V y s o k é u č e n í t e c h n i c k é v B r n ě
F a k u l t a s t r o j n í h o i n ž e n ý r s t v í

DIPLOMOVÁ PRÁCE

Využití platformy Eyebot pro řízení mobilních robotů

vypracoval: Bc. Miroslav Světlík
vedoucí práce: Ing. Robert Grepl, Ph.D.
obor: Inženýrská mechanika
specializace: Mechatronika
2008

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2007/08

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Světlík Miroslav, Bc.

který/která studuje v **magisterském studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Využití platformy Eyebot pro řízení mobilních robotů

v anglickém jazyce:

Use of Eyebot platform for control of mobile robots

Stručná charakteristika problematiky úkolu:

Práce se zabývá využitím hardwarové platformy Eyebot pro řízení experimentálních mobilních zařízení. Student se seznámí s dodaným OS RoBIOS, který obsahuje knihovny pro obsluhu periferií a zpracování sensorických dat. Následně vytvoří řídicí programy pro konkrétní mobilní systémy a otestuje jejich vlastnosti.

Cíle diplomové práce:

1) Seznamte se s vlastnostmi a programováním platformy Eyebot.
2) Otestujte chování a vlastnosti systému při kombinaci několika úloh (vstupně/výstupní kanály).
3) Vypracujte řešení následujících konkrétních problémů:
- Řízení pohybu kráčejícího čtyřnohého robotu (12 servopohonů, snímání dat z akcelerometrů, sil v došlapových snímačích).
- Řízení dvoukolového robotu (snímání dvou enkodérů, řízení dvou DC motorů, snímání hodnot z akcelerometru, implementace dodaného hotového LQ řízení.
- Řízení čtyřkolového robotu (čtyři DC motory, čtyři enkodéry, osm servopohonů, dva trojosé akcelerometry, MEMS gyro).
4) Vypracujte komunikační software pro PC (komunikace s Eyebotem přes BlueTooth).
5) Vypracujte podrobnou dokumentaci k řešeným problémům dle pokynů vedoucího DP.

Abstrakt

Diplomová práce je zaměřená na využití platformy Eyebot pro řízení mobilních robotů. Jejím cílem je seznámit se s vlastnostmi a programováním platformy Eyebot a otestovat vlastnosti a chování systému. Dále se zabývá komunikací řídící jednotky Eyebot a PC za použití bezdrátové technologie Bluetooth. Následně pak využitím komunikační platformy k řízení kráčejícího čtyřnohého experimentálního robotu. Veškeré navržené algoritmy pro řídící jednotku Eyebot byly navrženy v jazyce C a algoritmy pro komunikaci a GUI řízení na PC byly realizovány v programovém prostředí Matlab 2008a.

Abstract

This diploma thesis focuses on the Eyebot mobile robot control platform utilization. The aim of this thesis is to become familiar with the characteristics and programming of the Eyebot platform, and to test the attributes and performance of its system. This piece deals more with the Eyebot control unit and PC communication via Bluetooth wireless technology, than with utilizing the experimental four-legged walking robot communication platform. All draft algorithms for the Eyebot control unit have been drafted in C language and the communication and GUI PC control algorithms have been implemented in the Matlab 2008A program environment.

Poděkování:

Děkuji vedoucímu diplomové práce ing. Robertu Greplovi, Ph.D. za metodické vedení a konzultace.

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně s použitím uvedené literatury.

V Brně dne 30.9.2008

Podpis:

Obsah:

1 Úvod

Od počátku lidské společnosti se lidé snaží ulehčit si svoji činnost tvorbou různých technických vynálezů, které zastupují lidské působení při daných operacích, zejména pak při rutinních a namáhavých pracích. Dnes už zasahuje i do takových oblastí průmyslu a jiných odvětví, ve kterých dříve bylo nepředstavitelné, jak lze lidskou činnost nahradit.

Názorným příkladem usnadnění práce může být právě vědní oblast zabývající se robotikou. K tomu, aby robot mohl nahrazovat lidské konání, se potřebuje umět pohybovat a orientovat v prostoru. Uvědomíme-li si, jakou prostupnost a přizpůsobivost má člověk v okolním prostředí, je nedostatečné spokojit se s konstrukcemi robotů opatřených kolovými nebo pásovými podvozky, které nemají zdaleka takovou adaptabilitu vůči prostředí, jakou mají kráčiví roboti. Je to však na úkor složitosti konstrukce, energetické náročnosti robotu, zkoumání stability a zajištění dynamičnosti chůze a především řízení.

Proto musíme zkoumat různé metody a způsoby řízení robotů. Často se k řízení používají různé mikrokontrolery s vhodným řídícím softwarem.

1.1 Cíle práce

Tato práce si klade za cíl:

- Seznámit se s platformou Eyebot a jejím využitím
- Vytvořit experimenty, kterými otestuji vlastnosti a chování platformy Eyebot
- Vytvořit komunikační prostředí mezi řídící jednotkou Eyebot a PC
- Navrhnout a realizovat řídící program pro Eyebot v jazyce C
- Navrhnout a realizovat řídící software pro PC na ovládání jednotky Eyebot, komunikace přes Bluetooth

2 Rešeršní studie

Veškeré podklady týkající se platformy Eyebot jsem čerpal z veřejných webových stránek Robotics & Automation Lab [1].

2.1 Popis Eyebotu

Eyebot je controller pro mobilní roboty kráčející, kolové, létací a další (např.:Obr. 1). Skládá se ze silné 32 bitové mikrokontrolérové desky s grafickým displayem a kamery. Tato kamera je přímo připojená k desce, to umožňuje rychlé zpracování obrazu.

Obr. 1 Mobilní roboti s použitou řídící jednotkou Eyebot

2.2 Hlavní rysy Eyebotu

Eyebot disponuje jednotlivými periferiemi jako (Obr. 2):

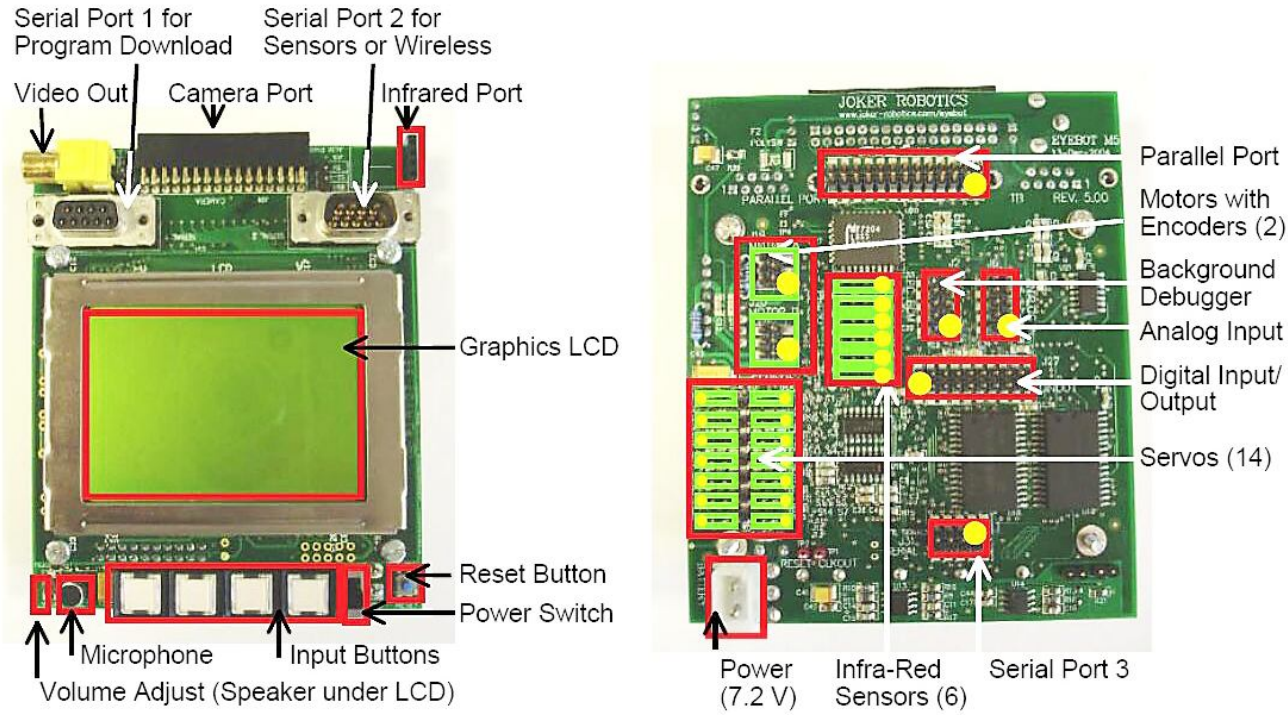
- 1 paralelní port
- 3 sériové porty (2 V24, 1 TTL)
- 8 digitálních vstupů
- 8 digitálních výstupů
- 8 analogových vstupů
- 2 řízení motorů
- kompaktní jednotka PCB

- rozhraní pro barevnou kameru
- velký grafický display LCD (128x64 pixelů)
- 4 tlačítka
- tlačítko reset, vypínač napájení
- reproduktor
- mikrofon
- indikátor stavu baterie

Připojení:

- pro akční člen: 2 DC motory a/nebo 14 servopohonů (PWM)
- pro senzory: digitální kamera, infračervené snímače pro měření vzdálenosti (PSD)
- pro vstup/výstup: 8 digitálních vstupů, 8 digitálních výstupů, 8 analogových vstupů

2.3 Způsoby připojení jednotlivých periferií k Eyebotu



Obr. 2 Popis konektorů Eyebotu (verze M5)

Popis jednotlivých konektorů:

Serial Port 1 (9 pin, SUB-D samice)

Standardní RS-232 sériový port (**12 V**) pro propojení s PC

Serial Port 2 (9 pin SUB-D samec)

Standardní RS-232 sériový port (**12 V**)

Usage: BlueTooth Wireless or GPS

Serial Port 3 (10 pin)

Sériový RS-232 na TTL level (**5 V**)

Infrared Port – infračervený konektor (3 pin)

Standardní dálkový ovládač od TV, se dá použít na ovládaní Eyebotu.

Poznámka: různé TV ovladače se liší. Software byl vyvinut pro Nokia VCN620.

Video Output – video výstup (2 pin, RCA)

Je to výstup z kamery.

Poznámka:

- výstup je pouze černobílý
- jestli je kamera přizpůsobena k pomalejší obnovovací frekvenci, výstupní obrazový signál nemůže být použit

Digital Camera – digitální kamera (32 pin)

Konektor pro připojení digitální kamery k Eyebotu

Speaker (2 pin plus 2 pin)

Piezo reproduktor s impedancí 8 Ohm

Parallel Port (26 pin)

Standardní paralelní port

Background Debugger Module (10 pin)

Konektor pro připojení PC přes paralelní port, slouží ke čtení hodnot z procesoru.

DC Motor a Encoder (2 times 6 pin)

1. Motor -
2. Motor +
3. GND
4. Vcc (5V)
5. Enkoder kanál B
6. Enkoder kanál A

Servo (14 times 3 pin)

Konektor pro připojení až 14 servopohonů. Signály pro servopohony jsou mapované na TPU kanály 0..13.

Poznámka: konektory jsou ve 2 řadách po 7, jsou uspořádány z leva do prava: Signal_ServoA, Vcc, GND, 5V (reg.), GND, Vcc, Signal_ServoB.

Position Sensitive Device (6 times 4 pin)

Hodinový vstup je mapován na digitalní výstup 0 pro všechny PSDs.

PSD výstupy jsou mapovány: Digital Output 0..5

Analog Input (10 pin)

Mikrofon je mapován na: Analog Input 0

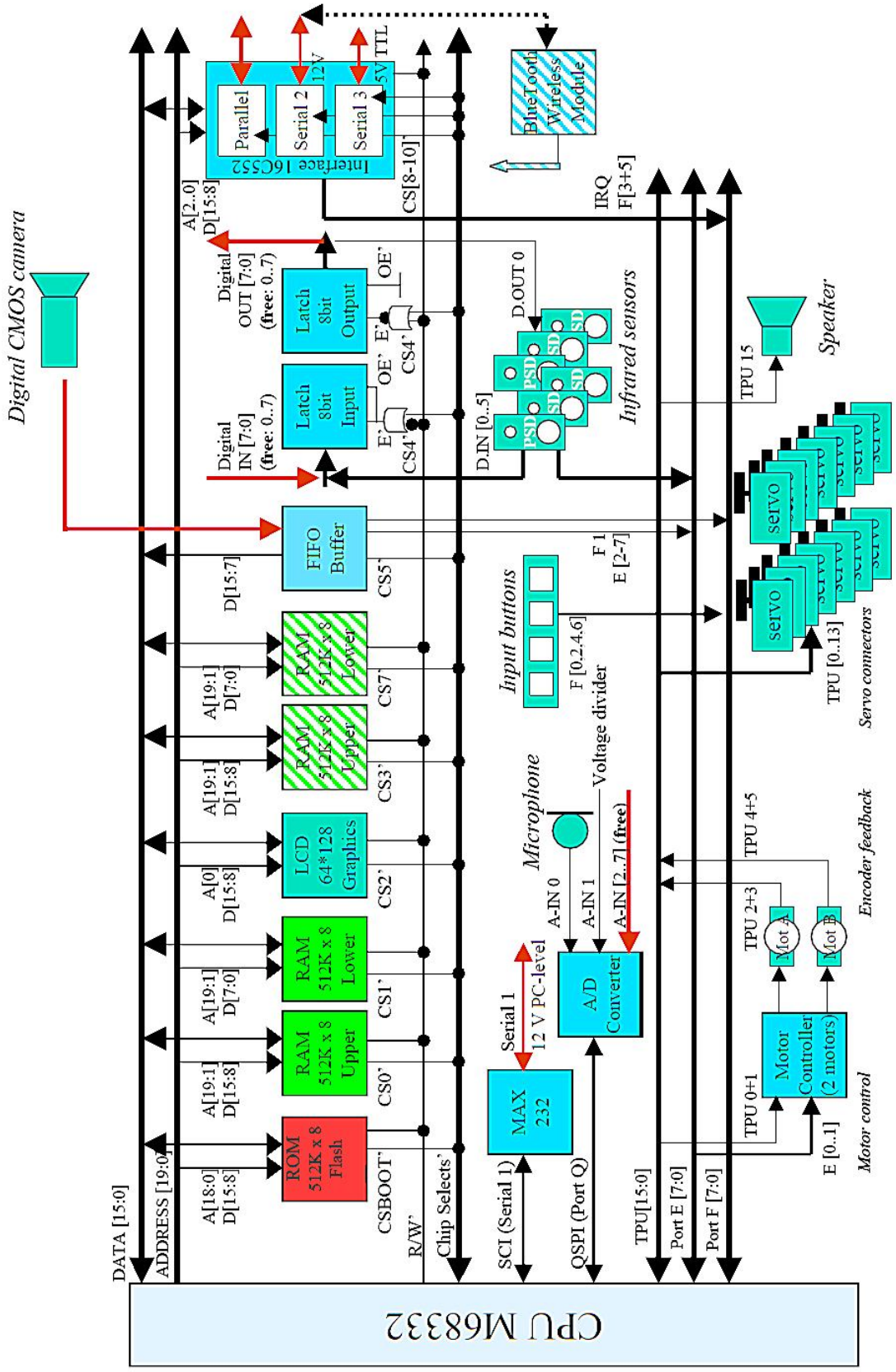
Stav baterie je mapován na: Analog Input 1

1 Vcc (5V regulated)
2 Vcc (5V regulated)
3 Analog Input 2
4 Analog Input 3
5 Analog Input 4
6 Analog Input 5
7 Analog Input 6
8 Analog Input 7
9 Analog GND
10 Analog GND

Digital Input/Output (20 pin)

Infračervené snímače PSD jsou mapovány na Digital Output 0 a Digital Input 0..5

```
 1- 8  Digital Output 0..7
 9-16  Digital Input  0..7
17+18  Vcc (5V)
19+20  GND
```



Obr. 3 Blokové schéma zapojení periferií Eyebotu

3 Kompilační prostředí

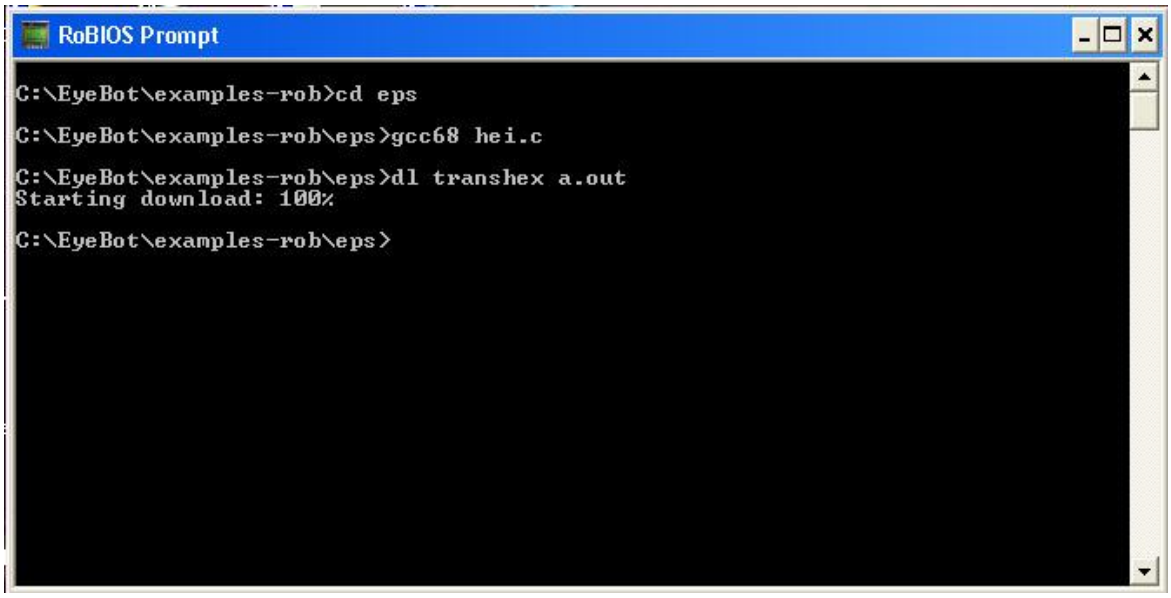
Kompilovat vlastní programy je možno v systému Windows i Linux. Programy **files.c** vytvořené v jazyce **C** zkompilujeme pomocí kompilátoru **gcc68** a programy **files.s** vytvořené v jazyce **ASM** se kompilují pomocí **gas68,** programy **files.s** vytvořené v jazyce **C++** se kompilují pomocí **g++68**. Je možno použít Online Compilator na URL: http://roboti.cs.siue.edu/eyebot/68compiler/. Zkompilované soubory jsou ve formátu **files.hex**. Tyto soubory již nahráváme přes sériový port COM1, 2.. do Eyebotu.

3.1 Kompilátor GCC68 ve Windows

Nejdříve začneme stažením kompilátoru z webu URL: http://robotics.ee.uwa.edu.au/eyebot/ftp/, kde stáhneme aktuální kompilátor (ke dni 2007-09-03 je to verze **rob65win.exe**). Tento kompilátor nainstalujeme na **C:\EyeBot**. Po dokončení instalace se nám na pracovní ploše vytvoří ikona **RoBIOS Prompt**. Po spuštění kompilátoru se otevře okno RoBIOS Prompt s příkazovou řádkou s otevřeným adresářem **C:\Eyebot\examples>**. Nejjednodušší je mít svůj vytvořený myfiles.c v tomto adresáři. Kompilaci spustíme napsáním příkazu:

```
gcc68 myfiles.c –o myfiles.hex
```

Parametr **-o** uloží výstup do souboru s názvem **myfiles.hex**. **Myfile.c.** Je to název souboru, který chceme zkompilovat, a myfiles.hex je název souboru, který chceme vygenerovat kompilátorem. Po potvrzení příkazu stisknutím Enter se nám vytvoří soubor v hexagonálním formátu, který můžeme nahrát do Eyebotu.



Obr. 4 Příklad příkazu v RoBIOS

3.2 Chybová hlášení kompilátoru

Dojde-li k syntaktické chybě (špatné napsání syntaxe v jazyce C), kompilátor na to zareaguje chybovým hlášením a nevytvořením výstupního souboru. Příklad chybového hlášení Obr. 5, chyba na 4 řádku.

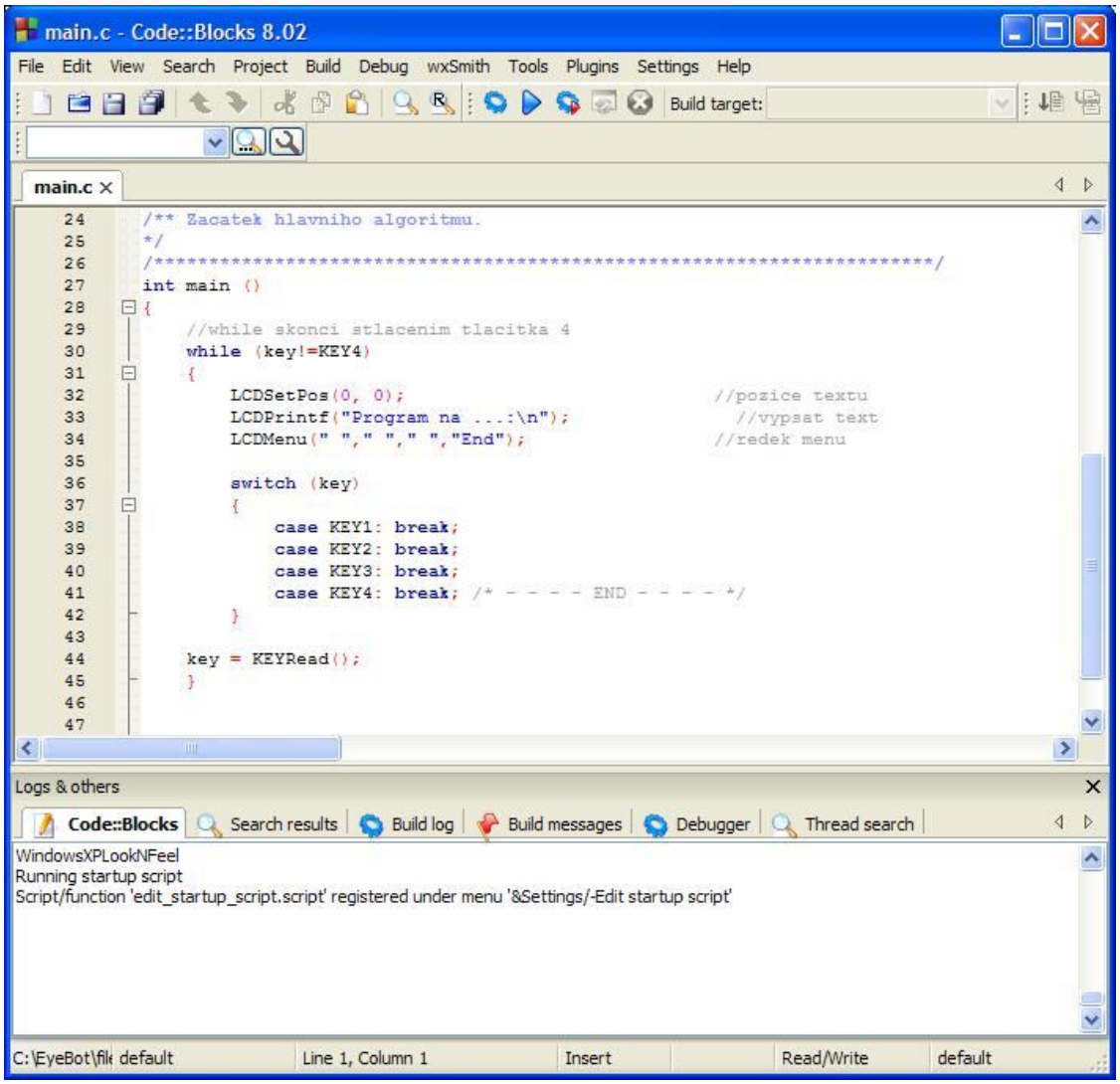
Obr. 5 Příklad chybového hlášení při špatné syntaxi

4 Vytváření a editace myfiles.c

Vytváření a editace programů je možné v jakémkoliv editoru, který zobrazí syntaktické chyby - doporučoval bych jednoduchý a snadno použitelný **PSPad** (ke stažení na url: http://www.pspad.com/cz/download.php) nebo **Code::Blocks** (http://www.codeblocks.org) na Obr. 6.

Code::Blocks lze použít v operačních systémech:

- Windows 2000 / XP / Vista
- Linux (Ubuntu & Debian, 32 & 64 bits)
- Mac OS X 10.4+



Obr. 6 Editační prostředí programu Code::Blocks

5 Nahrávání programu do Eyebotu

5.1 Připojení Eyebotu

Eyebot připojíme k PC pomocí kabelu pro sériovou konfiguraci zařízení pro RS-232. Eyebot napájíme 7.2V.

5.2 Ovládání Eyebotu

Po zapnutí Eyebotu nám naběhne RoBIOS. Pro přístup k programům, stiskneme tlačítko **Use** (Obr. 7).

Obr. 7 Startovní stav Eyebotu

Nyní jsme v uživatelské sekci Obr. 8

Obr. 8 Uživatelská sekce

Kde jsou tlačítka:

- Ld - nahrávání nového programu
- Run - spuštění nahraného aktuálního programu
- Rom - obsluha programů v paměti

Obr. 9 Nahrávací sekce

Po stlačení tlačítka **Ld** se dostaneme do nahrávací sekce (Obr. 9). Zde Eyebot vyčkává na to, než začne počítat a posílat data do Eyebotu. Jak odeslat program do EyeBotu je popsáno v kapitole 3.3 Příkazy na přenos dat. Na LCD displayi se zobrazuje stav přenosu dat. Takto nahraný program je aktivní a lze jej ihned spustit tlačítkem **Run**. Tento program je uložen jen v RAM a po vypnutí Eyebotu by byl program ztracen.

Obr. 10 Sekce obsluhy nahraných programů

Abychom mohli program používat i po vypnutí a opětovném zapnutí, je potřeba jej uložit do ROM. Po stlačení tlačítka **Rom** se dostaneme do sekce obsluhy nahraných programů (Obr. 10). Zde můžeme uložit maximálně 3 programy. Zároveň se nám zde zobrazuje aktivní program nahraný v RAM.

Zde jsou tlačítka:

- Save - uložení nahraného programu
- Ld - nahrání programu do RAM, tím se program stane aktivním a je ho možno spustit
- Next - další program (* je ukazatel)
- END – návrat zpět

5.3 Příkazy na přenos programu přes sériové rozhraní

V programu RoBIOS Prompt pro zkompilování souboru, napíšeme příkaz (Obr. 4):

```
dl myfiles.hex
```

Na Obr. 4 je příklad zkompilovaní programu hei.c a odeslaní do Eyebotu příkazem

```
dl transhex a.out
```

Zde musel být použitý parametr **transhex**, protože zkompilovaný soubor je v a.out.

5.3.1 Přenos dat

Během přenosu zaváděcí program poskytuje jméno programu, který nahráváme, a zprávu s počtem bajtů, které již byly přeneseny. Po úspěšném nahrání programu je na LCD zobrazena zpráva o úspěšném nahrání programu.

5.3.2 Zpráva o chybě

Zaváděcí program poskytuje návratový kód sytému Eyebotu v případě, že se setká s invalidními událostmi přenosu dat. Uživatelský program nemůže pracovat, jestliže nastane chyba během jeho přenosu. Krátká zpráva o chybě bude zobrazená na LCD.

- Seznam chybových hlášení:
  http://robotics.ee.uwa.edu.au/eyebot/doc/dl/exceptions.html

6 Ovládání periferií Eyebotu

Ovládání různých periferií je zde usnadněno používáním jednotlivých knihoven, které byly pro RoBIOS vytvořeny. Tyto knihovny lze najít na URL: http://robotics.ee.uwa.edu.au/eyebot/doc/API/library.html

Knihovny se zavádějí na začátku programu příkazem:

```
#include "eyebot.h"
```

6.1 Ovládání serva

Vytvořil jsem příklad programu, který promocí dvou tlačítek ovládá jeden servopohon v rozsahu 0 - 90˚. Hodnoty aktuálního natočení se zobrazují na displeji v desítkové soustavě v rozsahu 0 – 255. Jedním tlačítkem se zvyšuje hodnota a druhým se snižuje hodnota aktuálního natočení serva. Servo je zapojeno jak SERVO1 (vedle odporu R20).

Obr. 11 Eyebot s připojeným servem

Serva si zinicializujeme příkazy:

```
ServoHandle ser00;  // ser00 je nase pojmenovani ser-
va
#define SERVO_00 SERVO0
```

Poté příkazem:

```
ser00 = SERVOInit(SERVO_00);
```

kde můžeme zkontrolovat správnou inicializaci.

Servo se ovládá příkazem:

```
SERVOSet(ser00, servo_value);
```

Kde **servo_value** je název naší proměnné, která nabývá hodnot 0 až 255 podle úhlu natočení serva.

6.2 Sériová komunikace RS-232

Program na přijímání a odesílaní znaků z Eyebotu do počítače a obráceně za pomocí programu Docklight (V1.7). Nejdříve provedeme inicializaci sériového přenosu příkazem:

```
OSInitRS232(SER9600, NONE, SERIAL1);
```

První parametr je nastavení přenosové rychlosti od SER1200 do SER115200 (defaultní hodnota 9600). Druhý parametr je handshake. Třetí je rozhraní, přes které se bude komunikovat:

SERIAL1 – 12V PC-level - připojení sériového kabelu

SERIAL2 – 12V konektor pro připojení BlueTooth, Wireless nebo GPS

SERIAL3 – 5V TTL (vývod na spodní straně)

Příkaz na odeslání znaku z Eyebotu do PC přes RS232, v proměnné **sendchar** je znak, který se odešle:

```
OSSendRS232(&sendchar, SERIAL1);
```

Příkaz na načtení přijatého znaku z PC do Eyebotu. Přijatý znak je uložen do proměnné **ch**:

```
OSRecvRS232(&ch, SERIAL1);
```

Přijaté znaky čteme jednotlivě z implementovaného FIFO (first in first out) zásobníku.

Dále je možno odesílat celé řetězce znaků, viz příklad: **uplod_eye2.c**

(c:\EyeBot\examples\EyeCon\serial\ )

6.3 Přenos dat přes BlueTooth (BT)

Pro bezdrátovou komunikaci s Eyebotem je možné použít modul HandyWave **HPS-120.** Modul má 2 LED diody:

OPR (červená): když svítí nebo bliká, modul je zapnutý.

LNK (zelená): když svítí nebo bliká, modul je zapnutý v konfiguračním módu, kde lze nakonfigurovat pomocí programu (který tedy předem musíme nahrát) **bt-config.hex** (zdroj bt-config.c).

Přepínání mezi módy provádíme tlačítkem RST, je potřeba je držet alespoň 5 sekund. Dále toto tlačítko slouží jako reset nebo na odpojení a znovu připojení.

Pro zpárování HPS-120 modulu a BlueTooth v počítači je dobré použít program **BlueSoleil**. V tomto programu se nám modul zobrazí jako zařízení 1977. Poté je možno připojit port pro seriovou komunikaci. Po připojení se vytvoří sériový port (SPP) např. COMx.

V programu použijeme inicializaci pro seriový port SERIAL2, který je určený pro BlueTooth modul následujícím způsobem:

```
OSInitRS232(SER9600, NONE, SERIAL2);
```

Viz. kapitola 6.2. Dopuručuji používat defaultní rychlost SER9600.

Pozor: bluetooth modul nepracuje s nejvišší rychlostí SER115200.

6.4 Připojení motoru s encoderem

K Eyebotu můžeme připojit dva DC motory s enkodery.

Piny pomocí nichž mohu připojit DC Motor a Enkoder (2 times 6 pin):

```
Motor A: Speed = TPU0, Direction = Port_E0, Encoders =
TPU2+3
Motor B: Speed = TPU1, Direction = Port_E1, Encoders =
TPU4+5
      1 Motor -
      2 Motor +
      3 GND
      4 Vcc (5V regulated)
      5 Encoder Channel B
      6 Encoder Channel A
```

6.4.1 Příklad jednoduchého ovládání:

Motory si zinicializuji příkazy:

```
MotorHandle     leftmotor;
MotorHandle     rightmotor;
leftmotor = MOTORInit(LEFTMOTOR);
```

```
rightmotor = MOTORInit(RIGHTMOTOR);
```

Motor pak ovládám příkazy:

```
MOTORDrive (rightmotor,x);
MOTORDrive (leftmotor,x);
```

kde proměnná „x“ může nabývat hodnot od -100 do 100.

Po skončení ovládání motorů se ukončí relace příkazem:

```
MOTORRelease (rightmotor|leftmotor);
```

Používání enkoderu se nainicializuje funkcí:

```
QuadHandle QUADInit(DeviceSemantics semantics);
```

Čtení hodnot z enkoderu:

```
QUADRead(QuadHandle handle);
```

6.5 Obsluha časovačů

Pro nastavení systémového času máme funkci:

```
OSSetTime(int hrs,int mins,int secs);
```

Kde je vstupem:

- hrs - počet hodin
- mins - počet minut
- secs - počet vteřin

Pro zobrazení času máme funkci:

```
OSShowTime(void);
```

Pro zpoždění můžeme použít funkci, kde délka zpoždění je n*1/100 sekund:

```
OSWait (int n);
```

Při používání časovače s přerušením si ho nejdříve zinicializujeme funkcí:

```
TimerHandle OSAttachTimer(int scale, TimerFnc function);
```

Vstupem je:

- scale – dělící hodnoty pro 100Hz časovač
- TimerFnc – funkce, která bude periodicky volaná

Výstup je:

- TimerHandle – hlášení o chybě

Četnost volání funkce je 100/scale Hz.

Zrušení volání funkce při přerušení provedme funkcí:

```
OSDetachTimer(TimerHandle handle)
```

6.6 Obsluha analogových vstupů

Eyebot je vybaven 10 bitovým AD převodníkem, takže výstupní hodnota bude 0-1023, což je hodnota od 0V do 5V. Tento AD převodník má 8 kanálů. Z toho první je jako vstup mikrofonu a druhý čte hodnotu stavu baterie.

Funkce pro čtení 10bitové hodnoty z vybraného kanálu:

```
OSGetAD(int channel);
```

Vstupem je číslo kanálu 0-15 a výstupem je přečtená hodnota z toho kanálu.

Další funkce je na ovládání napajení AD převodníku:

```
OSOffAD(int mode);
```

Vstup (mode):
0 = full powerdown
1 = fast powerdown

Rozmístění pinů na Eyebotu:

```
Analog Input (10 pin)
Microphone is mapped to:          Analog Input 0
Battery level gauge is mapped to: Analog Input 1
 1 Vcc (5V regulated)
 2 Vcc (5V regulated)
 3 Analog Input 2
 4 Analog Input 3
 5 Analog Input 4
 6 Analog Input 5
 7 Analog Input 6
 8 Analog Input 7
```

```
 9 Analog GND
10 Analog GND
```

6.7 Obsluha digitálních vstupů / výstupů

Eyebot disponuje 8mi digitálnímy vystupy a 8mi digitálnímy výstupy. Tyto vstupy a výstupy se obsluhují třemi funkcemi:

Funkce pro zápis logické hodnoty na výstup Eyebotu:

```
OSWriteOutLatch(int latchnr, BYTE mask, BYTE value);
```

Parametry:

- Latchnr: závora – v rozmezí od 0 do 3
- Mask: maska od 0x00 do 0xFF
- Value: hodnota od 0x00 do 0xFF

Př: OSWriteOutLatch (0, 0xF7, 0x08);

Bity	7	6	5	4	3	2	1	0
Přepočet	8	4	2	1	8	4	2	1
Maska	1	1	1	1	0	1	1	1
Value	0	0	0	0	1	0	0	0

Výsledek je, že se změní pouze 4bit na 1.

Funkce pro načtení logické hodnoty ze vstupu Eyebotu:

```
OSReadInLatch(int latchnr);
```

Funkce pro načtení logické hodnoty z výstupu Eyebotu:

```
OSReadOutLatch(int latchnr);
```

Rozmístění pinů na Eyebotu:

```
Digital Input/Output (20 pin)
Infrared PSDs are mapped to Digital Output 0 and Digital Input 0..5
 1- 8  Digital Output 0..7
 9-16  Digital Input  0..7
17+18  Vcc (5V)
19+20  GND
```

6.8 Multitasking u Eyebotu

Termínem multitasking (z angličtiny, multi = mnoho, task = úloha) se v informatice označuje schopnost provádět (přinejmenším zdánlivě) několik úloh současně. Eyeboty využívá Kooperativní multitasking a Preemptivní multitasking.

6.8.1 Kooperativní multitasking

Kooperativní multitasking (též **nepreemptivní**) vyžaduje aktivní spoluúčast běžících úloh. Každá úloha je povinna dostatečně často voláním systémové funkce předat řízení zpět operačnímu systému, který díky tomu může spustit jinou úlohu, která se po chvíli opět dobrovolně vzdá procesoru atd.

Kooperativní multitasking používaly například verze Microsoft Windows do Windows 3.x[1] nebo Mac OS před Mac OS X. Zatímco je v moderních systémech kooperativní multitasking na ústupu, například RISC OS ho stále používá.

Výhodou řešení je jednodušší implementace operačního systému, podstatnou nevýhodou skutečnost, že chybně naprogramovaná úloha, která nevrátí řízení zpět operačnímu systému, způsobí úplné zastavení systému i ostatních úloh.

6.8.2 Preemptivní multitasking

Při **preemptivním multitaskingu** o přidělování a odebírání procesoru jednotlivým úlohám plně rozhoduje operační systém. V pravidelných intervalech (typicky zhruba 10 ms) přeruší provádění běžícího programu. Vyhodnotí aktuální situaci (které úlohy žádají o přidělení procesoru, jejich priority atd.) a nechá běžet buď opět úlohu, kterou přerušil, nebo jinou úlohu, která má zájem o přidělení procesoru. I v preemptivním multitaskingu však může úloha dobrovolně požádat o přepnutí kontextu, vzdát se zbytku svého kvanta, např. pokud chce na něco čekat (např. na pomalou vstupní/výstupní operaci, jako je čtení dat z pevného disku).

Preemptivní multitasking používají MS Windows od verze 95 (v řadě NT je od začátku), Mac OS od Mac OS X, jádro Unix nebo Linux od svého vzniku.

Výhodou tohoto řešení je, že nedochází k „zatuhnutí“ počítače, neboť i v případě, že úloha zhavaruje, odebere operační systém dané úloze řízení a přidělí časové kvantum ostatním úlohám. Nevýhodou je složitější implementace operačního systému a vyšší nároky na hardwarové vybavení počítače.

6.8.3 Použití multitaskingu

Multitasking nainicializujeme funkcí:

```
OSMTInit(BYTE mode);
```

Vstup: mody

- COOP=DEFAULT – pro kooperativní multitasking
- PREEMPT – pro preemptivním multitaskingu

7 Vytvořené programy v jazyce „C“ a Matlabu

Při vytváření programů v jazyce C jsem se vedl normou ANSI/ISO viz. [3]. Nejdříve jsem vytvořil kratké programy, které ovládaly jen určitou malou část periferie Eyebotu. Pro snažší otestování co a jak funguje. Programy jsem se snažil vytvářet tak, aby byly univerzální a daly se snadno dále využít k sestavení složitejších ovládacích programů celých mobilních robotů.

7.1 Programy vytvořené pro Eyebot

Soupis programů, které byly vytvořeny pro Eyebot s informacemi co dělají:

main.c - předepsaný soubor, ze kterého lze teprve začít psát vlastní program. Program má předepsané důležité knihovny, které se mají načíst. Pak má předepsaný main, ve kterém je smyčka a v ní přepínač pro čtyři tlačítka.

Názvy složek s programy:

- **ADconv** (programy obsluhující AD převodník)
- *ADconv.c* - Program pro čtení analogové hodnoty z AD převodníku kanal 2 (10bit (0-1023)

- **BT** (programy pro přenos dat přes BlueTooth)
- *download_eye_BT_01.c* - programy na přijímání znaků z Eyebotu do PC přes BlueTooth při rychlosti SER9600
- *upload_eye_BT_01.c* - programy na odesílání znaků z Eyebotu do PC přes BlueTooth při rychlosti SER9600

- **DCmot** (programy pro řízení DC motorů a čtení hodnot z enkoderů)
- *Motor.c* – program roztočí motor A i B na 50%, program se ukončí stikem 4ho tlačítka
- *Motor_ncoder_01.c* - Program slouží k ovládání DC motoru a enkoderu

1. motor připojíme na výstup A s enkoderem
2. +/- od -100 do 100 měníme rychlost otáčení

3. Bez čtení hodnot z enkoderu

- *Motor_ncoder_02. c* - Program slouží k ovládání DC motoru a enkoderu

1. motor připojíme na výstup A s enkoderem
2. +/- od -100 do 100 měníme rychlost otáčení
3. na displeji se nám ukazuje hodnota čtená z enkoderu

- *Motor_ncoder_03.c* - Program slouží k ovládání DC motoru a enkoderu

1. motor připojíme na výstup A s enkoderem
2. +/- od -100 do 100 měníme rychlost otáčení
3. na displeji se nám ukazuje hodnota čtená z enkoderu a to absolutní počet děr a počet celých otáček

- **DigiIO** (programy pro digitalní přenos dat, přes digitální vstupy a výstupy na Eyebotu)
- *digiIO_01.c* - Program slouží k otestování rychlosti odezvy EYEBOTU:

1. připojím Digital IO k měřící desce v PC (stačí spojit první piny IO a GND)
2. ze Simulinku pomocí RealTime Toolbox pošlu na Digi IN EyeB. logickou 1
3. program vyhodnotí, že dostal 1 a okamžitě Digi OUT EyeB. nastaví také na 1
4. ve Simulinku se mi v Grafu ukáže prodleva náběžných hran

- **Jaromír** (programy pro řízení čtyřnohého robotu s 12 servopohony)
- *4noh_03.c* - Program bude ovládat 4nohý robot Jaromir

1. zinicializuje serva 02 az 14
2. postupně bude číst znaky (hodnoty 0-255) jako servo_value
3. servo se nastaví podle hodnoty servo_value
4. odešle zpět do PC potvrzení o tom, že mu došlo 12 hodnot pro serva a tím PC může poslat další, aby se nepřeplnil buffer pro seriovou komunikaci

Chyby: tento program rychle po sobě inicializuje všechna serva a tím se překročí povolený napájecí proud, bez nastavení ochrany na zdroji, což je proud do 3A by mohlo dojít i k poškození Eyebotu.

- *4noh_04.c* – Stejně jako verze 4noh_03.c. Přidáno:
  1. Inicializace je zpožděná
  2. Vypisuje se na LCD které servo se inicializuje
  3. První pokus o komunikační protokol, bohužel nefunkční Eyebot začne číst špatné hodnoty z baffru od seriového přenosu
- *4noh_05.c* – pokračování ve verzi 4noh_04 – zdokonalování komunikačního protokolu

- **Seriál** (programy pro komunikaci s PC po seriové lince)
- *dl_eye.c* – program opakovaně čte hodnoty z baffru, které mu přijdou z PC po seriovém kabelu a zobrazí je na LCD jako znaky. Pokud to nejsou známé znaky, vypíše jen prázdnou mezeru. Umí vypsat jen písmena, číslice a určité symboly. Program se ukončí stlačením 4tého tlačítka END.
- *download_eye.c* – takřka totožný jako program dl_eye.c, ale program přečtě hodnotu jen při stisku tlačítka DL.
- *download_eye1.c* – program se chová stejně jako program dl_eye.c, ale čtení se provádí přes příkaz fscanf.
- *upl_dl.c* - program na odesílání/primární znak eyebot/PC, data čtu pomocí Docklight
  - KEY1: odešle jeden znak A do PC
  - KEY2: příjme jeden znak z PC
- *upload_eye1.c* – program na odesílání dat do PC pomocí OSSendRS232, data čtu pomocí Docklight s nastavenou rychlostí 115200 b/s
  - KEY1: odešle jeden znak A do PC
  - KEY2: odešle řadu znaků (0123456789) do PC
- *upload_eye2.c* – program na odesílání dat do PC pomocí fprintf, data čtu pomocí Docklight s nastavenou rychlostí 115200 b/s
  - KEY1: odešle jeden znak d do PC

  - KEY2: odešle jeden znak 2 do PC
- *upload_eye3.c* – stejný jako upload_eye2.c odesílá „K“ nebo „2da“, data čtu pomocí Matlabu, kde jsem pro to vytvořil program serila_download_3char.m
- *upload_eye4.c* - stejný jako upload_eye3.c odesílá „short x=326;“

- **servo** (programy pro ovládání servopohnů)
- *servo1_toc.c* - Program na ovládání serva. Servo je připojeno jako SERVO1
  - KEY1: <- ubírá stupně
  - KEY2: -> přidává stupně
  - KEY4: End
- *toc14_01.c* - Program nainicializuje všech 14 serv a bude se všemi souběžně hýbat.
- *toc14_2.c* - Program nainicializuje 12 serv a bude se všemi souběžně hýbat. Program se chová stejně jako program 01.c, ale je v něm zjednodušený přístup k ovládaní servopohonu, je tedy i kratší.

- **servo_kabel** (programy pro řizení serva z PC pomoci dat posílaných přes seriovou linku).
- *ser_kabel_ch_02.c* – program ovládá servo připojené jako servo01, podle toho jaké hodnoty pošlu z PC po seriové lince. Na PC použiji Docklight, ze kterého odesílám hodnoty od 0 do 255 s nastavenou rychlostí 115200 b/s.
- **time** (programy na obsluhu časovačů na Eyebotu)
- *systime.c* – Program na zobrazování systémového času, je to doba od posledního spuštění nebo resetu Eyebotu. Program se ukončí stiskem libovolného tlačítka.
- *timer.c* – Program využívá 3 přerušení od časovačů. Když přijde přerušení od časovače, tak se zavolá funkce, která do proměnné, která slouží jako čítač, přičte 1. Periody časovašů jsou nastaveny na 0.01s, 0.02s a

0.03s. Každý z časovačů lze zastavit příslušným tlačítkem Ir1, Ir2 a Ir3. Program se ukončí stlačením tlačítka END.

- *timer_01.c* – program po spuštění vynuluje systémový čas. Stiskem prvního tlačítka se vypíše čas od spuštění programu.
- *timer_02.c* – program každých 10s přičte do čítače 1 a každou 1s pípne.

7.2 Programy vytvořené pro Matlab 2008a

Programy nalezneme ve složce Matlab2008a.

- **Xserial** (programy pro komunikaci s Eyebotem po seriové lince)
  - *serila_download_3char.m* – program pro čtení hodnot, které příjdou do PC z Eyebotu po seriové lince. Program využívá funkci xsrial pro přenos dat po seriové lince. Rychlost přenosu je nastavena na 115200 b/s. Pro odesílání dat z Eyebotu jsem vytvořil program *upload_eye4.c*
  - *serial_upload_1char.m* – program na odesílání hodnot z PC do Eyebotu. Rychlost přenosu dat 115200b/s. Pro čtení dat v Eyebotu jsem vytvořil program *download_eye.c*
  - *seriál_upload_ASS.m* – program pro odeslání ASS (aktuální stav serv, por 12 serv). Přípravný program pro řízení čtyřnohého kráčejícího robotu.

- **Chuze** (programy pro řízení čtyřnohého robotu po seriové lince)
  - *eyebot_start_krok_X_01.m* – program pro výpočet pohybu jednotlivých noh čtvyřnohého kráčejícího robotu. Program spočítá všechny stavy servopohonů v průběhu jednoho kroku robotu a poté je odešle po seriové lince do Eyebotu. Bohužel je tu nedopracovaný komunikační protokol, program odešle všechna data naráz a Eyebot je nestačí zpracovat.
  - *eyebot_start_krok_X_02.m* – vychází z verze *eyebot_start_krok_X_01.m*, s dopracovaným komunikačním protokolem.
- **digitalIO** (modely v Simulinku s využitím RealTime Toolbox)
  - *mereni_digiIO_2pin.mdl* – model s manuálním přepínačem na odeslání logické 0 nebo 1. Po startu modelu manuálně přepneme přepínač z 0 na 1 a Eyebot

nám hned pošle log. 1 na vstup. Výsledek zpoždení se nám ukáže v grafu jako zpoždění přepnutí z 0 do 1. V Eyebotu musíme mít spuštěný program *digiIO_01.c*

- *mereni_digi_IN_8.mdl* – model, který nám přečtě 8bit ze vstupu měřící karty.
- *mereni_digi_OUT_8.mdl* - model, který nám mění 8bit na výstupu z měřící karty.

8 Provedené experimenty

Při experimentech s řízením jsem vycházel ze skládání jednoduchých programů, které jsem si dříve vytvořil.

8.1 Testování řízení servopohonů

Bylo potřeba otestovat, jestli Eyebot zvládne řídit větší počet servopohonů naráz. Vycházel jsem z programu *servo1_toc.c* a *toc14_01.c.*

8.1.1 Připojení sevopohonu k Eyebotu

Serva se zapojují ze spod Eyebotu dle obrázku Obr. 12. Celkem je jich možno připojit a řídit 14 s nahranou verzí HDT (Hardware Description Table), nejlépe hdt-M5-servo.hex najdete ji na C:\EyeBot\robios\hex a zdrojový C-file hdt-M5-servo.c na C:\EyeBot\robios\dhtdata.

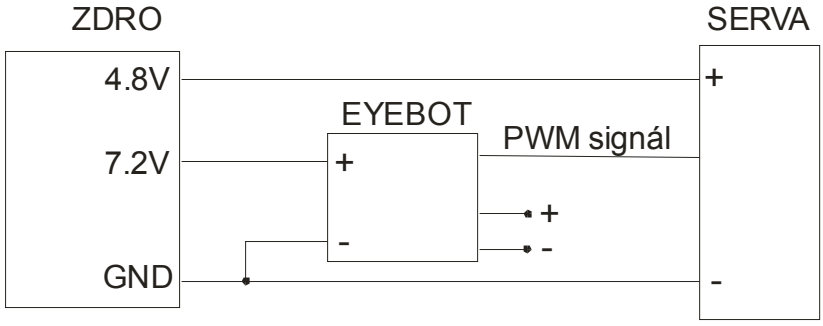
Obr. 12 Způsob připojení serv

8.1.2 Změření PWM signálu pro serva

Nejdříve jsem si na osciloskopu naměřil výstupní PWM signál pro servopohony. Tento signál odpovídal požadavkům podle předpokladů pro řízení servopohonů (viz. [5] kap. 3. Regulační vlastnosti elektrických pohonů a výkonových členů). Střídu signálu můžeme nastavit od 899µs do 2105µs. Tento signál nám generoval Eyebot při použití programu *servo1_toc.c* a s nastavenou hodnotou, při které by se servo natočilo na 45°, což odpovídá střídě 1502µs.

8.1.3 Test zapojení více servopohonů

Při tomto testu jsem vycházel z programu *toc14_01.c.* Kde program zinicializuje všech 14 servopohonů naráz a se všemi zaroveň začne hýbat od nulové výchylky do maximální. Eyebot nemá problém ovládat všechny serva naráz, ale je tu výkonový problém. Eyebot je možné používat do maximálního proudu 3A. Pak dojde k vypnutí pojistky. Tento poblém je možno řešit externím napajením serv a využívat tedy jen řídícího signálu z Eyebotu Obr. 13. Proto je zapotřebí vytvořit redukci na zapojení serv k externímu zdroji napájení a k signálu z Eyebotu.



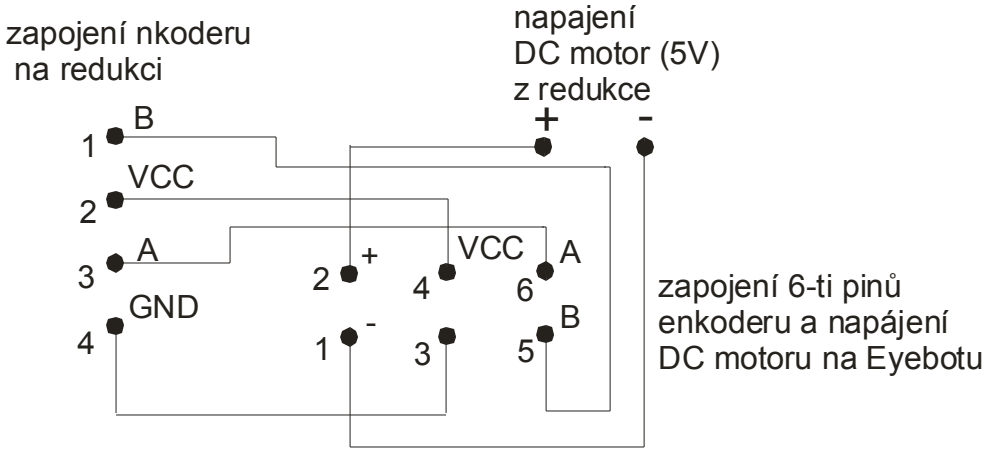
Obr. 13 Externí napájení servopohonů připojených k Eyebotu

8.2 Testování řízení DCmotoru a čtení hodnot z encoderu

Bylo potřeba vyzkoušet chování řizení DCmotorů a zároveň čtení hodnot z enkoderu. K tomuto učel jsem si vytvořil program *Motor.c* a *Motor_ncoder_03.c.*

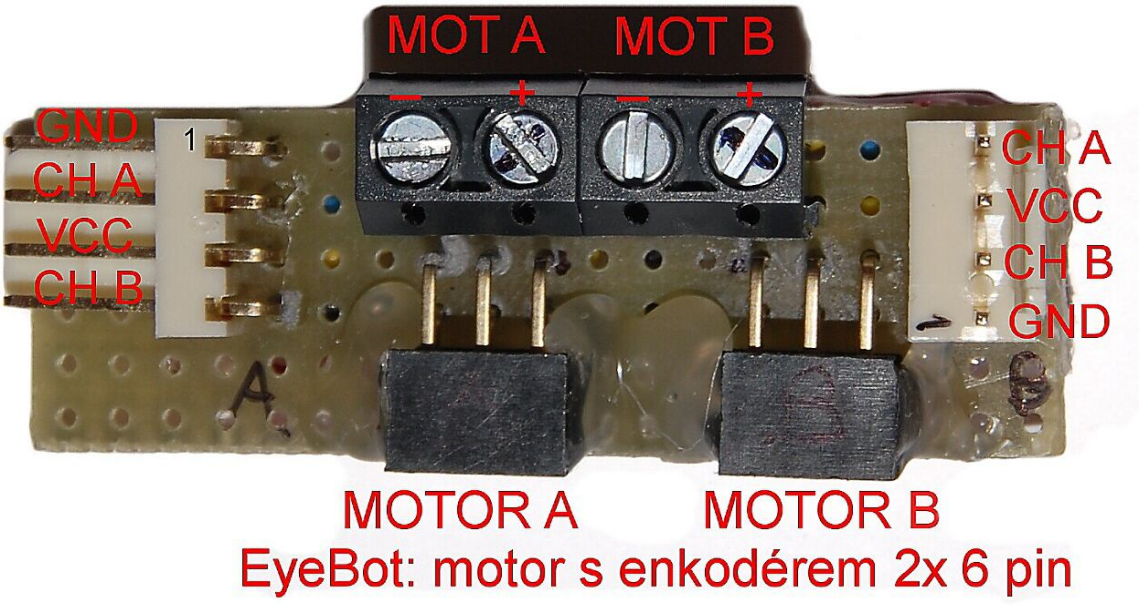
8.2.1 Vytvoření redukce pro připojení DCmotoru a enkodéru

Pro připojení dvou DCmotorů s enkodery jsem vytvořil redukci, která usnadňuje připojení a také zabraňuje svým tvarem, aby došlo ke špatnému zapojení jednotlivých pinů. Vycházel jsem z návrhu schématu zapojení jednotlivých pinů Obr. 14



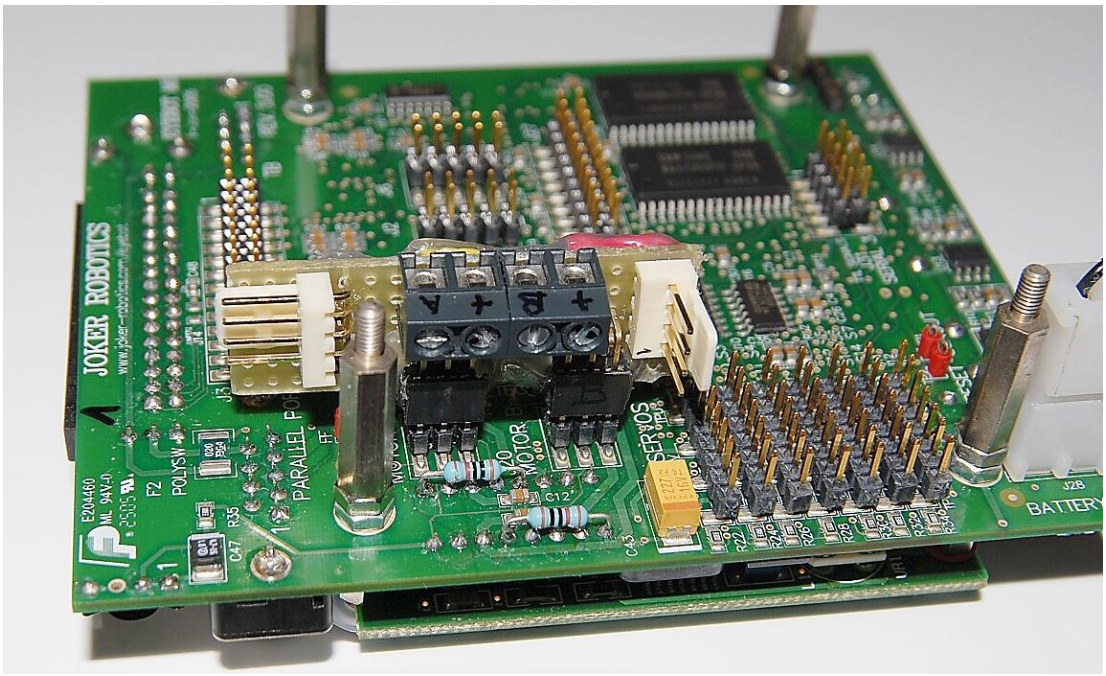
Obr. 14 Schema zpojení redukce

Redukci jsem vytvořil na pajecím poli drátovým propojením jednotlivých pinů. Hotová redukce s popisem je na Obr. 15



Obr. 15 Vytvořená redukce

Na Obr. 16 je vydět zapojená redukce ze spod Eyebotu, kde je 2x 6 pinů pro připojení dvou DC motorů A a B a enkodéry. Do svorek se připojuje motor plusovým a minusovým vývodem. Enkodery se připojují na 4pinové konektory se vstupy pro signál A a B.



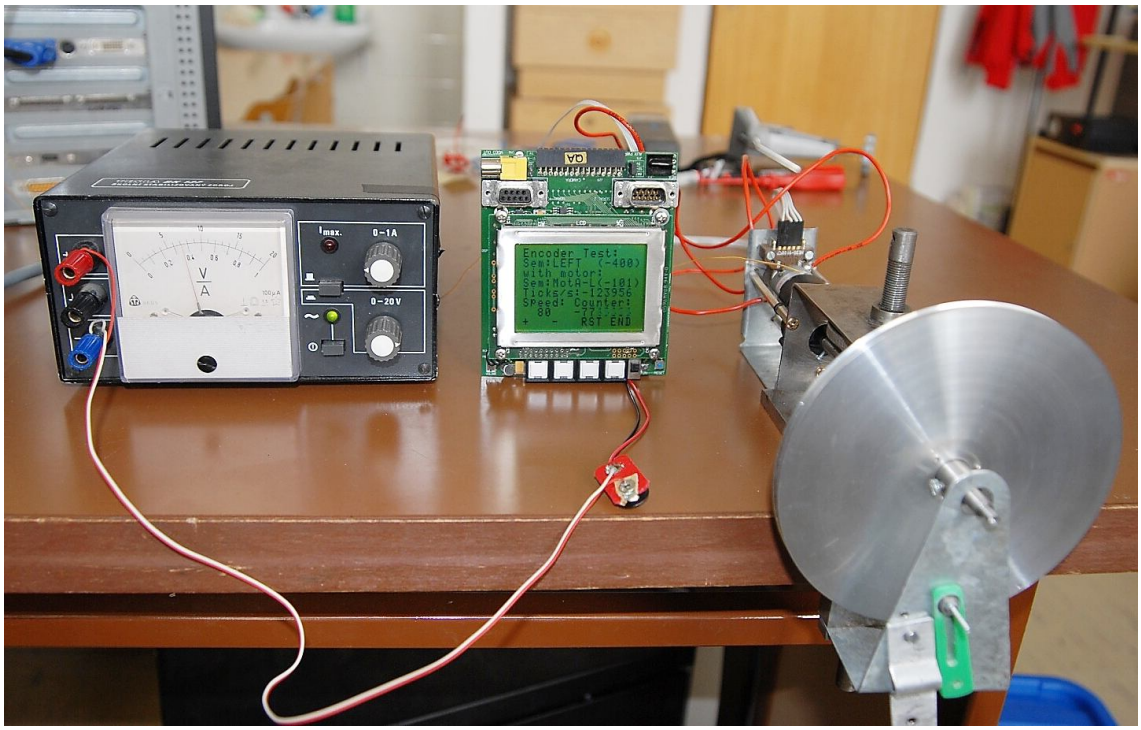
Obr. 16 Zapojení redukce DC motorů s enkodéry

8.2.2 Otestování výstupu z Eyebotu

Nejdříve jsem si změřil výstupní signál pro DCmotor. Použil jsem program *Motor.c*, který mi nastavil výstup tak, že by se motor točil na 50% rychlosti. Signál jsem změřil na osciloskopu.

8.2.3 Pokus s experimentálním DCmotorem

Při otestování řízení motoru a souběžného čtení hodnot z enkodéru jsem použil předvytvořený program: *Motor_ncoder_03.c.* Tento program umožnuje měnit rychlost otáčení od zaporných hodnot do kladných od 0% do 100%, kde 100% odpovídá napájení 5V. Opět zde musíme hlídat proudový odběr, aby nepřekročil 3A.



Obr. 17 Eyebot propojený s experimrntálním DC motorem a enkodérem

8.3 Testování digitálních vstupů a výstupů

Pro testování digitálních vstupů a výstupů jsem si vytvořil programy *digiIO_01.c* pro Eyebot a modely pro PC do Simulinku s využitím RealTime Toolbox a to jsou *mereni_digi_IN_8.mdl*, *mereni_digi_OUT_8.mdl* a *mereni_digi_IO_2pin.mdl*.

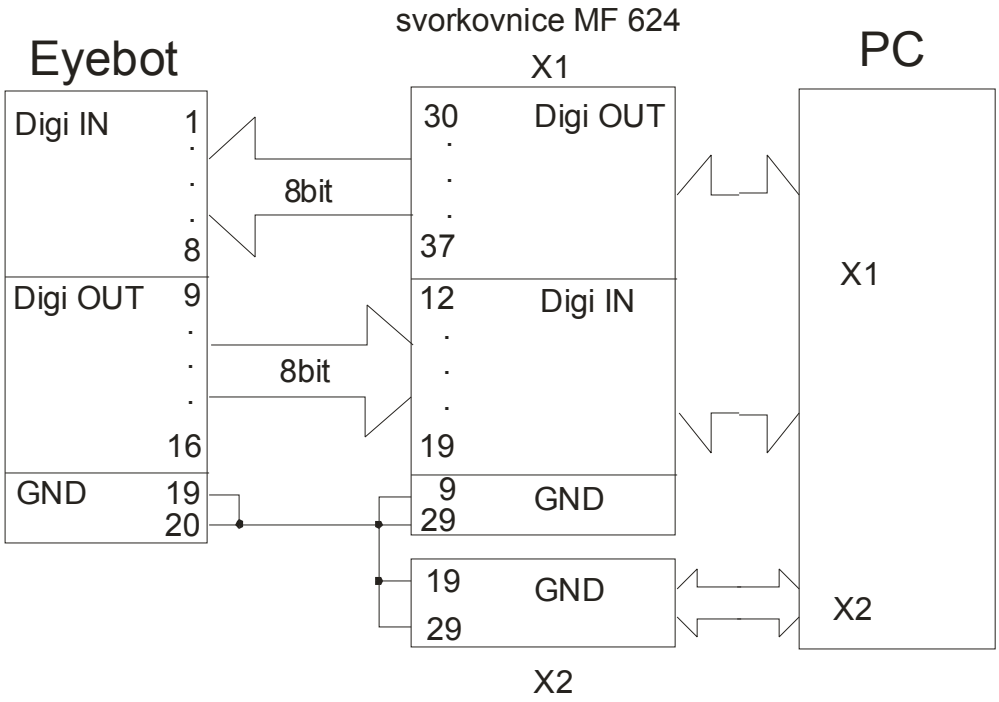
8.3.1 Návrh zapojení digitálních vstupů a výstupů Eyebotu

Bylo zapotřebí propojit digitální vstupy a výstupy Eyebotu a měřící karty v PC přes sběrnici MF624. Pro správné zapojení jsem si vyhledal v manuálu pro zapojení sběrnice tabulku Tab.: 8.3.1. s očíslovanými svorkami na sběrnici.

AD0	1	20	DA0
AD1	2	21	DA1
AD2	3	22	DA2
AD3	4	23	DA3
AD4	5	24	DA4
AD5	6	25	DA5
AD6	7	26	-12V
AD7	8	27	+12V
AGND	9	28	+5V
DA6	10	29	GND
DA7	11	30	DOUT0
DIN0	12	31	DOUT1
DIN1	13	32	DOUT2
DIN2	14	33	DOUT3
DIN3	15	34	DOUT4
DIN4	16	35	DOUT5
DIN5	17	36	DOUT6
DIN6	18	37	DOUT7
DIN7	19		

Tab.: 8.3.1 Zapojení jednotlivých svorek na sběrnici MF 624

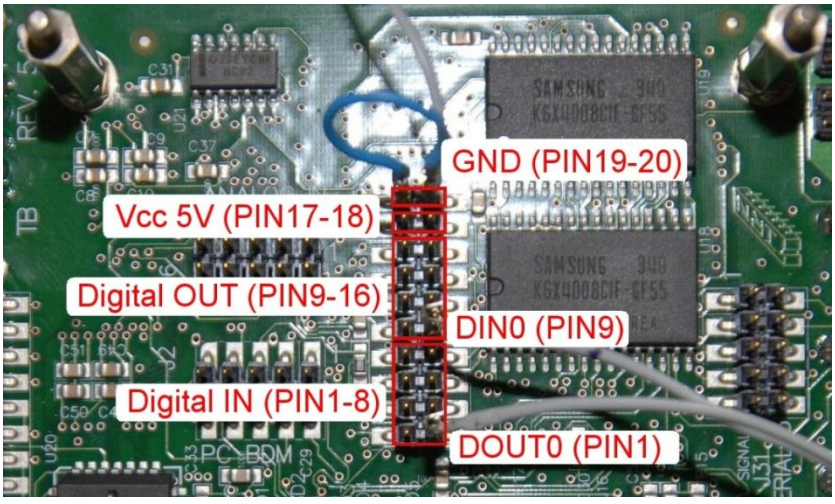
Poté jsem si navrhl schéma propojení Eyebotu se sběrnicí a s PC (Obr. 18).



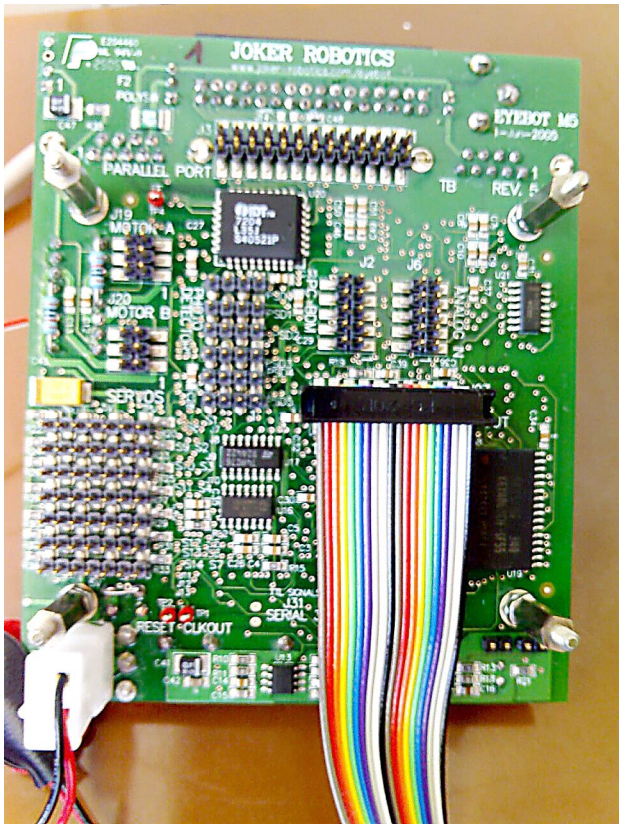
Obr. 18 Schéma propojení Eyebotu s PC

8.3.2 Zapojení Eyebotu a měřící karty v PC

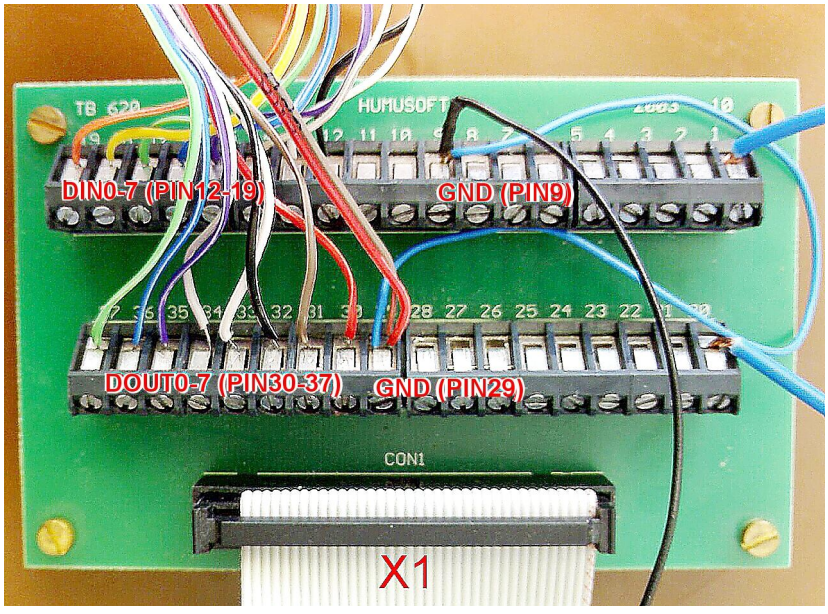
Pro propojení Eyebotu se sběrnicí MF 624 je potřeba 20ti žílový kabel, pro propojení všech pinů Obr. 19.



Obr. 19 Zapojení Digitálních vstupů/výstupů na Eyebotu



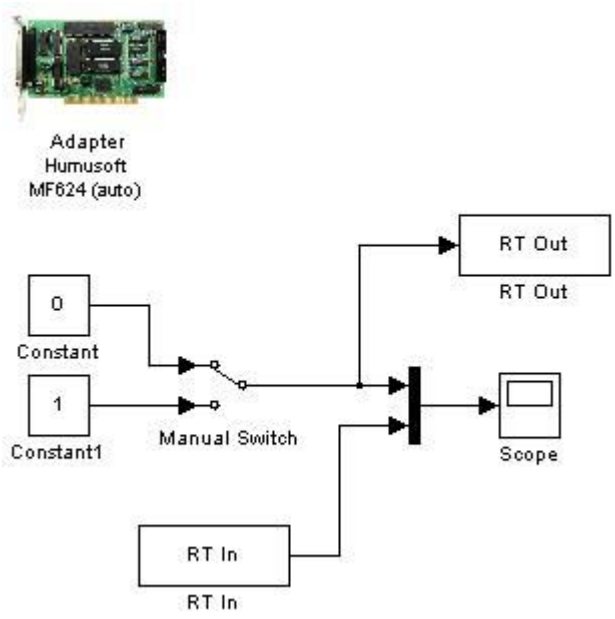
Obr. 20 Propojovací kabel (20pin) mezi Eyebotem a svorkovnicí MF 624



Obr. 21 Zapojení svorkovnice MF 624

8.3.3 Programy pro vyhodnocování digitálních vstupů a výstupů na PC

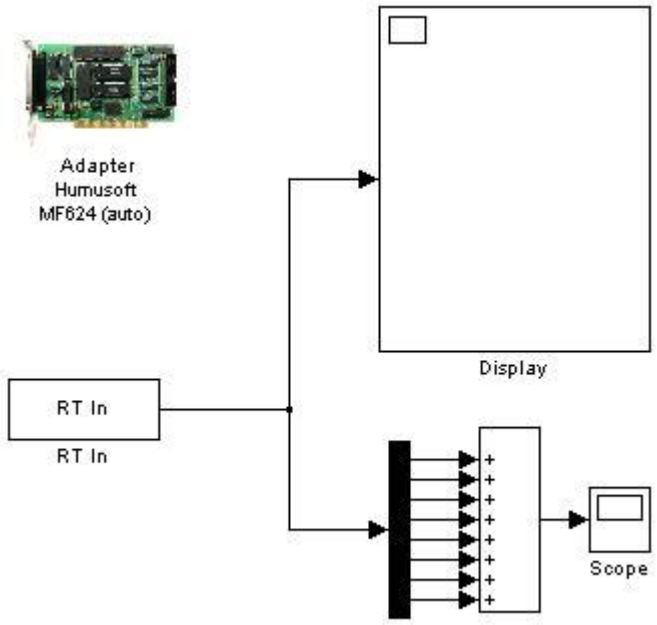
Pro naměření rychlosti odezvy digitálních vstupů a vystupů Eyebotu jsem vytvořil model v Simulinku, který pomocí RealTime Toolbox obsluhuje měřící kartu se sběrnicí MF624. Model: *mereni_digiIO_2pin.mdl* – (Obr. 22) je model s manuálním přepínačem na odeslání logické 0 nebo 1. Model se spouští s přepínačem na hodnotě 0. Manuálně přepneme přepínač z 0 na 1 a Eyebot nám hned pošle log. 1 na vstup měřící karty. Výsledek zpoždení se nám vykreslí do grafu. Graf nám zaznamená časový průběh odesílané hodnoty a přijímané hodnoty. V Eyebotu musíme mít spuštěný program *digiIO_01.c*. Tento program po spuštění čeká na to, až mu na vstup přijde logická 1 a poté ihned přepíše výstup taky na logickou 1 a program skončí.



Obr. 22 Simulink - model na testování rychlosti odezvy Eyebotu

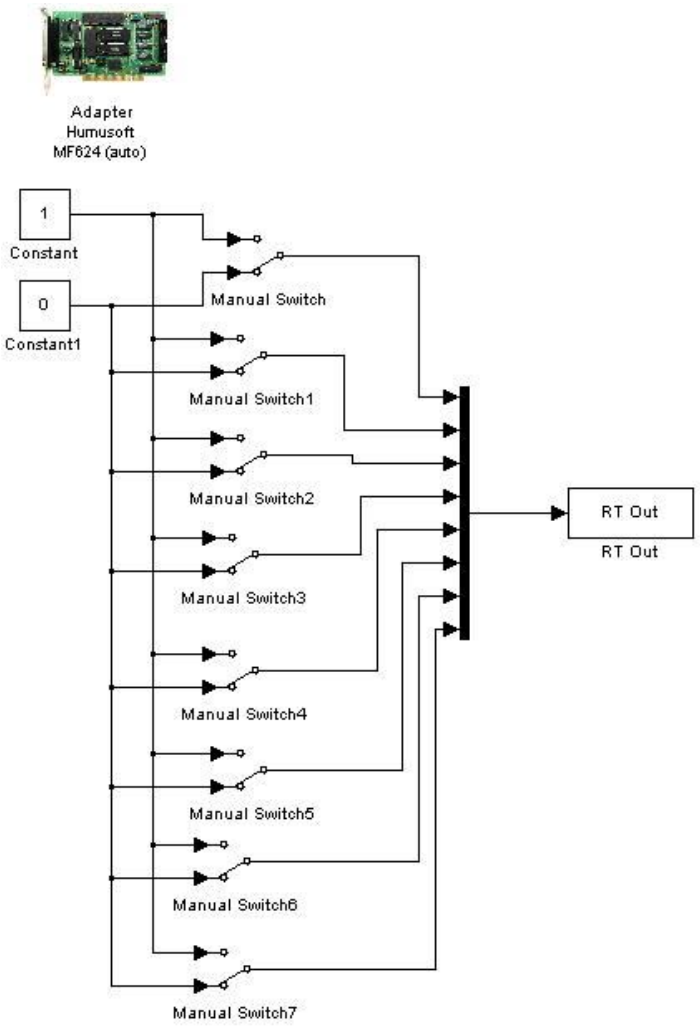
Při použití programu *digiIO_01.c*, který neprováděl žadný jiný úkol než, že vyčkával na hodnotu a okamžitě odpověděl, jsem naměřil zpoždení 0,009s . Při použití programu *digiIO_02.c*, který pracuje jako program verze 01, ale Eyebot je navíc zaměstnáván třemi přerušenímy od časovačů, které se volají po 0.01s 0.02s a 0.03s. V tomto případě jsem po opakovaném měření naměřil nejvyšší zpoždení 0,08s.

Další model: *mereni_digiIO_IN_8.mdl* – (Obr. 23) nám přečtě hodnoty z výstupu Eyebotu, které jsou připojeny na vstupy měřící karty. Na displayi se nám budou zobrazovat jednotlivé bity a na Scope se nám vykresluje součtová hodnota všech bitů.



Obr. 23 Simulink - model na čtení hodnot z digitálního vstupu do PC

Další model: *mereni_digiIO_OUT_8.mdl* – (Obr. 24) nám odešle hodnoty na vstup Eyebotu. V modelu máme 8 manuálních přepínačů, pomocí kterých můžeme změnit hodnotu na jednotlivých bitech.



Obr. 24 Simulink - model na vysílání dat po digitálním výstupu z PC

9 Řízení čtyřnohého robotu – Jaromír

Řízení čtyřnohého robotu probíhá pomocí kinematických vypočtů za využití Matlabu a prostřednictvím seriové linky se za pomoc komunikačního protokolu v Eyebotu řídí pohyb robotu. Tedy bylo zapotřebí nejdříve vytvořit komunikační protokol mezi Eyebotem a PC. Tato komunikace muže probíhat za použití kabelu určeného pro seriovnou komunikaci nebo za pomoci Bleotooth modulu, připojeného k Eyebotu.

9.1 Komunikační protokol přes seriovou linku

Po seriové lince může Eyebot a PC komunikovat prostřednictvím paketů, které můžou mít ruzné parametry a velikosti.

Pakety jsou složeny ze čtyř hlavních částí:

- startovní bity
- hlavička
- data
- CRC součet

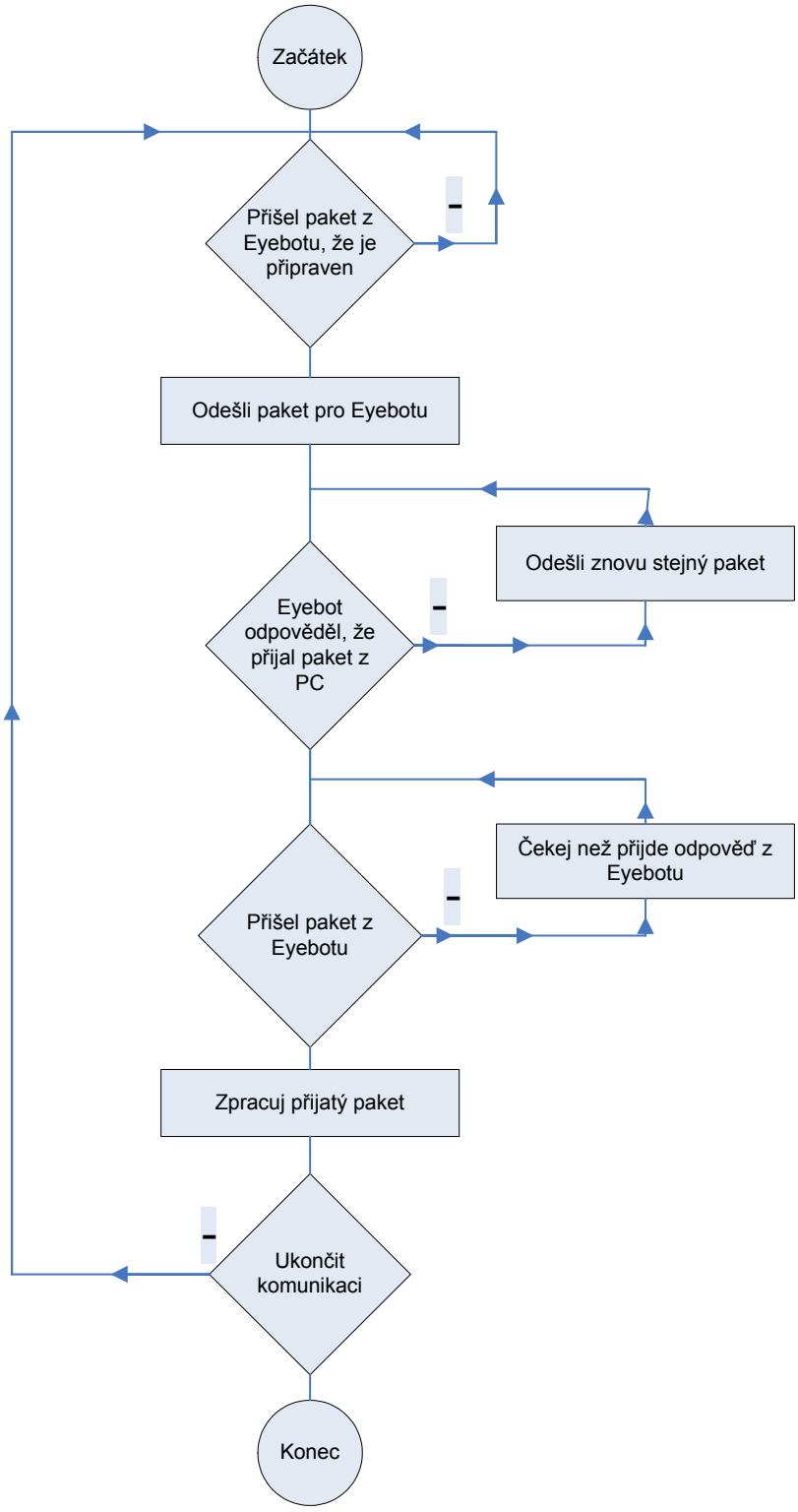
Jsou použity dva startovní bity jako rozumný kompromis mezi pravděpodobností výskytu a jejich velikostí. Hlavička obsahuje adresu příjemce, adresu odesílatele, velikost přenášených dat a typ dat. Každá z těchto informací má velikost 1 bit, tudíž hlavička má celkem 4 bity. Za hlavičkou již přímo následují posílaná data. Konec paketu uzavírá kontrolní součet o velikosti 2 bity, slouží k odhalení případné chyby v paketu. Schéma je na Obr. 25

Obr. 25 Schéma paketu

9.1.1 Komunikační program pro PC

Program pro komunikaci na PC jsem vytvořil v programovém prostředí Matlab 2008a. Řídící algoritmus komunikace na PC zastává funkci MASTER pro řízení ko-

munikace. Tedy počítač bude rozhodovat o tom, kdy komunikace začne a kdy se ukončí.



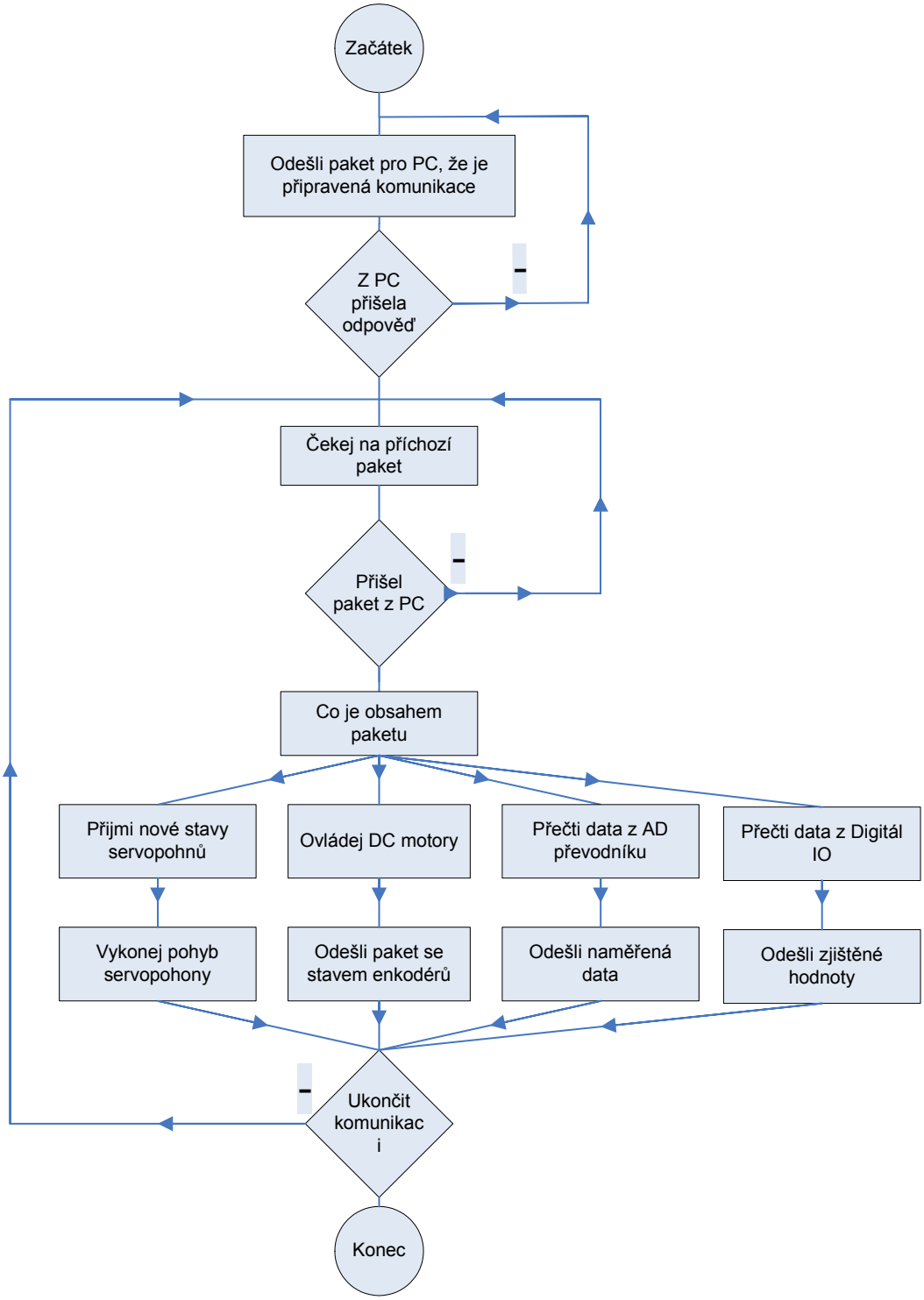
Schema. 1: Blokové schema komunikačního algoritmu na PC

Komunikační algoritmus (viz. Schema. 1) nejdříve vyčká na příchod paketu z Eyebotu, který obsahuje údaj o tom, že je Eyebot spuštěn a připraven k přijimaní paketu z PC. Poté se odešle paket z PC obsahující řídící údaje pro Eyebot, které se zpracují a podle nich se budou řídit různé periferie Eyebotu jako například servopohony, DC motory a ostatní periferie. Po odeslání tohoto paketu PC vyčkavá na paket z Eyebotu, který potvrdí doručení paketu. Kdyby toto potvrzení nebylo doručeno, PC odešle znovu stejný paket a opět vyčká na potvrzení. Tím se zajistí, že nedojde ke špatnému řízení Eyebotu tím, že se ztratí některé řídící pakety. Když se z PC odešle paket s dotazem na některé informace, například z měřících periferií, tak poté algoritmus vyčká na odpovídající paket s naměřenými hodnotami a poté PC tento paket zpracuje. Bez použití Multitaskingu komunikace probíhá vždy jen jedním směrem. Tato komunikace je bezpečnější, ale pomalejší.

9.1.2 Komunikační program po Eyebot

Algoritmus pro komunikaci pro Eyebot je vytvořen tak, že Eyebot bude v režimu SLAVE. Teda bude přijímat rozkazy od PC a poté vykonávat příkazy, které mu budou přicházet v jednotlivých paketech.

Po spuštění komunikačního algoritmu na řídící jednotce Eyebot jednotka nejdříve vyšle paket pro PC s informací o tom, že je připravena ke komunikaci a může tedy přijímat pakety. Po té už jen Eyebot vyčkává na přicházející pakety, které po přijetí rozhodnou o tom co bude provedeno, které periferie se budou ovládat. Po vykonání řízení serv nebo DC motorů se odešle zpět do PC paket s informací, že úkon byl vykonán. Když přijde paket s dotazem na některé z měřících periferií, odešle Eyebot paket s naměřenými hodnotami. Tento postup se opakuje do té doby, než dojde k ukončení komunikace.



Schema. 2 Blokové schema komunikačního algoritmu pro Eyebot

9.2 Komunikace PC a Eyebotu připojeného k čtyřnohému robotu

9.2.1 Připojení Eyebotu ke čtyřnohému experimentálnímu robotu „Jaromir“

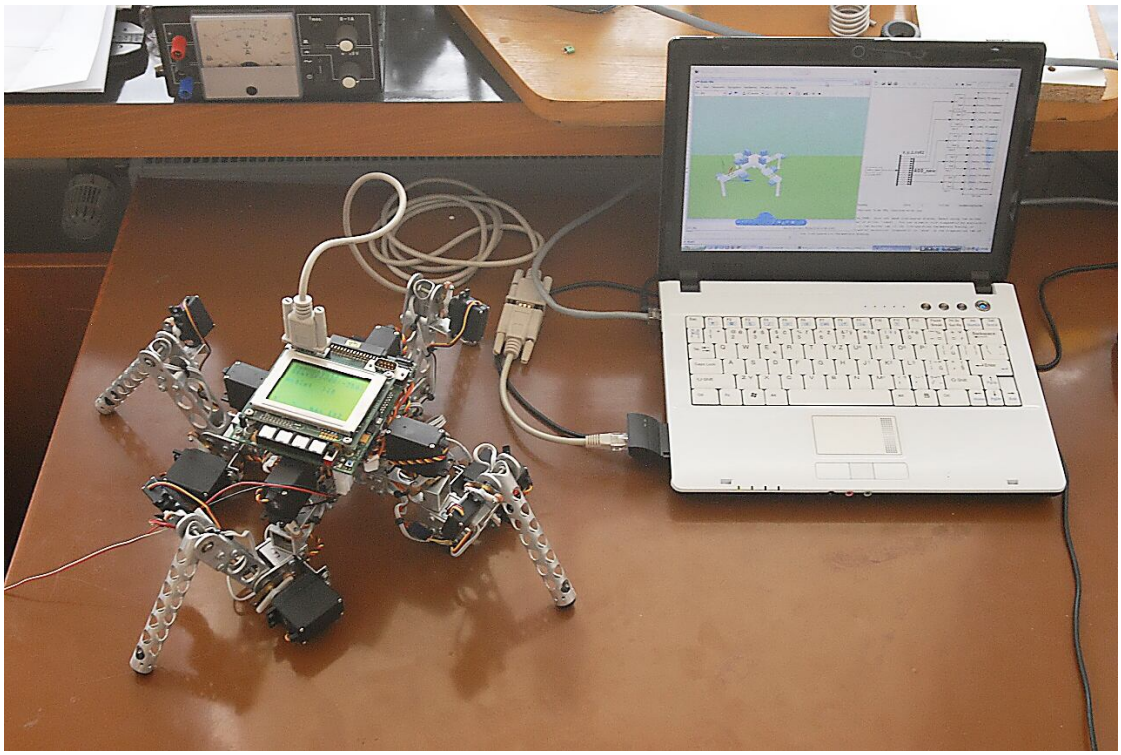
Čtyřnohý robot Jaromír [2] je poháněn 12ti servopohony. Každá noha se ovládalá 3mi servy. Serva jsem připojil na piny Eyebotu pro servo 00 až servo 11.

Bohužel použitá serva na robotu Jaromir mají větší proudový odběr než povolený limit na Eyebotu, což jsou 3A, proto je zapotřebí serva napojit na externí napájení a nejde je tedy přímo napojit na Eyebot, kdy by mohlo dojít k poškození Eyebotu. Jak připojit serva k externímu zdroji napájení jsem řešil v kapitole 8.1.3. Tedy za pomocí redukce připojíme servopohony k externímu zdroji napájení a k řídícímu signálu z Eyebotu. (viz. Obr. 13 Externí napájení servopohonů připojených k Eyebotu)

9.2.2 Komunikační programy

Jelikož jsem věděl, že budu nejdříve ovládat pouze 12 servopohonů, tak můj první program na zpracování paketů *4noh_06.c* uměl přijmout zjednodušený paket pouze omezený na dva startovací bity 0xFA a 0x34. Další bit byla hodnota, která nám určovala, kterým servem se má hýbat a čtvrtý bit byla hodnota, na jakím směre se má servo natočit.

Další program *4noh_07.c* jsem rozšířil. Prvně se pošlou startovací bity. Vynechal jsem adresy přijemce a odesílatele, protože komunikace bude probíhat jenom mezi EyeBotem a PC, a rovnou pokračuji. Třetí bit určuje velikost dat a čtvrtý bit nám určuje, jakou funkcí se mají data zpracovávat. Když má hodnotu *„A“* znamená to, že data budou hodnoty servopohonu a spustí se procedura *ASS_prijmy(void).* Tato procedura nám přijme hodnoty pro všech 12 servopohonů a uloží je do proměnné *ASS[12]*, což je pole o 12ti hodnotách. Každá hodnota je aktuální stav servopohonu. Je to globální proměnná.

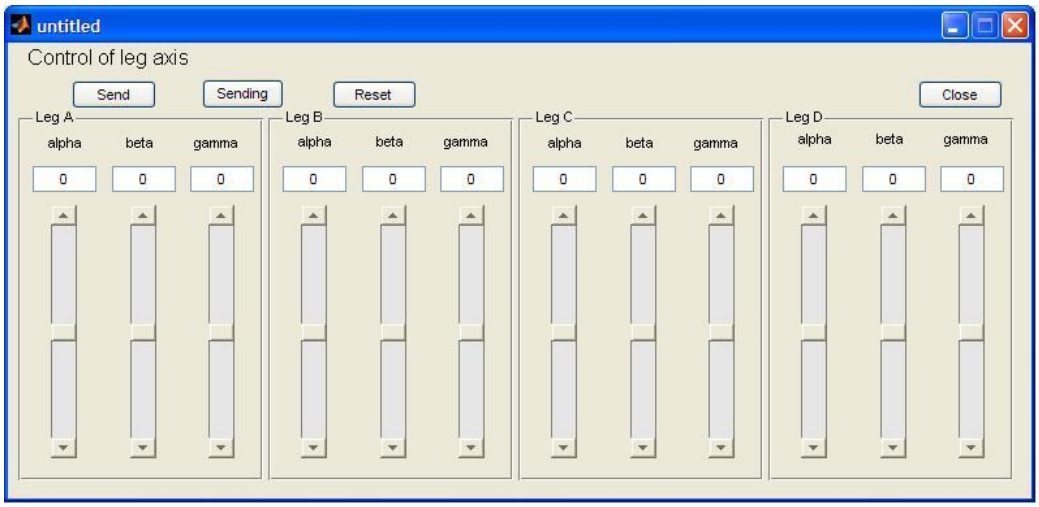


Obr. 26 Čtyřnohý robot s řidicí jednotkou Eyebot připojenou k PC přes serovou linku

9.2.3 GUI pro ovládání jednotlivých serv u experimentálního robotu

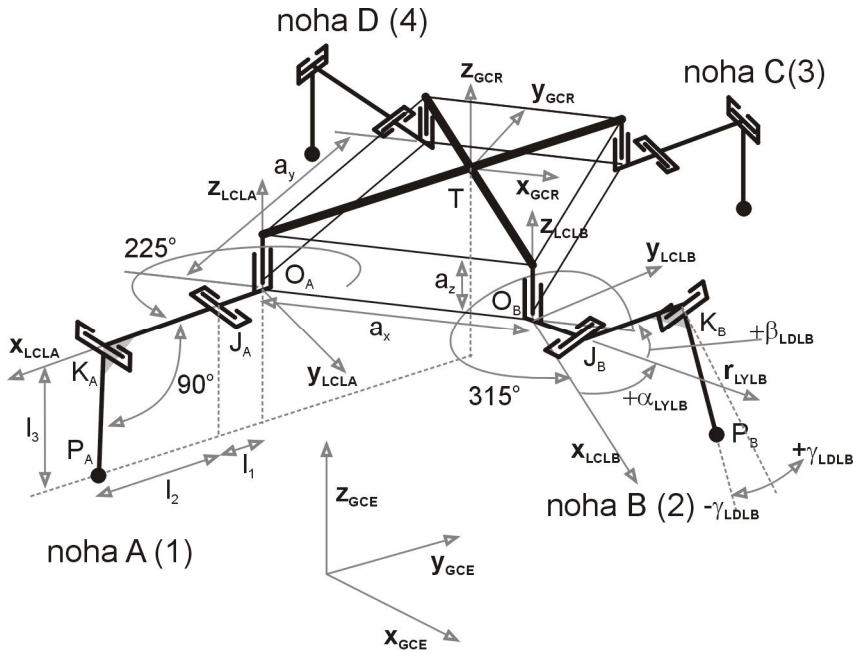
Pro jednoduché a snadné ovládání pohybu čtyřnohého robotu jsem vytvořil v programovém prostředí Matlab 2008a grafické uživatelské rozhraní.

GUI „Control of leg axis“ (na Obr. 27) je možno ovládat natočení jednotlyvých servopohonů umístěných v nohách robotu.



Obr. 27 GUI pro ovládání úhlů natočení jedntlivch serv v nohách robotu

Ovládání vychází z LYL – lokální cylindrického souřadného systému nohy robotu, který je znázorněn na Obr. 28 (viz. [2] kapitola:.2.2.2 Geometrie čtyřnohého robotu).

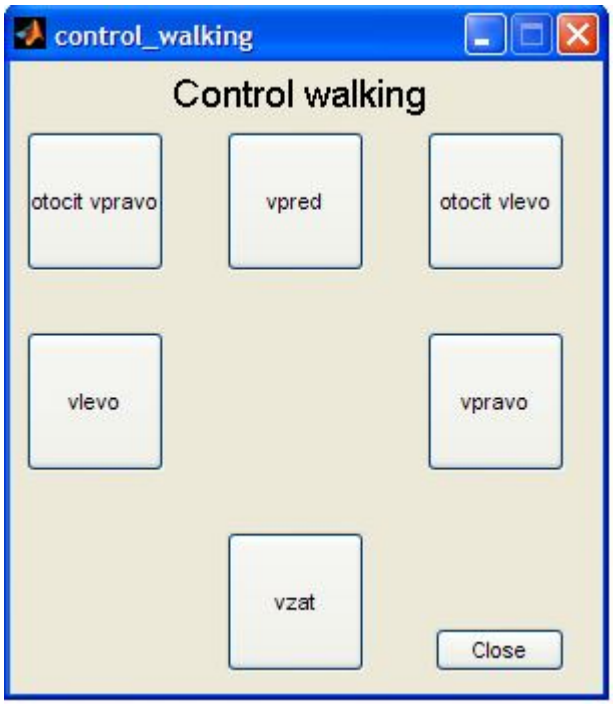


Obr. 28 Čtyřnohý experimentální robot s popsanými souřadnými systémy

9.2.4 GUI pro řízení směru chůze experimentálního robotu

Další grafické uživatelské rozhraní jsem vytvořil pro ovládání směru chůze čtyřnohého experimentálního robotu „Jaromír“ viz. Obr. 29. Způsob chůze jsme navrhl již v mé bakalářké práci „Kinematické řízení a vizualizace virtuálního prototypu čtyřnohého kráčejícího robotu“ viz. [2] kapitola 4. Chůze čtyřnohého robotu po neregulérním terénu. Program vytvořený pro ovládání chůze čtyřnohého robotu vypočítával pomocí inverzního kinematického modelu natočení jednotlivých servopohonů podle zamýšleného cílového bodu, kam měla noha robotu došlápnout a dále vypočítaval natočení servopohonů, když se hýbalo tělo robotu a koncové body nohou měly zůstat v klidové poloze. Veškeré vypočtené hodnoty byly dále zpracovávány řídícím programem, který tyto hodnoty posílal do VRML prostředí, ve kterém se zobrazoval pohyb modelu experimentálního kráčejícího čtyřnohého robotu.

V novém řídícím programu se tyto vypočtené hodnoty pro natočení serv při pohybu robotu odesílají prostřednictvím seriové linky do řídíci jednotky Eyebot a ta poté vykonává řízení pohybu robotu.



Obr. 29 GUI - pro ovládání směru chůze kráčejícího čtyřnohého robotu

9.2.5 Komunikace Eyebotu s PC přez Bluetooth

Komunikace s PC po seriové lince může probíhat přes připojený kabel do seriového portu nebo přes Bluetooth modul – HandyPort (Obr. 9.2.3.1)

Obr. 9.2.5.1 HandyPort - Bluetoth modul pro Eyebot

V programu pro komunikaci na Eyebotu stačí pomocí malé změny nadefinovat, jestli bude komunikace probíhat přes SERIAL1, což je komunikace po sériovém kabe-

lu nebo SERIAL2 a komunikace bude probíhat přes Bluetooth modul. Pak už stačí jen nastavit rychlost komunikace.

Příklad nastavení:

```
//#define SERIALx SERIAL1       // seriova linka
#define SERIALx SERIAL2        //BlueTooth
#define SER_SPEED SER9600
```

První řádek je okomentovaný, tím komunikace bude probíhat přes SERIAL2, tedy přes bluetooth modul. Rychlost komunikace bude 9600kbps.

Na počítači bude komunikaci v programovém prostředí Matlab2008a, zprostředkovávat toolbox **XSERIAL** viz.[4].

Příklad nastavení komunikace:

```
ser = xserial ('Port','COM5','BaudRate', 9600);
```

Přes objekt **ser** bude probíhat komunikace. Port COM5 připojíme k Bluetooth zařízení v PC, pomocí programu Bluesoleil 3.2.2.8.

Závěr:

Detailně jsem se seznámil s vlastnostmi a programováním platformy Eyebot, prostřednictvím experimentů vykonaných s touto řídící jednotkou. Během těchto experimentů jsem vytvořil experimentální programy na obsluhu a ovládání periférií, kterými řídící jednotka Eyebot disponuje. Veškeré programy pro řídící jednotku Eyebot jsem vytvořil v jazice C.

Dále jsem vytvořil komunikační protokol řídící jednotky a počítače po seriové lince. Tento komunikační protokol lze použit k řízení po drátě nebo prostřednictvím bezdrátové technologie Bluetooth.

Za pomocí tohoto komunikačního protokolu jsem vytvořil program pro řízení chůze experimentálního čtyřnohého robotu „Jaromír“. Tento robot lze ovládat v programovém prostředí Matlab 2008a prostřednictvím grafického uživatelského rozhraní.

Použitelnost platformy je omezená. Dá se použít prakticky jen na školní příklady mobilních robotů. V technické praxi je však prakticky nepoužitelná z důvodů řady omezujících parametů, jako je například počet servomotorů a DC motorů, které lze připojit. Další nevýhodou je absence analogového výstupu.

Seznam použité literatury:

[1] Robotics & Automation Lab: veřejné webové stránky http://robotics.ee.uwa.edu.au

[2] Světlík, M.: Kinematické řízení a vizualizace virtuálního prototypu čtyřnohého kráčejícího robotu, VUT v Brne, 2006

[3] Dirk Louis, Petr Mejzlík, Miroslav Virius: Jazyky C a C++ podle normy ANSI/ISO; GRADA Publishing, 1999

[4] HUMUSOFT: REAL TIME TOOLBOX, for use with SIMULINK®; version 3.12

[5] Hlava J.:Prostředky automatického řízení II, ČVUT v Praha, 2000