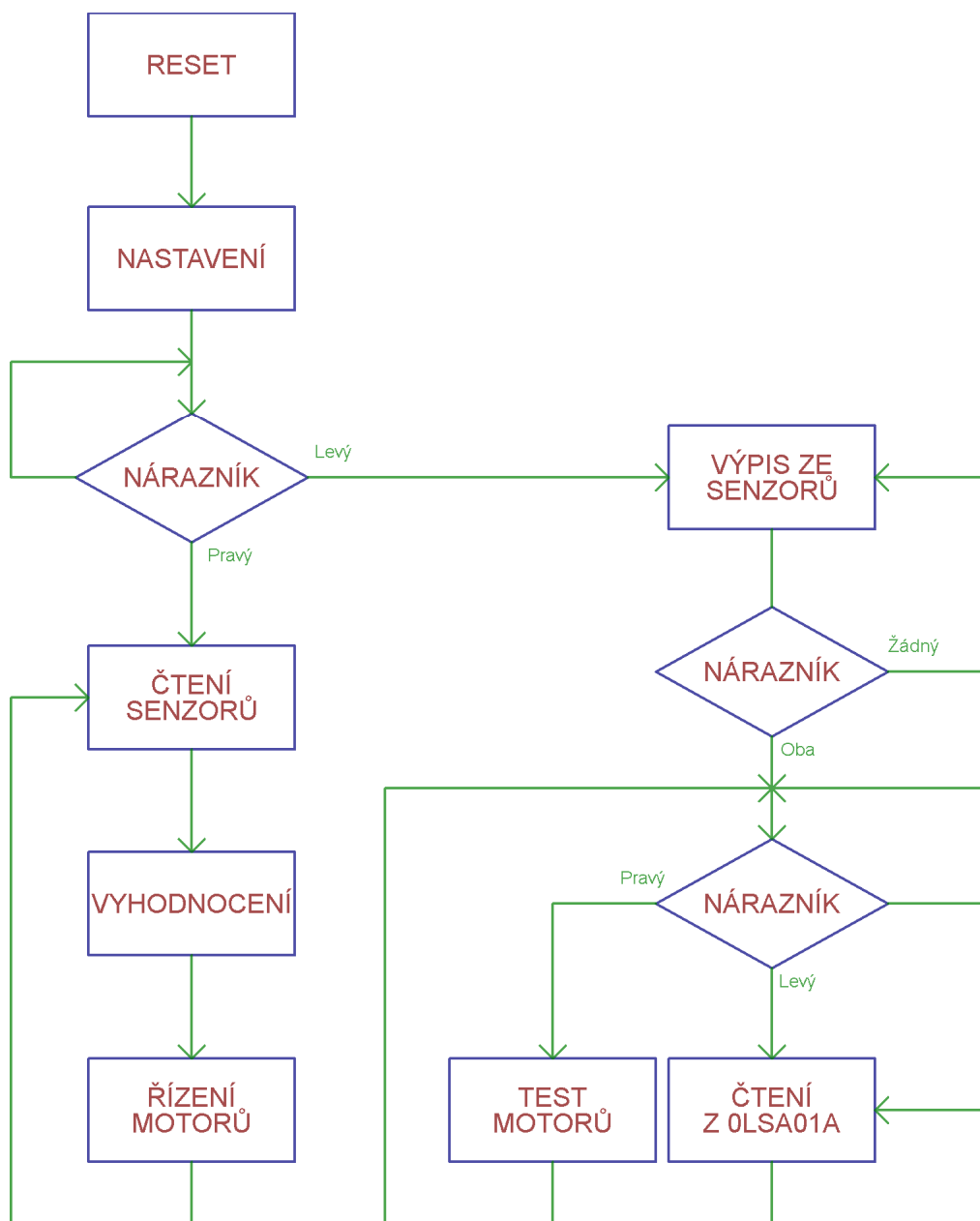
Všechny vývody kromě vývodů RB2, RD7 a RD6 jsou v programu nastaveny jako výstupní. Tyto piny slouží jako digitální vstupy. Piny RC0 a RC1 slouží pro přenos signálu PWM pro řízení rychlosti motorů.

## 4. Řídící program

Program byl napsán v jazyce C ve vývojovém prostředí PIC C Compiler od firmy PCW. Zkompilovaný program jsem do robota nahrál pomocí programátoru PICPROGUSB02A, kompatibilním s programátorem PICkit2 firmy Microchip.

### 4.1. PROCESNÍ DIAGRAM

Níže vidíte procesní diagram programu. Nezobrazuje chod jednotlivých kroků programu, ale pohlíží na program jako takový a ukazuje jeho základní strukturu. Podrobnější popis obsluhy senzorů a jejich vyhodnocení a řízení motorů bude v dalších kapitolách.



Jak je vidět z vývojového diagramu výše, po resetování procesoru dojde ke nastavení registrů procesoru, ale i jeho periferií. Poté se čeká na výběr režimu.

Prvním režimem je diagnostika, která se vyvolá stisknutím levého nárazníčku. V tomto režimu procesor čte údaje ze senzorů, vyhodnocuje pozici čáry a hodnoty posílá na sériovou linku, ale zároveň kontroluje i stav nárazníku. Po stisknutí obou tlačítek nárazníku si lze vybrat mezi testem motorů nebo úplným výpisem ze senzoru OLSA01A. Z praktických důvodů není možné diagnostiku opustit, a tak se změna režimu provádí resetováním procesoru.

Test motorů zajistí, že každý motor sám začne postupně zrychlovat na maximální hodnotu a poté postupně zpomalí. Nejdříve se motor točí dopředu, poté dozadu. Jako první se testuje levý motor a jako druhý motor pravý. Po ukončení lze spustit opět test motorů nebo čtení ze senzoru OLSA01A.

Čtení ze senzoru OLSA01A zajistí, že se do paměti procesoru uloží naměřené hodnoty všech 102 pixelů. Ty se následně vypíšou v hexadecimálním formátu na sériovou linku. V praxi se ukázalo lepší, když procesor vypisuje hodnoty kontinuálně a co nejrychleji. Proto v tomto režimu není možné provést jinou akci a pro opětovné spuštění diagnostiky je potřeba procesor resetovat.

V režimu sledování čáry procesor kontroluje všechny senzory, provádí jejich vyhodnocení a podle získaných údajů nastavuje motory tak, aby se černá čára držela uprostřed čtené řádky, tedy, čím více se čára odklání od středu, tím více robot zatáčí. Aby při sledování čáry nedošlo ke spuštění diagnostiky, je potřeba pro změnu režimu resetovat procesor.

## 4.2. POPIS PROGRAMU

Nyní rozeberu nejdůležitější části programu, které se týkají čtení senzorů a jejich vyhodnocení, indikátorů a řízení motorů. Celý program je uložen na přiloženém CD.

### 4.2.1. INDIKACE

Na robotovi je umístěno několik indikačních prvků. Ty slouží k jednoduchému zjištění zvoleného režimu či upozornění. Jedná se o dvě červené SMD LED diody a piezo sirénku.

#### LED diody

Led diody se obsluhují stejným způsobem jako digitální výstupy, jen je potřeba mít na paměti, že pokud nastavím výstup do stavu logické 0, bude dioda svítit a pokud do logické 1, dioda bude zhasnutá, protože je připojena anodou na zdroj napětí a katodou na výstup procesoru.

Obsluha vývodů poté probíhá následovně:

```
output_high(LED1);           // zhasne LED1
output_low(LED2);            // rozsvítí LED2
```

#### Piezo sirénka

Aby sirénka udělala delší zvuk - pípnutí, je potřebné, aby na ní bylo přivedeno proměnné napětí. K tomuto účelu jsem vytvořil funkci *beep*, kterou lze vidět na další straně.

```

void beep(int16 period,int16 length)
{
    int16 bp; // promenna pro nastaveni delky
    for(bp=length;bp>0;bp--) // prepina vystupy tolikrat, jakou jsme zadali delku
    {
        output_high(SOUND_HI);
        output_low(SOUND_LO);
        delay_us(period);
        output_high(SOUND_LO);
        output_low(SOUND_HI);
        delay_us(period);
    }
}

```

Jako parametr funkce slouží dvě proměnné. Proměnná *period* určuje výslednou frekvenci tónu a proměnná *length* délku pípnutí (určuje, kolikrát proběhne cyklus). Cyklus *for* opakuje sled instrukcí, které ve výsledku mění polaritu napětí přivedeného na sirénku.

## 4.2.2. ŘÍZENÍ MOTORŮ

Obsluha motorů vyžaduje nastavit směr otáčení a rychlost. K tomu slouží příkazy s nastavením logické úrovně na výstupu (*output\_low(PIN)* / *output\_high(PIN)*). Rychlost se nastavuje pomocí PWM, k čemuž je nutné nastavit časovač číslo 2, PWM povolit a a k nastavení hodnoty použít příkaz *set\_pwm2\_duty(RYCHLOST)*.

```

void l_motor_fwd(int8 speedl) // Levy motor dopredu
{
    output_high(LMF);
    output_low(LMB);
    set_pwm2_duty(speedl);
}

void l_motor_bwd(int8 speedl) // Levy motor dozadu
{
    output_high(LMB);
    output_low(LMF);
    set_pwm2_duty(speedl);
}

void r_motor_fwd(int8 speedr) // pravy motor dopredu
{
    output_high(RMF);
    output_low(RMB);
    set_pwm1_duty(speedr);
}

void r_motor_bwd(int8 speedr) // pravy motor dozadu
{
    output_high(RMB);
    output_low(RMF);
    set_pwm1_duty(speedr);
}

void l_motor_off() // Levy motor vypnut
{
    output_low(LMF);
    output_low(LMB);
    set_pwm2_duty(0);
}

void r_motor_off() // pravy motor vypnut
{
    output_low(RMF);
    output_low(RMB);
    set_pwm1_duty(0);
}

```

### 4.2.3. ČTENÍ ANALOGOVÝCH SENZORŮ

Pro používání analogových vstupů je potřeba je nastavit. Musí se definovat, které vývody budou sloužit jako analogové vstupy a povolit interní oscilátor pro interní AD převodník. Tato deklarace probíhá v nastavení.

```
setup_adc_ports(sAN0-sAN1-sAN2); // aktivní analogové vstupy RA0, RA1 a RA2
setup_adc(ADC_CLOCK_INTERNAL); // interní hodiny pro AD převodník
```

Nyní lze začít číst ze senzorů hodnoty. To jsem udělal tak, že jsem vytvořil funkci, do níž jsem vložil příkazy, které uloží výstup z interního AD převodníku do globální proměnné *line\_l* nebo *line\_r*. Konstanty *LINEL* a *LINER* označují číslo analogového vstupu. Pauzy 10 us zajišťují, že proběhne celý převod.

```
void read_blue_sensors() // ctení nouzovych senzoru
{
    set_adc_channel(LINEL); // cti levý nouzový senzor
    delay_us(10);
    line_l=read_adc();
    set_adc_channel(LINER); // cti pravý nouzový senzor
    delay_us(10);
    line_r=read_adc();
}
```

Pokud robot stojí na černé, je výstupní hodnota blízka 0, pokud na bílé, je v proměnné uloženo číslo blízke 255. Následné vyhodnocení probíhá velmi jednoduše pomocí podmínky *if*, která porovnává konstantu *TRESHOLD* (rozhodující hodnota) s hodnotou, která se uložila do proměnných.

Pokud přejede robot černou čáru, dojde k prudkému zatočení, aby se čára dostala zpět mezi záchranné senzory (a tedy i do zorného úhlu řádkového snímače) a zapamatuje si pozici, kde byla čára viděna naposledy. Řídící program rozeberu později.

### 4.2.4. PRÁCE S OPTICKÝM ŘÁDKOVÝM SNÍMAČEM

Pro komunikaci se řádkovým senzorem se používá sériová linka. Tu má sice procesor integrovanou hardwarově, ale je zde sloučen sériový vstup a hodinový signál, tedy, procesor neumí pracovat se sériovou synchronní linkou, kterou optický řádkový senzor vyžaduje.

Proto jsem musel synchronní sériový přenos naprogramovat. Slouží k němu tři vývody, v programu pojmenované jako *SCLK*, *SDIN* a *SDOUT*. Toto označení je shodné s označením vývodu modulu *OLSA01A*, a tak *SDIN* slouží jako výstup (z pohledu procesoru) a *SDOUT* jako vstup dat (z pohledu procesoru). *SCLK* je hodinový signál, který generuje procesor na základě programu.

Informace, které procesor posílá, jsou uloženy do polí. V každém poli je uloženo osm bitů řídicí instrukce. Pro odesílání dat po sériové lince bylo vytvořeno několik funkcí. Jedná se o funkce *olsa\_pulse()*, *olsa\_pulses(POČET)* a *olsa\_send(INFORMACE)*. Pro čtení ze senzoru jsem napsal funkci *read\_olsa()*.

#### OLSA\_PULSE()

Jedná se o velmi jednoduchou funkci. Jejím cílem je pouze zpřehlednit program komunikace. Vytváří jeden impulz na hodinovém výstupu *SCLK*.

```
void olsa_pulse() // vytvori jeden impulz
{
    output_high(SCLK);
    output_low(SCLK);
}
```

## OLSA\_PULSES(POČET)

Tato funkce je velmi podobná funkci `OLSA_PULSE()`. Jejím cílem je vytvořit na výstupu hodinového signálu `SCLK` počet impulzů, který je uveden v závorce. Byla vytvořena proto, že některé příkazy po zaslání potřebují více impulzů, než se provedou. Jedná se např. o příkazy `reset`, `start_integration` a `stop_integration`.

Pro provedení správného počtu impulzů slouží v této funkci cyklus `for`.

```
void olsa_pulses(int count)           // vytvori impulzy pro ridici logiku
{
    int8 ct;
    for(ct=0;ct<=count;ct++)
    {
        output_high(SCLK);
        output_low(SCLK);
    }
}
```

## OLSA\_SEND(PŘÍKAZ)

Funkce `olsa_send(příkaz)` je jedna ze složitějších funkcí pro obsluhu senzoru. Zajišťuje odeslání zprávy do senzoru. Zpráva je uložena v osmibitovém poli, které je celé prohlédnuto a podle hodnoty, které jsou uloženy v buňkách (0 a 1) je nastaven výstup procesoru. Jako první se odesílá nejméně významný bit.

Zprávy a obsah, které lze do senzoru posílat určuje výrobce v dokumentaci k integrovanému obvodu TSL3301. Podrobněji jsem se o práci s tímto obvodem rozepsal v jeho dokumentaci.

Krom prohlédnutí pole zajišťuje i odeslání start bitu a stop bitu a synchronizaci (po každém odeslání osmibitové zprávy je důležité nastavit výstup `SDIN` na logickou jedničku a hodinový výstup `SCLK` na logickou nulu).

```
void olsa_send(int8 info[8])          // USART komunikace s modulem OLSA01A - poslani zpravy
{
    int *ip;                           // ukazatel na pole s informaci
    int1 i;                             // pomocna promenna pro nastaveni 0 nebo 1 na SDIN
    output_low(SDIN);                  // start bit
    olsa_pulse();
    for(ip=0;ip<8;ip++)                // predani informace - 8 bit, LSB prvni > MSB posledni
    {
        i=info[ip];                    // ziskani hodnoty z pole
        if(i==1)                        // vyhodnoceni obsahu informace - nastav 1
        {
            output_high(SDIN);
        }
        else                             // vyhodnoceni obsahu informace - nastav 0
        {
            output_low(SDIN);
        }
        olsa_pulse();
    }
    output_high(SDIN);                  // stop bit
    olsa_pulse();
}
```



## OLSA\_RESET()

Tato funkce se provede pouze jednou v nastavovací části programu. Jejím úkolem je vyresetovat senzor a připravit jej k použití poté, co bylo připojeno napájení, protože mohlo dojít k hazardním stavům v senzoru a čtení z něj by nemuselo korektně fungovat.

Vyresetování se provede tak, že se vstup senzoru *SDIN* vynuluje a pošle se třicet hodinových impulsů na vstup *SCLK*. Poté je vhodné provést synchronizaci start bitu. Ta se uskuteční tak, že se vstup *SDIN* nastaví na logickou 1 a pošle se deset hodinových impulsů na hodinový vstup *SCLK*.

Dále je potřeba vynulovat vnitřní registry. To se provede odesláním patřičné zprávy do senzoru.

```
void olsa_reset()                // hlavní RESET - provádí se po zapnutí
{
    output_low(SDIN);
    output_low(SCLK);
    olsa_pulses(30);              // reset radkového senzoru
    output_high(SDIN);
    olsa_pulses(10);             // start bit - synchronizace
    olsa_send(main_reset);
    olsa_pulses(5);
    olsa_send(set_mode_rg);      // vycisti mode registr
    olsa_send(clear_mode_rg);
}
```

## OLSA\_SETUP()

Po provedení resetu je potřeba opět nastavit všechny registry a zisk senzoru, který určuje citlivost - délku expozice. Hodnoty, které lze do registrů zapsat opět uvádí výrobce v dokumentaci k senzoru. Nastavení se provede tak, že se do senzoru odešle sled patřičných zpráv, jak můžete vidět níže.

Tato funkce se v programu volá pouze v nastavovací části, neboť pro mé účely není potřeba při běhu programu měnit nastavení senzoru. Praktické využití by však bylo možné, pokud by robot měl ještě senzor na detekci světla okolního prostředí, kdy by mírou expozice mohl kompenzovat úbytek světla (např. v tunelu, který bývá častou překážkou v soutěžích nebo při poruše osvětlovacího modulu).

```
void olsa_setup()                // kompletní nastavení, provádí se po resetu
{
    olsa_send(left_offset);       // nastavení levého segmentu (offset a zisk)
    olsa_send(offset);
    olsa_send(left_gain);
    olsa_send(gain);
    olsa_send(mid_offset);       // nastavení prostředního segmentu (offset a zisk)
    olsa_send(offset);
    olsa_send(mid_gain);
    olsa_send(gain);
    olsa_send(right_offset);     // nastavení pravého segmentu (offset a zisk)
    olsa_send(offset);
    olsa_send(right_gain);
    olsa_send(gain);
}
```

## OLSA\_INTEGRATION( )

Funkce *olsa\_integration()* krom odeslání příkazu o začátek měření - expozice, zasílá i požadavek o jeho ukončení. Jak lze vidět v dokumentaci od výrobce, je potřeba po poslání příkazu *start\_integration* poslat 22 hodinových impulzů, kdy senzor měří a zapisuje hodnoty do registru. Po odeslání příkazu *stop\_integration* je potřeba poslat 5 hodinových impulzů, kdy senzor připraví data k odeslání.

```
void olsa_integration()           // snimani pixelu
{
    olsa_send(start_int);         // zacatek integrace senzoru
    olsa_pulses(22);
    olsa_send(stop_int);         // konec integrace senzoru
    olsa_pulses(5);
}
```

## READ\_OLSA( )

Na začátku funkce odešle příkaz o naměření *olsa\_integration()* a posléze příkaz na odeslání naměřených hodnot *olsa\_send(readout)*. Dále bude zajišťovat příjem dat a jejich uložení.

Cyklus *do* zajišťuje, že se přečtou a uloží všechny hodnoty pixelů. Aby nedošlo ke čtení v momentě, kdy ještě nejsou na vstupu *SDOUT* platná data, je zde podmínka čekání na start bit a pokud nepřichází, odešle se jeden impulz na hodinový výstup *SCLK*.

Dále následuje cyklus *for*, který zajistí přečtení všech osmi bitů. Jednotlivé bity se uloží do bytu tak, že se provede operace logického součtu *or*, podle přijatého bitu buď s hodnotou 1 nebo 0 a následně posunutí pixelu.

Po získání celého bytu se jeho hodnota zapíše do pole hodnot. V závislosti na pořadí pixelu (0 až 101) se zapíše buď do pole pro levou nebo pravou část řádky, kde v každém může být 52 hodnot.

Toto řešení bylo zvoleno proto, protože maximální adresovatelná velikost u procesoru PIC16F887 je 96 bytů, což je nedostatečné pro uložení celé řádky. Sice by byla možnost vypustit několik pixelů, např. tři pixely z každé strany, ale tím by jsme se v podstatě ochudili o šest senzorů, představíme-li si řádku jako 102 bodových senzorů. A právě krajní pixely jsou pro udržení na čáře a plynulou jízdu důležité.

Po této funkci může následovat vyhodnocení polohy čáry a nastavení - řízení chodu motorů. Program funkce *read\_olsa()* je na další straně.

```

void read_olsa()
{
    int8 cpixel;           // pocet prectenych pixelu
    int8 cbit;            // pocet prectenych bitu
    int8 pixel;          // hodnota precteneho pixelu
    cpixel=0;
    lp=0;
    rp=0;
    olsa_integration();
    olsa_send(readout);
    do                    // precte 102 pixelu
    {
        if(!SDOUT)      // zacatek prenosu - zachycen start bit
        {
            pixel=0;
            for(cbit=0;cbit<8;cbit++) // cte jednotlivé bity (8 bitu - 0 az 7)
            {
                olsa_pulse();        // impulz pro generovani dalsiho bitu
                if(SDOUT)            // zachycena 1
                {
                    pixel|=1;        // zapise do bitu pixelu 1 - OR
                }
                else                // zachycena 0
                {
                    pixel|=0;        // zapise do bitu pixelu 0 - OR
                }
                pixel<<=1;          // posune pixel
            }
            olsa_pulse();           // generuje stop bit
            if(cpixel<52)          // ulozeni do pole
            {
                olsa_lseg[lp]=pixel; // leva polovina radky - leve pole
                lp++;
            }
            else
            {
                olsa_rseg[rp]=pixel; // prava polovina cary - prave pole
                rp++;
            }
            cpixel++;
        }
        else
        {
            olsa_pulse();          // generuje start bit, nebyl-li poslan
        }
    }
    while(cpixel<102);           // precte 102 pixelu
}

```

## 4.2.5. VYHODNOCENÍ

Po přípravě všech potřebných funkcí a vyzkoušení funkčnosti senzorů, jsem začal psát program pro vyhodnocení polohy čáry. Jakožto primární senzor na snímání slouží právě OLSA01A, a tak dva bodové senzory používám pouze jako ujištění, že robot nepřejel čáru, nebo v případě, že se ztratila čára z optického řádkového senzoru.

## OLSA\_POSITION( )

Úkolem této funkce je uložit do globální proměnné *position* hodnotu, která odpovídá poloze čáry na senzoru. To je provedeno tak, že funkce prohlíží dvě pole, přičemž v jednom jsou uloženy hodnoty levé poloviny řádky a ve druhém poloviny pravé.

Vyhodnocení je celkem jednoduché. Pokud funkce narazí na jistý počet po sobě jdoucích pixelů, jejichž hodnota odpovídá černé, lze s jistotou tvrdit, že jsme našli čáru. Pokud se pozici čáry nepovede z hodnot vyhodnotit, např. pokud čára zmizela ze zorného úhlu, uloží se do globální proměnné *position* hodnota 0.

Podle potřeby lze měnit v části programu, kde jsou uloženy konstanty, rozhodovací úroveň odpovídající černé barvě (uložena pod názvem *OLSA\_LEV*) nebo počet po sobě jdoucích pixelů, které slouží k detekci čáry (konstanta *LINE\_PX*). V mém případě používám *OLSA\_LEV=0x60* a *LINE\_PX = 4*.

```
void olsa_position() // vyhodnoti pozici cary
{
    int8 searchp; // ukazatel na pole
    int8 search; // ulozeni prectene hodnoty
    int8 protect_count; // opravdu vidime caru
    position=0; // nuluje pozici, pokud cara neni, ulozena 0
    for(searchp=0;searchp<52;searchp++) // prohlizi levou cast cary
    {
        search=olsa_lseg[searchp]; // vybira pixel
        if(search==OLSA_LEV) // cerna nebo bila?
        {
            protect_count++; // pokud nasleduje cerna, pricte 1 k poctu cernych pixelu
        }
        else
        {
            protect_count=0; // pokud nasleduje bila, pocet cernych pixelu vynuluje
        }
        if(protect_count>LINE_PX) // vidim caru
        {
            position=searchp; // zapis presnou pozici
            line_sector=LEFT; // cara je v leve polovine
            searchp=55; // ukonci hledani
        }
    }
    for(searchp=0;searchp<52;searchp++) // prohlizi pravou cast cary
    {
        search=olsa_rseg[searchp]; // vybira pixel
        if(search==OLSA_LEV)
        {
            protect_count++; // pokud nasleduje cerna, pricte 1 k poctu cernych pixelu
        }
        else
        {
            protect_count=0; // pokud nasleduje bila, pocet cernych pixelu vynuluje
        }
        if(protect_count>LINE_PX) // vidim caru
        {
            position=(searchp+50); // zapis presnou pozici
            line_sector=RIGHT; // cara je v prave polovine
            searchp=55; // ukonci hledani
        }
    }
}
```

## 4.2.6. ŘÍDÍCÍ ČÁST PROGRAMU A DIAGNOSTIKA

Na rozdíl od jednotlivých funkcí jsem se rozhodl, že zde nebudu uvádět stejně podrobně diagnostiku ani řídicí program, neboť by zabral několik stran, čímž by se tato práce znepřehlednila. Celý program přikládám na CD nosiči, kde je podrobně okomentován, a tak umožní zorientování v jeho funkci.

Řídicí program i diagnostika přesně odpovídá vývojovému diagramu a stručnému popisu v kapitole "*Vývojový diagram*".

Na začátku programu se do paměti uloží předchozí pozice robota. V paměti je uloženo posledních pět poloh robota, které se použijí pro výpočet PID regulátoru. Poté následuje čtení všech senzorů a jejich vyhodnocení. Ze současné pozice čáry se spočítá chyba, která se použije pro výpočet v PID regulátoru.

PID regulátor má dominantní složku P. Zavedení složky I zajistilo plynulejší jízdu po rovině a zavedení složky D se projevilo rychlejším průjezdem zatáček a lepší odezvou na rychlé změny polohy čáry vůči robotovi. Regulátor ukládá do globální proměnné absolutní hodnotu, tedy nerozlišuje rozdíl mezi regulací, pokud je čára v levé nebo v pravé polovině řádkového senzoru. Tento způsob se ukázal jako výhodnější. Pro nastavení motorů je proto nutné reagovat i na pozici čáry. To se zajistí porovnáním polohy čáry s konstantou 50, která odpovídá středu.

Bodové senzory, které jsou nasvíceny modře, se aktivují až v případě, že se několikrát po sobě po vyhodnocení polohy z optického řádkového senzoru vrátila hodnota 0, jež oznamuje, že se čára ztratila ze zorného úhlu senzoru. V tomto případě, pokud robot najede na některý ze dvou bodových senzorů, bude pozice, která odpovídá nule nahrazena imaginární pozicí, tedy, pokud robot najede na čáru levým senzorem, bude pozice rovna jedné (maximální výchylka vlevo), a pokud najede na čáru senzorem pravým, bude pozice rovna hodnotě 99 (maximální výchylka vpravo).

Odezva PID regulátoru a část s nastavením motorů poté zajistí, že se jejich rychlost nastaví tak, aby se čára vrátila zpět do zorného úhlu senzoru.

## 5. KONSTRUKCE

### 5.1. POPIS MECHANICKÉ KONSTRUKCE

Pro robota jsem zvolil dvoukolový podvozek s třecím bodem, který je pro zmenšení třecího odporu realizován kuličkou umístěnou v pouzdře s ložisky. Obě kola jsou nezávisle poháněny vlastním elektrickým komutátorovým motorem. Toto řešení umožňuje zatáčet na velmi malém prostoru, a tak by robotovi nemělo činit problém ani projetí ostrých zatáček.

Nosná část robota je zkonstruována ze dvou základních desek MLAB ALBASE1115, což umožňuje osadit robota v podstatě jakýmkoliv MLAB kompatibilním zařízením. Zároveň děrování desek v rastru 400 mils odlehčuje celkovou konstrukci. Desky jsou k sobě spojeny závitovými tyčemi, čímž se vytvořili dvě patra. Do prvního jsem umístil motory, odrušovací cívky a modul LION2CELL01A. Ve druhém patře se nachází zbytek elektroniky.

Na konstrukci jsem použil také díly ze stavebnice Merkur. Ty slouží k uchycení řádkového senzoru a osvětlení, k upevnění motorů a také jsem z nich vytvořil nárazník.

Důležitou částí konstrukce, kterou bylo potřeba vyřešit byl právě úchop optického řádkového senzoru OLSA01A a modulu LEDSENBAR01A. Rozhodující byl úhel, pod kterým snímá OLSA01A čáru. Pokud se úhel snímání blížil 90°, začínalo být snímání nespolehlivé kvůli stínění. Naopak, pokud byl úhel příliš malý, uplatňovaly se při snímání odlesky povrchu, které opět snižovaly spolehlivost snímání.

Modul s LED diodami LEDSENBAR01A jsem umístil do takové vzdálenosti, aby se světlo z jednotlivých LED diod na snímaném povrchu slévalo, což také pomáhá zvýšit spolehlivost snímání.

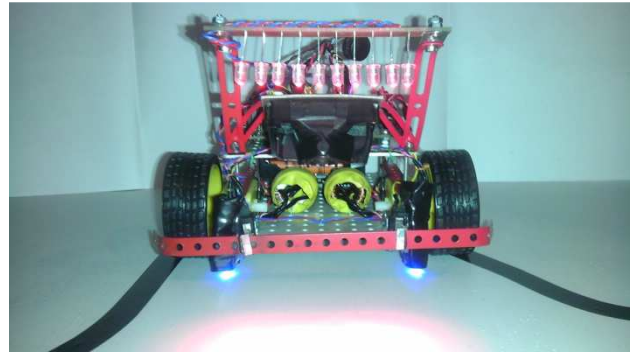
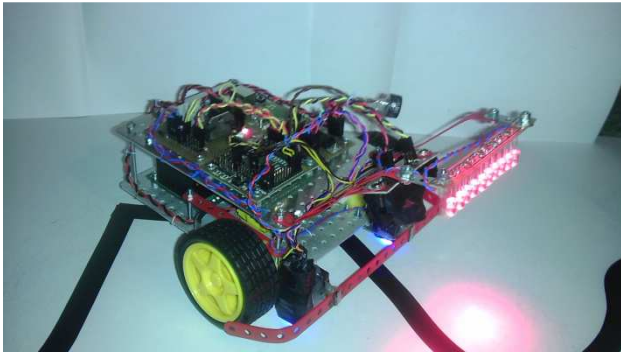
Robot je 20 cm široký a vysoký je 14 cm i s konstrukcí potřebnou pro řádkový senzor a modul s osvětlením. Váha i s bateriemi je 800g.

### 5.2. FOTOGRAFIE ROBOTA

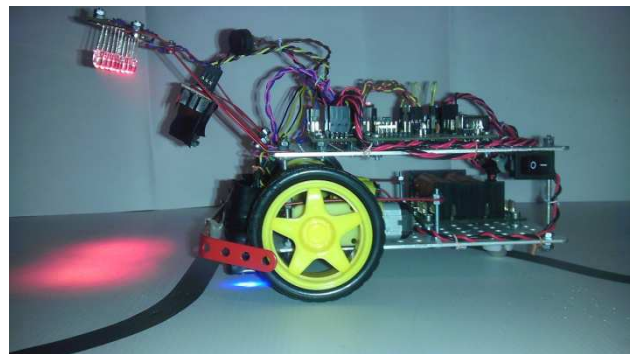
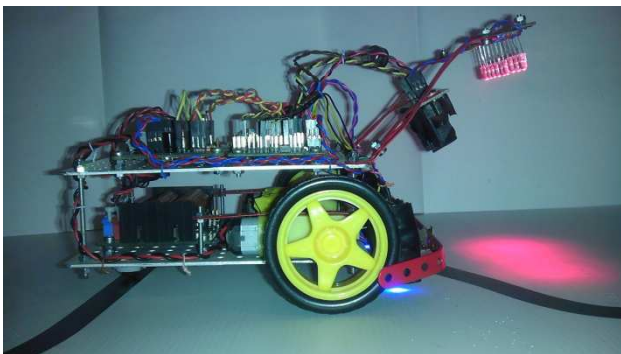
Po dobu vývoje jsem prováděl fotodokumentaci svých prací. Vybrané fotografie a videa najdete na příloženém CD nosiči. Na další straně si můžete prohlédnout na fotografiích finální podobu robota.

Jak lze vidět z fotografií, ve velkých mezích se měnila výška robota. To bylo způsobené jednak změnou velikosti bateriového modulu, ale také hledáním správného úhlu a výšky optického řádkového senzoru.

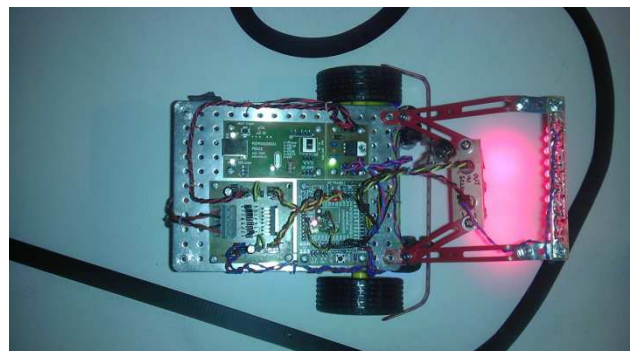
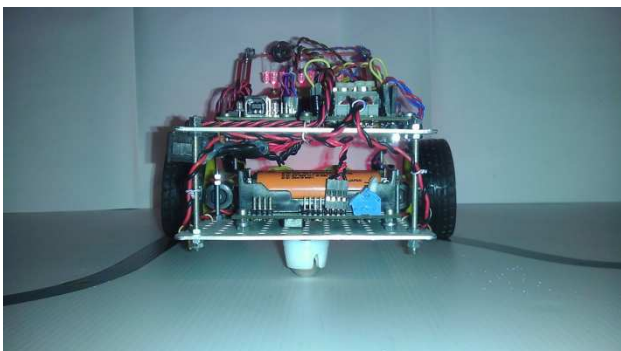
Na posledním páru fotografií na poslední straně jsou vidět snímky, kdy probíhalo testování optického řádkového snímače. Jak si lze všimnout, robot byl v té době napájen z provizorního zdroje elektrické energie - NiMh akumulátoru do airsoftových zbraní. Na průběhu z testování je patrné, že senzor odesílá data. Žlutá barva znázorňuje průběh dat, zelená hodinový signál, který jsem kvůli větší názornosti zobrazil s poloviční amplitudou.



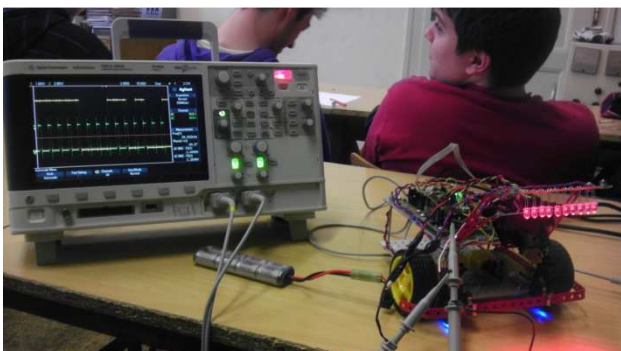
*Celkový pohled na robota (vlevo) a pohled zepředu (vpravo)*



*Pravá strana robota (vlevo) a levá strana robota (vpravo)*



*Pohled zezadu (vlevo) a pohled shora (vpravo)*



*Test funkčnosti optického řádkového senzoru (vlevo) a detail průběhu (vpravo)*

## 6. ZÁVĚR

Při řešení maturitní práce jsem prozkoumal problematiku týkající se senzorky, řízení a konstrukce robota sledujícího černou čáru na kontrastním - bílém podkladě. Zaobíral jsem se zejména nad problematikou užití optického řádkového senzoru, který se jeví jako sofistikovanější řešení pro snímání černé čáry, než jsou jednoduché bodové senzory.

Výsledkem této práce je prototyp robota sledujícího černou čáru. Tento robot všemi parametry odpovídá zadání a požadavkům na něj. Vyzdvihl bych zejména kompatibilitu se stavebnicí MLAB, která umožňuje použít robota v kombinaci s dalšími moduly, a tak jej rozšířit o mnoho dalších funkcí, nebo použít moduly navržené pro tohoto robota i k jiným účelům.

Podle dokumentace přiložené k této práci lze sestavit shodný typ robota a použít jej například k výuce na středních i vysokých školách nebo na mezinárodní soutěže, jako je např. Robotchallenge ve Vídni a po doplnění lepší detekce překážky, kterou lze nyní provádět pouze pomocí nárazníku, i na soutěž Istrobot v Bratislavě.

Přínosem této práce je vytvoření důležitých modulů do stavebnice MLAB, které mohou sloužit při sestrojování mobilních robotů (např. OLSA01A a HBRDGLL29801A) i dalších elektronických zařízení. (např. LIONCELL01A).



## 7. POUŽITÁ LITERATURA

Při realizaci své maturitní práce jsem čerpal z těchto knih a webových stránek:

1. PRŮMYSLOVÁ ELEKTRONIKA A INFORMAČNÍ TECHNOLOGIE  
*Heinz Haberle a kolektiv*
2. UČÍME SE PROGRAMOVAT V JAZYCE C  
*Václav Kadlec*
3. ELEKTRONIKA - SOUČÁSTKY A OBVODY, PRINCIPY A PŘÍKLADY  
*Jan Vobecký, Vít Záhlava*
4. STAVEBNICE MLAB  
*www.mlab.cz*
5. H-MŮSTKY A ELEKTRONICKÉ SPÍNAČE  
*www.robotrevue.cz*
6. NAPÁJECÍ ZDROJE, BATERIE A AKUMULÁTORY  
*www.wikipedie.org*

## 8. OBSAH PŘILOŽENÉHO CD

Součástí této práce je také přiložené CD, kde lze nalézt celou práci v elektronické podobě i s dokumentací ke všem použitým modulům. Dále se na CD nachází i program v jazyce C včetně všech souborů potřebných pro překlad ve vývojovém prostředí PCW PIC C Compiler, ale i přeložený soubor pro procesor PIC16F887.

Na CD jsou umístěny i videa a fotografie robota, jak z finální verze, tak z verze testovací.

### ADRESÁŘOVÁ STRUKTURA CD:

/DOC/MODULES	elektronická verze dokumentace všech použitých modulů
/DOC/MP	elektronická verze této práce
/PHOTOS	fotografie robota
/SW	soubory programu pro robota
/VIDEOS	videa robota

## **9. PŘÍLOHA A**

### **DOKUMENTACE NAVRŽENÝCH MODULŮ**