ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ŘÍDICÍ TECHNIKY

BAKALÁŘSKÁ PRÁCE
Senzory pro malé mobilní roboty

Praha, 2008 Milan Navrátil

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jan F r a n c

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Korelační a koherentní analýza v EEG

Pokyny pro vypracování:

Cílem práce je podrobně se seznámit s EEG signálem, dále se základními metodami jeho analýzy v časové a kmitočtové oblasti. Student při své práci naváže na již existující řešení a bude realizovat extrakci specifických popisných atributů. Úseky EEG signálu budou podrobeny počítačové analýze se stanovením korelace mezi jednotlivými kanály, hodnot jednotlivých spekter (delta, theta, alfa, beta, gama) a výpočtem EEG koherencí (metody výpočtu koherence podle zapojení: interhemisférická a intrahemisférická koherence). Dále bude implementováno prostředí pro přehlednou vizualizaci vypočtených hodnot.

1. Seznamte se s problematikou měření a hodnocení EEG.
2. Navrhněte strukturu programu obsahující různé metody klasifikace.
3. Implementujte navržené řešení.

Seznam odborné literatury:

[1] Teofilo, L.; Lee-Chiong: Sleep: a Comprehensive Hanbook. John Wiley & Sons, Inc., Hoboken, New Jersey, 2006
[2] Schlesinger, M. I.; Hlaváč, V.: Ten Lectures ond Statistical andStructural Pattern Recognition. CTU, Prague, 1999

Vedoucí bakalářské práce: Ing. Václav Gerla

Platnost zadání: do konce zimního semestru 2008/2009

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická

v.z. doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 18.02.2008

Abstrakt

V bakalářské práci je popsáno rozšíření robota pro robotický fotbal o senzorickou část, která robotovi umožňuje autonomní pohyb po prostoru, aniž by narážel do překážek. To je realizováno pomocí senzoru vzdálenosti Sharp GP2D12, který pracuje v rozsahu 10 až 80cm pro detekovatelnost předmětu. Dále umožňuje robotovi při jízdě sledovat černou čáru nakreslenou na podlaze. To mu umožňuje trojice odrazových senzorů QRD1114. Robot byl převzat z dřívějšího projektu a rozšířen o hardwarovou část a o programové vybavení. V textu práce jsou popsány jednotlivé registry A/D převodníku mikrokontroléru H8S/2638. Součástí práce je návrh elektronického obvodu včetně kompletního návrhu senzorické desky plošných spojů, na kterém jsou senzory umístěny. Programová část této práce je psaná v programovacím jazyce C. Je vytvořen řídící program, který zadané úkoly pohybu robota realizuje.

Abstract

This thesis describes the extension of the robot by the sensoric part, so that it can be used for the robotic football. This part allows the autonomic movement of the robot, all around the place, without crashing in to the barriers. It is realized by a distance sensor Sharp GP2D12, which works in a range of 10 - 80 cm of seeing the article. It also allows the robot to fallow the black line drown on the floor. This part works by the tern of picture sensors QRD1114. The robot was taken from the last project and extended by the hardware part and the program accessories. In the text of the work are described the registers of A/D convertor in the microcontroler H8S/2638.The part of the work is also an application of the electronic circuit, including the complete application of the sensoric electronic circuit board, on which are these sensors placed. The program part of this work is written in the program language C. There is a control program made, which implements the movemet of the robot.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne 8.6.2008

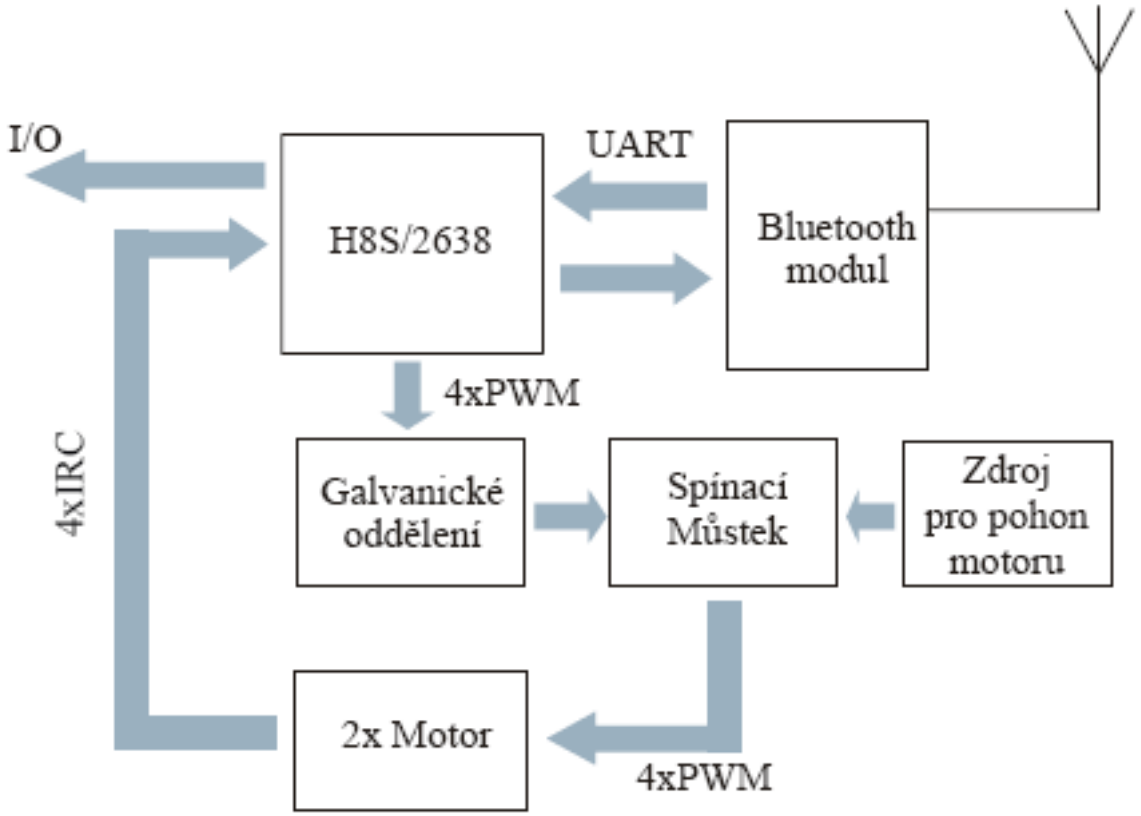
.....
Milan Navrátil

Především bych chtěl poděkovat Ing. Michalu Sojkovi,
za jeho připomínky a rady, které mi vždy otevřely tu správnou cestu.
Stejně tak Marku Pecovi za cenné rady při realizaci této práce.
Dále si velké díky zaslouží má přítelkyně a rodina, za morální podporu při studiu na této škole.

Obsah

1 Úvod

V roce 2006 realizoval Petr Kovačik jako svou diplomovou práci tohoto fotbalového robota, podle pravidel mezinárodní organizace FIRA. Robot je řízen mikrokontrolérem H8S/2638 od firmy Renesas, který pracuje na kmitočtu do 20MHz. Celkové obvodové řešení navrženého robota je vyjádřeno blokovým schématem na obrázku 1-1.



Obrázek 1-1: Původní blokové schéma robota

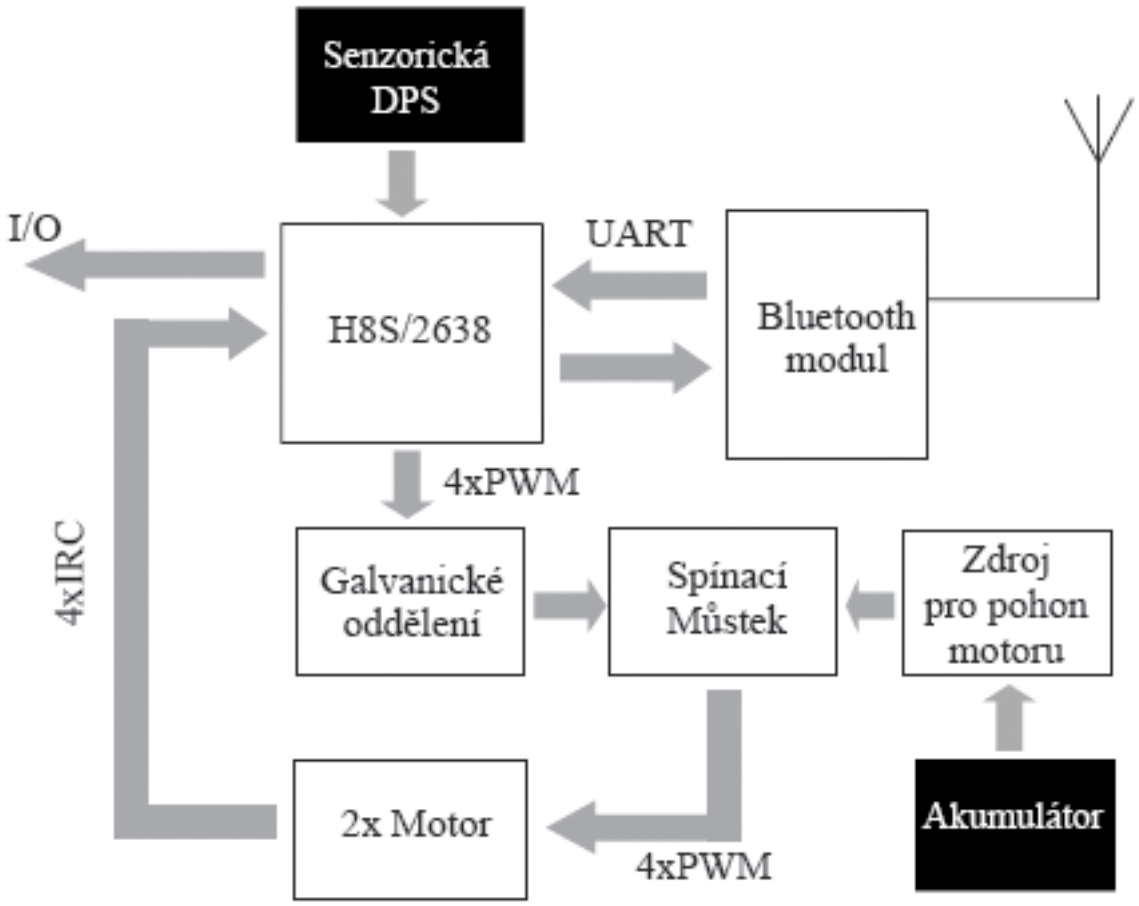
Pohyb robota je zajišťován prostřednictvím dvou stejnosměrných motorů, kde se každý nezávisle stará o jedno kolečko. Díky tomu je možný libovolný pohyb robota. Pro napájení motoru je použit signál PWM. V ose motoru je pak umístěn IRC snímač , který snímá aktuální rychlost motoru. Pro komunikaci s okolím je robot vybaven bluetooth modulem. Více informací lze nalézt v [1].

Úkolem mé bakalářské práce bylo rozšíření stávajícího stavu robota o dva úkoly. První byl pohyb robota po prostoru, aniž by narážel do překážek. Druhým úkolem byl pohyb robota po černé čáře nakreslené na podlaze.

První úkol je realizován pomocí senzoru vzdálenosti Sharp GP2D12. Jeho návrh je popsán v kapitole 2.3. Druhý úkol je realizován pomocí trojice odrazových senzorů QRD1114, které jsou popsány v kapitole 2.6.

Dále bylo potřeba vybrat a zakoupit akumulátor pro napájení robota. Výběr je popsán v kapitole 2.1.

Pro realizaci těchto úkolů bylo nutné nejdříve navrhnou senzorickou desku plošných spojů (DPS) pomocí návrhového programu OrCAD [3]. K pochopení ovládání programu mi pomohl volitelný předmět X34PPN vypsaný katedrou mikroelektroniky a vyučovaný Ing. Vítem Záhlavou, CSc. Po přidání senzorické DPS a akumulátoru, se blokové schéma robota rozšířilo podle obrázku 1-2.

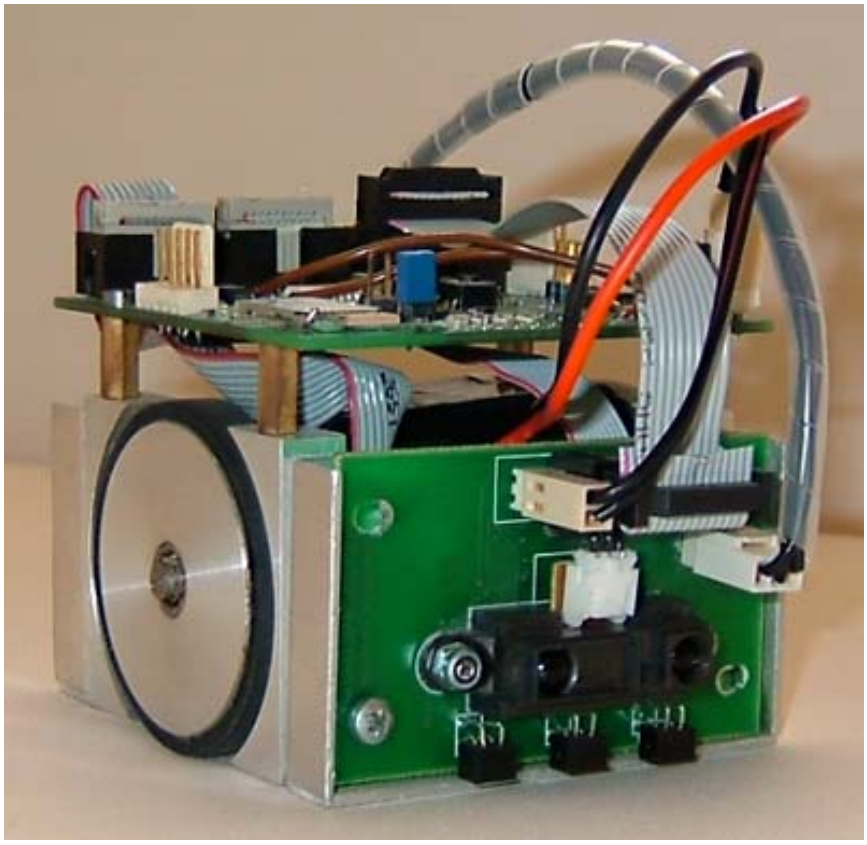


Obrázek 1-2: Rozšířené blokové schéma robota o senzorickou DPS a Akumulátor

Po samotné realizaci senzorové DPS, bylo potřeba desku otestovat a naprogramovat mikrokontrolér. Konkrétně převod analogového výstupu ze senzorů na digitální signál.

K tomu byl použit A/D převodník v mikrokontroléru. Potřebné nastavení registrů A/D převodníku popisuje kapitola 4.

Výsledná podoba robota i s osazenou senzorickou DPS je vidět na obrázku 1-3.



Obrázek 1-3: Testovací zapojeni robota po rozšíření

2 Obvodové řešení

Tato kapitola se zabývá hardwarovým návrhem senzorické desky. Nejprve je popsán akumulátor pro napájení robota. Následující sekce jsou věnovány jednotlivým částem senzorické desky. Konkrétně senzorům vzdálenosti GP2D12, senzorům odrazivosti QRD1114, výpočtu filtru typu dolní propust, operačnímu zesilovači TLC272 a stabilizátoru napětí KA7805.

2.1 Napájení robota

V návrhu robota je použito několik hodnot napětí. Základní vstupní napětí je poskytováno akumulátorem. Je použito k napájení dvojice motorů a k odvození dalších dvou hodnot napětí (5V pro mikrokontrolér, 3,3V pro Bluetooth modul). Potřebná hodnota vstupního napětí je kolem 10V.

Pro napájení mobilního robota jsme potřebovali navrhnout akumulátor. Při návrhu akumulátoru jsme byli nejvíce limitováni jeho velikostí, protože se musel vejít do těla robota. Tedy jeho maximální rozměr byl 52x50x25. Což jsme byli schopni splnit pouze použitím článku velikosti AAA.

Použité články mají napětí 1,2V. Pro získání požadovaného napětí kolem 10V jsme použili 9 článků, tedy 10,8V. Jejich uspořádání je patrné z obrázku 2-1.

Vhodnost chemického složení článku jsme rozhodovali mezi akumulátory Ni-Cd, Ni-MH a Li-Po. Jejich vzájemné porovnání je uvedeno v tabulce 2-1 [11], [12].

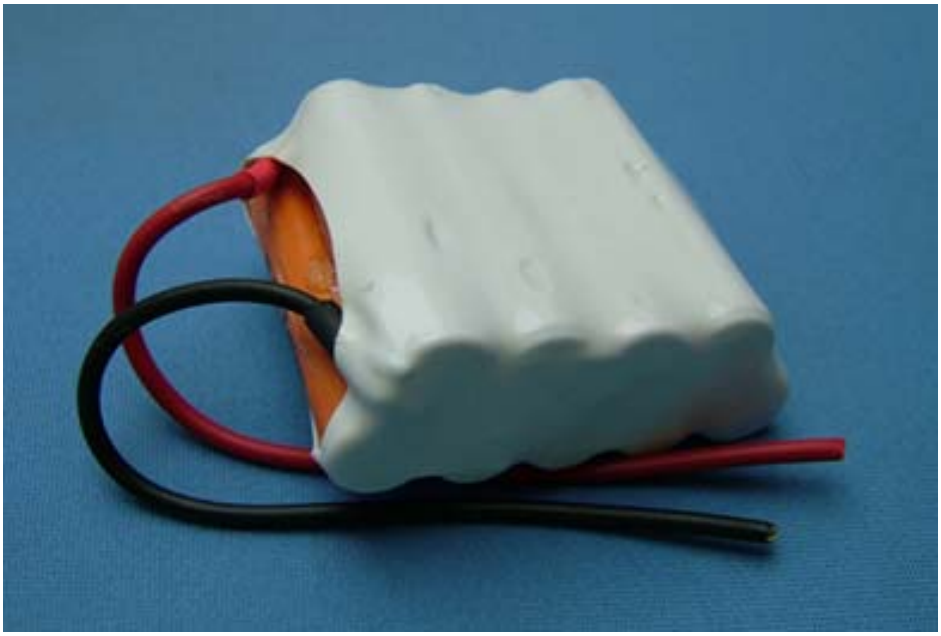
	Ni-Cd	Ni-MH	Li-Po
Energetická hustota (Wh/kg)	45-80	60-120	90-120
Životnost (počet cyklů)	1500	300-500	1000
Rychlost nabíjení (h)	1	2 až 4	< 1
Samovolné vybíjení za měsíc (%)	20	30	< 10
Napětí jednoho článku (V)	1,2	1,2	3,7
Cena (kč/V) pro 1200 mAh	55	33	162

Tabulka 2-1: Porovnání vlastností jednotlivých článků

Kvůli vysoké ceně jsme z výběru vyřadili Li-Po články. Dále pak kvůli nedostupnosti Ni-Cd článků o velikosti AAA byly vybrány Ni-MH články o kapacitě 650mAh.

Články nám poskládali do konečné podoby v brněnské firmě Fulgurbattman (http://www.fulgurbattman.cz/).

Společně s baterkami jsme objednali nabíječku Ansmann ACS 410M. Průběh nabíjení je zde řízen mikroprocesorem a nabíječka je určena pro akumulátorové sestavy od 4 do 10 článků Ni-MH (Ni-Cd), tedy 4,8V až 12V.



Obrázek 2-1: Akumulátorový článek

2.2 Senzorická deska

Pro připojení dálkových a odrazových senzorů ke stávající řídící desce, byla navržena senzorická DPS, na kterou byl umístěn také stabilizátor napětí z 10V na 5V, filtr typu dolní propust druhého řádu pro odstranění rušení signálu získaného ze senzoru vzdálenosti a konektor pro připojení sensorické desky k desce řídící.

Velikost senzorické DPS byla přizpůsobena šířce robota, z důvodu vhodného připevnění k přední části robota. Rozměr senzorické DPS je tedy 68x40mm. Schéma zapojení a návrh plošného spoje je v příloze A.

Součástky jsou montovány SMT technologií na spodní stranu desky. Na horní straně se nacházejí jen senzory a konektor pro napájení a na propojení s řídící deskou. Samotné osazení součástek jsem provedl v dílně katedry řídící techniky.

2.3 Senzor vzdálenosti SHARP GP2D12

Aby se mohl robot samovolně pohybovat po prostoru a nenarážel do překážek, musely být použity senzory pro měření vzdálenosti robota od překážek. Z velkého množství dostupných senzorů na trhu nám nejlépe vyšla řada GP2 od výrobce SHARP. Na výběr jsme měli senzory s rozsahem vzdáleností uvedených v tabulce 2-2.

Typ	Výstup	Minimální vzdálenost (cm)	Maximální vzdálenost (cm)
GP2D02	Sériový	10	80
GP2D05	Digitální	-	konstantní 24
GP2D12	Analogový	10	80
GP2D15	Digitální	-	konstantní 24
GP2D120	Analogový	4	30
GP2Y0A02YK	Analogový	20	150
GP2Y0D02YK	Digitální	-	konstantní 80

Tabulka 2-2: Přehled dálkových senzorů řady GP2 od výrobce SHARP

Jak již bylo popsáno výše, tak celkový rozměr těla robota je 72x79x72 (ŠxDxV). Pro nejlepší orientaci v prostoru, vztaženou k velikosti robota, byl vybrán rozsah rozlišitelnosti senzoru 10-80 cm. Dále bylo potřeba pro komunikaci s řídící deskou a tedy s mikrokontrolérem použít senzor s analogovým výstupem. Těmto podmínkám odpovídal senzor GP2D12.

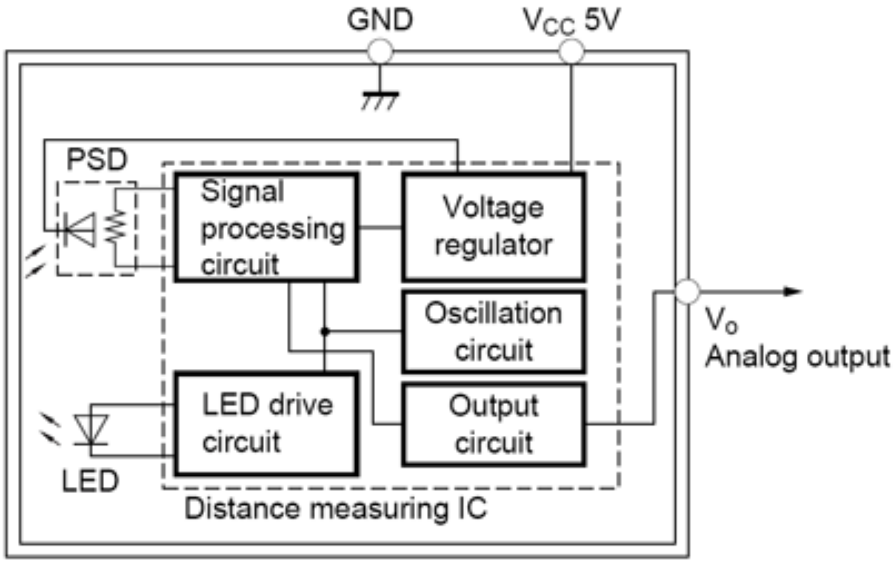
Byly použity dva senzory vzdálenosti. Jeden umístěný směrem dopředu, druhý směrem dozadu. To z důvodu možného pohybu robota oběma směry.

2.3.1 Technické specifikace senzoru GP2D12

Sharp GP2D12 je analogový senzor vzdálenosti, který vyzařuje infračervený paprsek o vlnové délce 850 nm. Pokud je nějaký předmět vzdálen v rozsahu detekovatelnosti senzorem, pak odražený infračervený paprsek tvoří obraz v lineárním CCD poli přijímače.

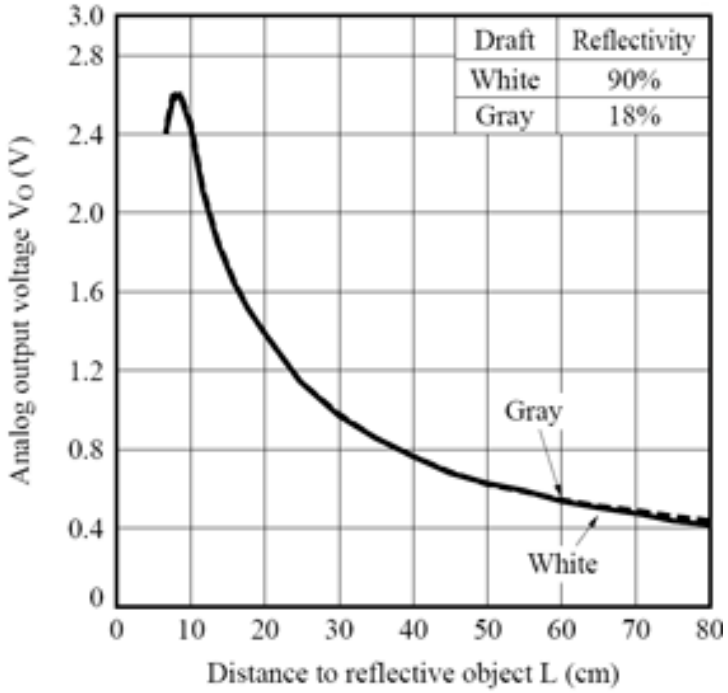
Detektor je celkem necitlivý k okolnímu osvětlení, díky čemuž je možné detekovat tmavé předměty v plném slunečním světle.

Vnitřní struktura senzoru je na obrázku 2-2.



Obrázek 2-2: Vnitřní struktura senzoru Sharp GP2D12

Senzor může spojitě měřit vzdálenost od objektu v použitelném rozsahu 10 až 80 cm. Výstupem ze senzoru je analogové napětí, které je funkcí vzdálenosti od překážky podle obrázku 2-3.



Obrázek 2-3: Graf funkce výstupního napětí / vzdálenosti od překážky

2.3.2 Měření na senzoru GP2D12

Na hotové senzorické DPS jsme provedli konečné měření funkce výstupního napětí na vzdálenosti. Změřené hodnoty jsou uvedeny v tabulce 2-3.

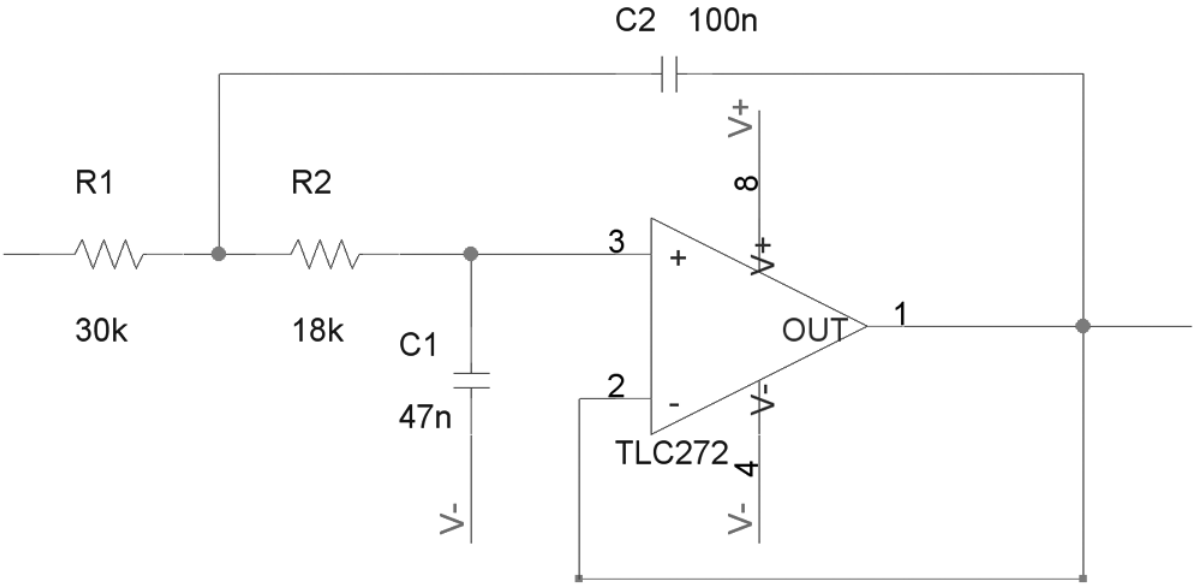
Vzdálenost překážky [cm]	6	7	8	9	10	11	12	13	14
U [V] (Přední senzor)	2,42	2,74	2,62	2,52	2,26	2,14	2,02	1,88	1,76
U [V] (Zadní senzor)	2,58	2,76	2,62	2,52	2,34	2,18	2,02	1,88	1,76
Vzdálenost překážky [cm]	15	16	17	18	19	20	25	30	35
U [V] (Přední senzor)	1,64	1,58	1,52	1,44	1,4	1,34	1,1	0,94	0,78
U [V] (Zadní senzor)	1,64	1,58	1,52	1,44	1,4	1,32	1,08	0,94	0,78
Vzdálenost překážky [cm]	40	45	50	55	60	65	70	75	80
U [V] (Přední senzor)	0,68	0,62	0,56	0,52	0,48	0,44	0,42	0,4	0,38
U [V] (Zadní senzor)	0,68	0,64	0,56	0,52	0,48	0,44	0,42	0,4	0,38

Tabulka 2-3: Závislost výstupního napětí na vzdálenosti od překážky

Porovnáním hodnot z obrázku 2-3 a z tabulky 2-3 je vidět, že naše změřené hodnoty odpovídají těm katalogovým.

2.4 Dolnopropustný filtr

Z výstupu ze senzoru vzdálenosti Sharp GP2D12 jsme dostali analogový signál zašuměný náhodnými vysokofrekvenčními skoky. Pro jejich odstranění byl navržen filtr typu dolní propust, který je zobrazen na obrázku 2-4.



Obrázek 2-4: Filtr dolní propust 2. řádu

Byla zvolena dolní propust 2. řádu se zlomovou frekvencí f = 100 Hz, což odpovídá kruhové frekvenci $\omega_0 = 2\pi * f = 628{,}32$ rad/s.

Dále byly zvoleny hodnoty rezistorů $R_1 = 30\text{k}\Omega$ a $R_2 = 18\text{k}\Omega$. Vztahy (2.1) až (2.4) jsou převzaty z [2].

Výpočet C1 a C2:

$$\frac{u_o}{u_i} = \frac{k \cdot \omega_p^2}{p^2 + \alpha_p \cdot \omega_p + \omega_p^2} \tag{2.1}$$

$$\omega_p^2 = \frac{1}{R_1 \cdot R_2 \cdot C_1 \cdot C_2} \tag{2.2}$$

$$\alpha_p = 2 \cdot \sqrt{\frac{C_1}{C_2}} \tag{2.3}$$

Porovnáním s nenormovanou funkcí:

$$\frac{u_o}{u_i} = \frac{b_k \cdot \omega_0^2}{p^2 + a_k \cdot \omega_0 \cdot p + b_k \cdot \omega_0^2} \tag{2.4}$$

Použitím koeficientů Butterworthovy aproximace $b_1 = 1$ a $a_1 = 1{,}414$ dostaneme:

$$\omega_p^2 = b_k \cdot \omega_0^2 \rightarrow \frac{1}{R_1 \cdot R_2 \cdot C_1 \cdot C_2} = 1 \cdot \omega_0^2$$

$$\alpha_p \cdot \omega_p = a_k \cdot \omega_0 \rightarrow 2 \cdot \sqrt{\frac{C_1}{C_2}} \cdot \frac{1}{\sqrt{R_1 \cdot R_2 \cdot C_1 \cdot C_2}} = a_k \cdot \omega_0$$

Odsud dostaneme hledané C1 a C2:

$$C_1 = \frac{a_k}{2 \cdot \sqrt{R_1 \cdot R_2} \cdot b_k \cdot \omega_0}$$

$$C_2 = \frac{2}{\sqrt{R_1 \cdot R_2} \cdot a_k \cdot \omega_0}$$

Po číselném dosazení dostáváme C1 = 48nF a C2 = 98nF. Jelikož nám nejde přesně o zlomovou frekvenci 100Hz volíme kondenzátory podle výrobní řady a konečná zvolená hodnota je C1 = 47nF a C2 = 100nF.

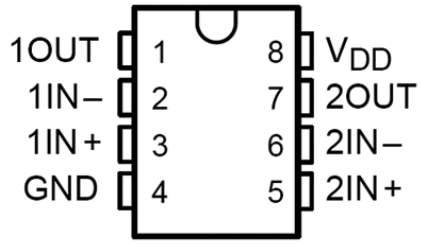
u_o ... výstupní napětí [V]

u_i ... vstupní napětí [V]

α_p ... zvlnění [dB]

2.5 Operační zesilovač TLC272

Operační zesilovač TLC272 byl vybrán pro jeho rail-to-rail architekturu. Jedno SMD pouzdro obsahuje dva zesilovače. Funkce nožiček pro zapojení je vidět na obrázku 2-5 [7].



Obrázek 2-5: Pouzdro operačního zesilovače

2.6 Senzor odrazivosti QRD1114

Další úkol byl autonomní pohyb robota po černé čáře nakreslené na světlé podložce. Pro jeho realizaci byly použity tři odrazové senzory QRD1114. Jejich počet byl vybrán záměrně, jelikož je díky nim možné určit přesnou polohu robota vzhledem k čáře. Základem je prostřední senzor, který primárně snímá černou čáru . Pokud čára změní směr, prostřední senzor již nebude snímat její povrch, do činnosti se zapojí krajní senzory. Podle toho, který z nich detekuje čáru, se bude zatáčet do té doby, než čáru opět detekuje prostřední senzor.

2.6.1 Technické specifikace senzoru QRD1114

QRD1114 je odrazový senzor skládající se z infračervené, svítící diody a křemíkového NPN foto tranzistoru. Ty jsou umístěny těsně vedle sebe v černém plastovém pouzdře.

Světlo z diody se odráží od reflexivní podložky a je zachyceno foto tranzistorem. Foto tranzistor reaguje na odražené světlo vydávané z diody pouze když reflexivní povrch je vzdálený v rozmezí 1-5 mm. Naměřené hodnoty napětí na výstupu senzoru podle povrchu jsou uvedeny v tabulce 2-4 [5]. Při pohledu zpředu je QRD1114-1 levý senzor, QRD1114-2 pravý senzor a QRD114-3 prostřední senzor.

	U [V]		
Povrch podložky	QRD1114-1	QRD1114-2	QRD1114-3
Bílá	0,96	0,96	0,98
Černá	3,52	3,52	3,54

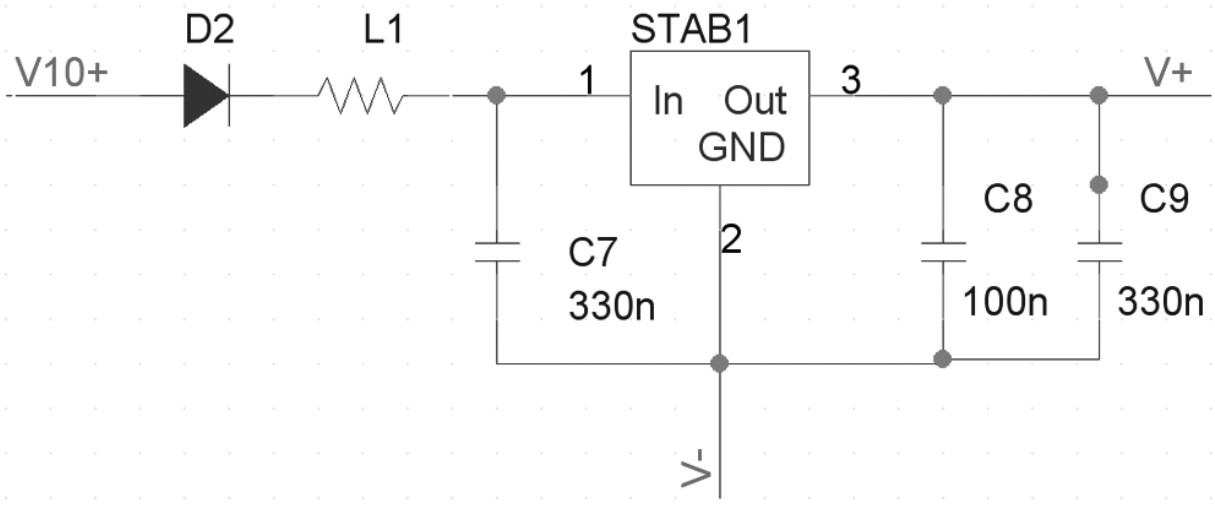
Tabulka 2-4: Napětí na výstupu senzoru QRD1114 podle barvy podložky

2.7 Stabilizátor napětí KA7805

Z konstrukčních důvodů jsme chtěli oddělit elektrickou část řídící desky od desky senzorické. To z důvodu, že odběr motorů by mohl způsobit napěťové poklesy na napájení senzorů a tím silně ovlivnit přesnost dat získaných ze senzorů a zhoršit kvalitu řízení. Proto senzorickou desku napájíme přímo z baterií a samostatným stabilizátorem dostáváme požadované napětí 5V pro napájení jednotlivých senzorů.

Použit byl stabilizátor KA7805 v pouzdře D-PAK pro SMT montáž o rozměru 6,6x9,5x2,3mm. Jeho maximální vstupní napětí je 35V. Usměrňuje na 5V, kde výchylka je maximálně ± 0,2V. Provozní teplota je 0-125°C [6]. Pro naše potřeby nebylo potřeba přidávat chladič.

Z katalogového listu stabilizátoru jsme použili doporučené zapojení doplněné o tlumivku a ochrannou diodu, dle obrázku 2-6.



Obrázek 2-6: Zapojení stabilizátoru napětí

2.8 Vyřešené problémy

Při měření analogového napětí na senzoru vzdálenosti, bylo po připojení akumulátoru k řídící a senzorové DPS zjištěno silné zkreslení signálu. To bylo způsobeno kondenzátory C15 a C16 proti zemi, na vstupech A/D převodníku mikrokontroléru umístěného na řídící DPS, kterou navrhoval Petr Kovačik [1]. Po odpájení těchto kondenzátorů byl signál bez zkreslení.

Z konstrukčních důvodů se nevešel 20-pinový konektor J3. Pro naše potřeby byly důležité piny 10-14 a 19, 20, proto bylo možné použít jen 14-pinový konektor. Ten byl umístěn od pinu číslo 7.

Na aktuální senzorové DPS je pozměněno napájení. Původně mělo být napájení přivedeno z řídící DPS pomocí konektoru J3. Ale po proměření bylo zjištěno, že na pinu 20 není předpokládané napětí 10,8V, ale jen 5V. Proto byl na senzorickou desku, na piny 1 a 2 konektoru J3, připájen samostatný konektor pro přímé napájení z baterie.

Při testování jsme dále zjistili, že spodní hrana senzorů vzdálenosti GP2D12 musí být při montování, na DPS a zadní stranu těla robota, o 2 mm odchýlena od podložky, jinak je senzor namířen směrem k zemi a tedy se zmenší jeho rozsah detekovatelnosti překážky na vzdálenost 10-60cm.

3 Programové řešení

V této kapitole jsou popsány jednotlivé registry A/D převodníku a jejich potřebné nastavení [9], tak jak je používám v programu.

Měřitelným výstupem ze senzorické desky popsané v kapitole 2 je analogový signál, který je dále zpracováván v mikrokontroléru umístěném na hlavní řídící desce. Pro převod získaného analogového signálu ze senzorů na signál digitální je použit A/D převodník mikrokontroléru. Měřené hodnoty z výstupu senzorů jsou uvedeny v kapitole 3.6.

V našem případě používáme vstupy mikrokontroléru 109(AN6), 110(AN7), 111(AN8), 112(AN9) a 113(AN10).

Řídící program je napsán v programovacím jazyce C [8]. K vývoji byl použit operační systém Linux, distribuce Ubuntu verze 7.10 [10].

3.1 A/D převodník

Vlastnosti:

- 10-bitové rozlišení
- 12 vstupů
- lze nastavit rozlišení převodníku podle napěťového rozsahu
- vysoká rychlost převodu, 13,3μs na kanál (při 20 MHz)
- čtyři 16-bitové datové registry pro uchování naměřených hodnot
- možnost generování přerušení po dokončení převodu (ADI)
- na výběr jsou dva módy: single mode a scan mode

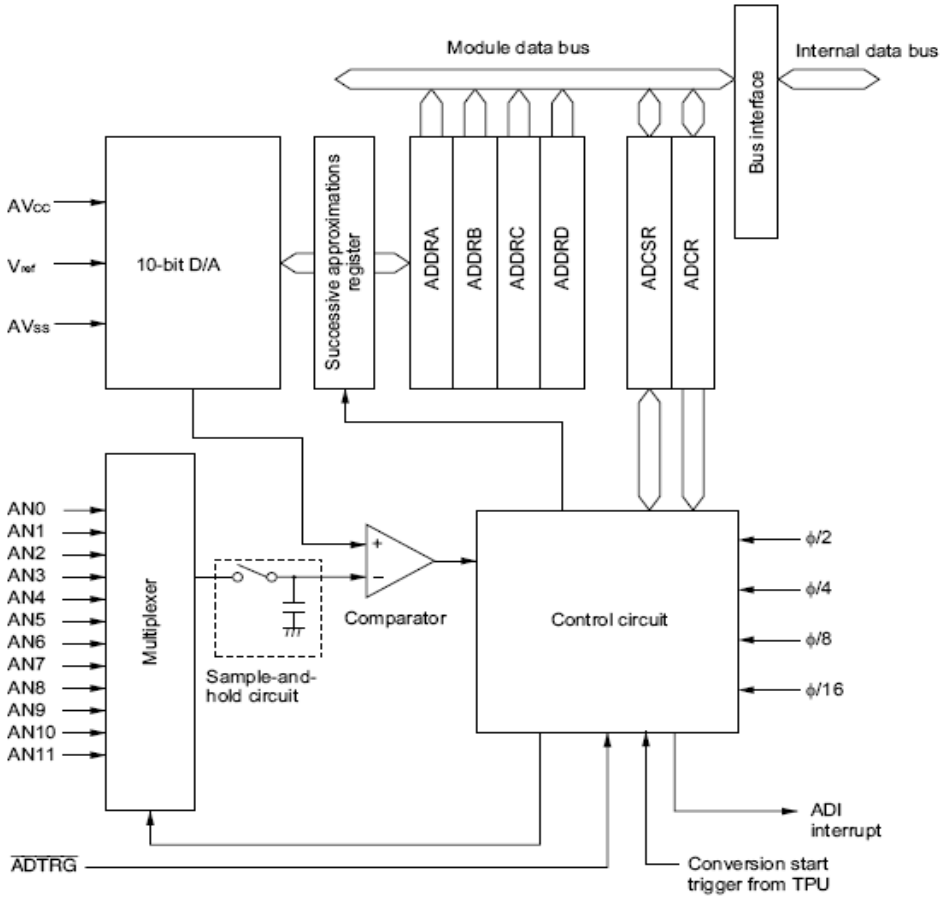
Pro nastavení potřebných vstupů do A/D převodníku je nutné zvolit příslušné kanály. Celkově 12 použitelných vstupů je rozděleno do dvou kanálů a dvou skupin. Kanál nastavený na 0 je pro vstupy AN0 až A7, nastavený na 1 je pro vstupy A8 až A11. Další dělení podle skupin je AN0 až AN3 a AN8 až A11 při nastavení skupiny na 0. Vstupy AN4 až AN7 při nastavení skupiny na 1. Jejich rozdělení je uvedeno v tabulce 3-1.

Název vstupu	Symbol	Kanál	Skupina
Analogový vstup 0	AN0	0	0
Analogový vstup 1	AN1	0	0
Analogový vstup 2	AN2	0	0
Analogový vstup 3	AN3	0	0

Analogový vstup 4	AN4	0	1
Analogový vstup 5	AN5	0	1
Analogový vstup 6	AN6	0	1
Analogový vstup 7	AN7	0	1
Analogový vstup 8	AN8	1	0
Analogový vstup 9	AN9	1	0
Analogový vstup 10	AN10	1	0
Analogový vstup 11	AN11	1	0

Tabulka 3-1: Nastavení jednotlivých vstupů A/D převodníku

Blokový diagram A/D převodníku je zobrazen na obrázku 3-1.



Legenda:

ADCR: A/D control register

ADCSR: A/D control/status register

ADDRA: A/D data register A

ADDRB: A/D data register B

ADDRC: A/D data register C

ADDRD: A/D data register D

Obrázek 3-1: Blokový diagram A/D převodníku (H8S/2638)

3.1.1 Data Register ADDR (ADDRA - ADDRD)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	-	-	-	-	-	-
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

A/D převodník má čtyři 16-bitové datové registry ADDRA až ADDRD. 10-bitový výsledek z A/D převodníku je podle zvoleného kanálu a skupiny uložen do příslušného registru. Horních 8-bitů je uloženo do horního bytu (bity 8 až 15) registru ADDR, dolní 2-bity jsou uloženy do dolního bytu (bity 6 a 7). Bity 0 až 5 jsou vždy 0.

Souvislosti mezi analogovými vstupy a datovými registry ADDR je v tabulce 3-2.

Kanál 0 (CH3 = 0)		Kanál 1 (CH3 = 1)		
Skupina 0	Skupina 1	Skupina 0	Skupina 1	Data registr
AN0	AN4	AN8	-	ADDRA
AN1	AN5	AN9	-	ADDRB
AN2	AN6	AN10	-	ADDRC
AN3	AN7	AN11	-	ADDRD

Tabulka 3-2: Analogové vstupy a souvislost s datovými registry

CPU může stále číst datové registry. Horní byte je čten přímo. Jelikož je datová sběrnice mezi A/D a CPU 8-bitová, přesun dat z dolního bytu je proveden přes pomocný registr (TEMP). Tedy současně se čte horní i dolní byte, ale jen horní byte je odeslán přes sběrnici do CPU a dolní byte je uložen do TEMP registru. V druhém cyklu je z TEMP registru odeslán dolní byte do CPU.

3.1.2 Control/Status Register (ADCSR)

7	6	5	4	3	2	1	0
ADF	ADIE	ADST	SCAN	CH3	CH2	CH1	CH0
0	0	0	0	0	0	0	0
R/(W)*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

* Pouze 0 může být zapsána do ADF na vymazání tohoto příznaku

ADCSR je 8-bitový R/W registr, který řídí A/D převodní operace.

Bit 7 – A/D konec příznaku (*A/D End Flag*)

ADF: Stavový příznak signalizující konec A/D převodu

ADF	Popis
0	[Mazací podmínka] (Počáteční hodnota) 0 je zapsána do ADF po přečtení ADF = 1 Když DTC (Data Transfer Controller) je aktivován ADI přerušením a ADDR je čtená
1	[Nastavovací podmínka] Single mode: Když skončil A/D převod Scan mode: Když skončil A/D převod na všech předepsaných kanálech

Bit 6 – A/D povolení přerušení (*A/D Interrupt Enable*)

ADIE: Povoluje/zakazuje dotaz na přerušení po skončení A/D převodu

ADIE	Popis
0	Zakazuje přerušení při skončení A/D převodu (Počáteční hodnota)
1	Povoluje přerušení při skončení A/D převodu

Bit 5 – A/D Start

ADST: Spouští/zastavuje A/D převod. Drží hodnotu v 1 po dobu A/D převodu. Může být nastaven do 1 softwarově, nastaven časovačem převodu, nebo externím vstupem $\overline{ADTRG}$.

ADST	Popis
0	Zastavený A/D převod (Počáteční hodnota)
1	Single mode: A/D převod je spuštěn. Vynulován do 0, když skončí převod na předepsaných kanálech Scan mode: A/D převod je spuštěn. Převod pokračuje po vybraných kanálech dokud není softwarově zastaveno, resetováno, nebo přejde do stand by módu.

Bit 4 – Scan Mode

SCAN: Na výběr je Single mode a Scan mode.

SCAN se smí nasnovat pouze, když ADST = 0.

SCAN	Popis	
0	Single mode	(Počáteční hodnota)
1	Scan mode	

Bit 3 – Výběr kanálu 3 (*Channel Select 3*)

CH3: Přepíná mezi kanály analogových vstupů.

Může být nastaven pouze když ADST = 0.

CH3	Popis
0	Přepnuto do kanálu 0, tedy dostupné jsou vstupy AN0 až AN7 (Poč.hod)
1	Přepnuto do kanálu 1. tedy dostupné jsou analogové vstupy AN8 až AN11

Bity 0 až 2 – Výběr kanálu 0 až 2 (Channel Select 0 to 2)

Společně se SCAN bitem definují, který analogový vstup bude vybrán.

Mohou být nastaveny pouze když ADST = 0.

ChannelSelection				Popis	
CH3	CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	0	AN0 (Počáteční)	AN0
			1	AN1	AN0,AN1
		1	0	AN2	AN0 až AN2
			1	AN3	AN0 až AN3
	1	0	0	AN4	AN4
			1	AN5	AN4, AN5
		1	0	AN6	AN4 až AN6
			1	AN7	AN4 až AN7

1	0	0	0	AN8	AN8
			1	AN9	AN8, AN9
		1	0	AN10	AN8-AN10
			1	AN11	AN8-AN11
	1	0	0	-	-
			1	-	-
		1	0	-	-
			1	-	-

3.1.3 Control Register (ADCR)

7	6	5	4	3	2	1	0
TRGS1	TRGS0	-	-	CKS1	CKS0	-	-
0	0	1	1	0	0	1	1
R/W	R/W	-	-	R/W	R/W	-	-

ADCR je 8-bitový R/W registr, který povoluje/zakazuje externí spouštění a nastavuje čas A/D převodu.

Bity 6 a 7 **Nastavení spouštěcího časovače 0 a 1 (*Timer Trigger Select*)**

TRGS0, TRGS1: Nastavení povoluje/zakazuje spuštění A/D převodu spouštěcím signálem (softwarově).

TRGS0 a TRGS1 mohou být nastaveny pouze, když ADST = 0.

Bit 7	**Bit 6**	
TRGS1	**TRGS0**	**Popis**
0	0	Softwarové spuštění A/D převodu povoleno (Počáteční nastav.)
	1	Spuštění A/D převodu pomocí TPU povoleno
1	0	-
	1	Spuštění A/D převodu externím $\overline{ADTRG}$ povoleno

Bit 5, 4, 1 a 0 **Chráněný:** Tyto bity jsou chráněny. Jsou čteny jako 1 a nelze je změnit.

Bit 3 a 2 **Nastavení hodin 1 a 2 (*Clock Select*)**

CKS1, CKS0: Tyto bity nastavují čas A/D převodu.

Lze ho nastavit pouze, když ADST = 0.

Bit 3	Bit 2		
CKS1	**CKS0**	**Popis**	
0	0	Čas převodu = 530 stavů (max.)	(Počáteční nastavení)
	1	Čas převodu = 266 stavů (max.)	
1	0	Čas převodu = 134 stavů (max.)	
	1	Čas převodu = 68 stavů (max.	

3.1.4 Module Stop Control Register A (MSTPCRA)

7	6	5	4	3	2	1	0
MSTPA7	MSTPA6	MSTPA5	MSTPA4	MSTPA3	MSTPA2	MSTPA1	MSTPA0
0	0	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MSTPCR je 8-bitový R/W registr, který spouští samotnou činnost A/D převodníku. V počátečním režimu je vypnut, aby se šetřila energie v bateriích. Pokud chceme A/D převodník používat, tak musíme nastavit bit 1 do 0.

Bit 1 – Module Stop

MSTPA1: Určuje zastavovací mód A/D převodníku.

Bit 1	
MSTPA1	Popis
0	Vynulování stop módu A/D převodníku – tímto bitem zapnu A/D
1	Nastavení stop módu A/D převodníku (Počáteční nastavení)

3.2 Činnost A/D převodníku

A/D převodník pracuje s postupnou 10-bitovou aproximací. Má dva pracovní režimy: single mode a scan mode.

3.2.1 Single Mode (SCAN = 0)

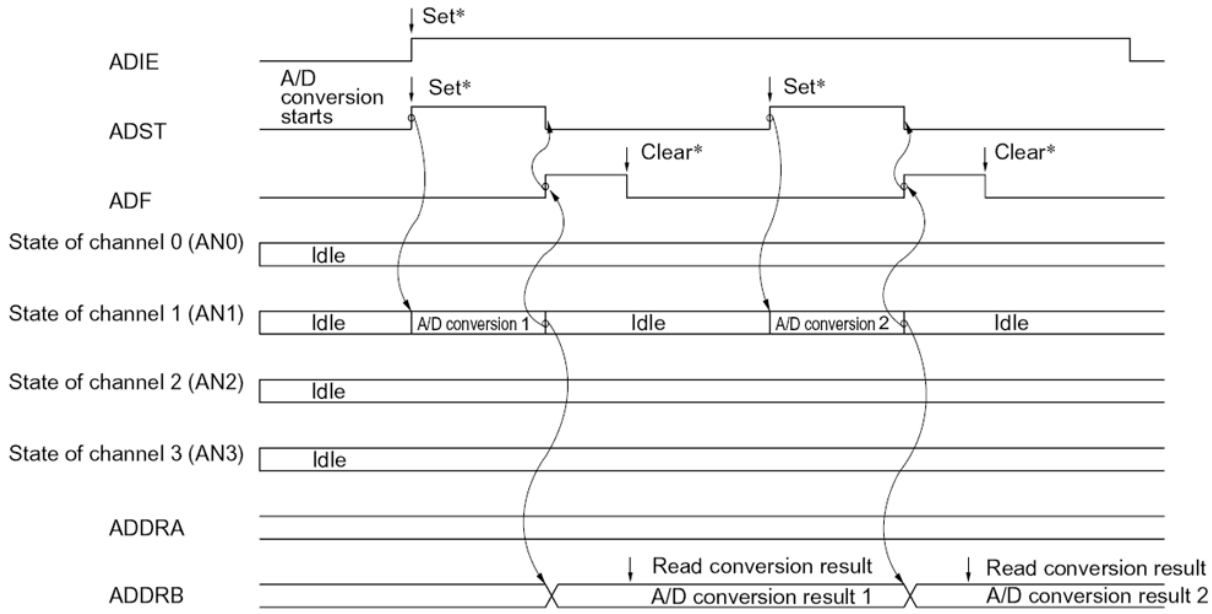
Tento mód je nastaven, pokud chceme získávat výsledek A/D převodu jen z jednoho zvoleného vstupu. A/D převod se spustí nastavením bitu ADST do 1. ADST zůstává na hodnotě 1 celou dobu A/D převodu a po skončení je nastaven na 0.

Po kompletním A/D převodu je nastaven příznak ADF do 1. Pokud je i bit ADIE nastaven do 1, je generován požadavek na přerušení. Po přečtení převodu z datového registru se ADF softwarově nuluje.

Pokud chceme změnit operační mód nebo vstup, musíme nejdříve vypnout A/D převod nastavením ADST do 0. Po provedení nezbytné změny se nastaví ADST do 1 a převod se opět spustí. Příklad A/D převodu v single módu je vidět na obrázku 3-2.

Příklad postupu, kde je nastaven vstup 1 (AN1)

1. Nastavení single módu (SCAN = 0), výběr vstupu AN1 (CH3 = 0, CH2 = 0, CH1 = 0, CH0 = 1) je popsán v kapitole 3.1.2., přerušení povoleno (ADIE = 1), spuštění A/D převodu (ADST = 1)
2. Po dokončení A/D převodu je výsledek uložen v datovém registru (ADDRB). Ve stejný okamžik je nastaven ADF do 1 a ADST je nastaven do 0.
3. V okamžiku, kdy se příznak ADF nastavil do 1, byl spuštěn dotaz na přerušení.
4. Obsluha přerušení je spuštěna
5. čtení ADCSR.
6. Čte se a zpracovává výsledek A/D převodu z ADDRB.
7. Pokud je po návratu z přerušení nastaven bit ADST do 1, A/D převod je znovu spuštěn a opakují se kroky 2 až 7.



Poznámka: * kolmé šipky označují instrukce provedeny softwarově

Obrázek 3-2: Příklad A/D převodu v single módu a vybraném kanálu AN1

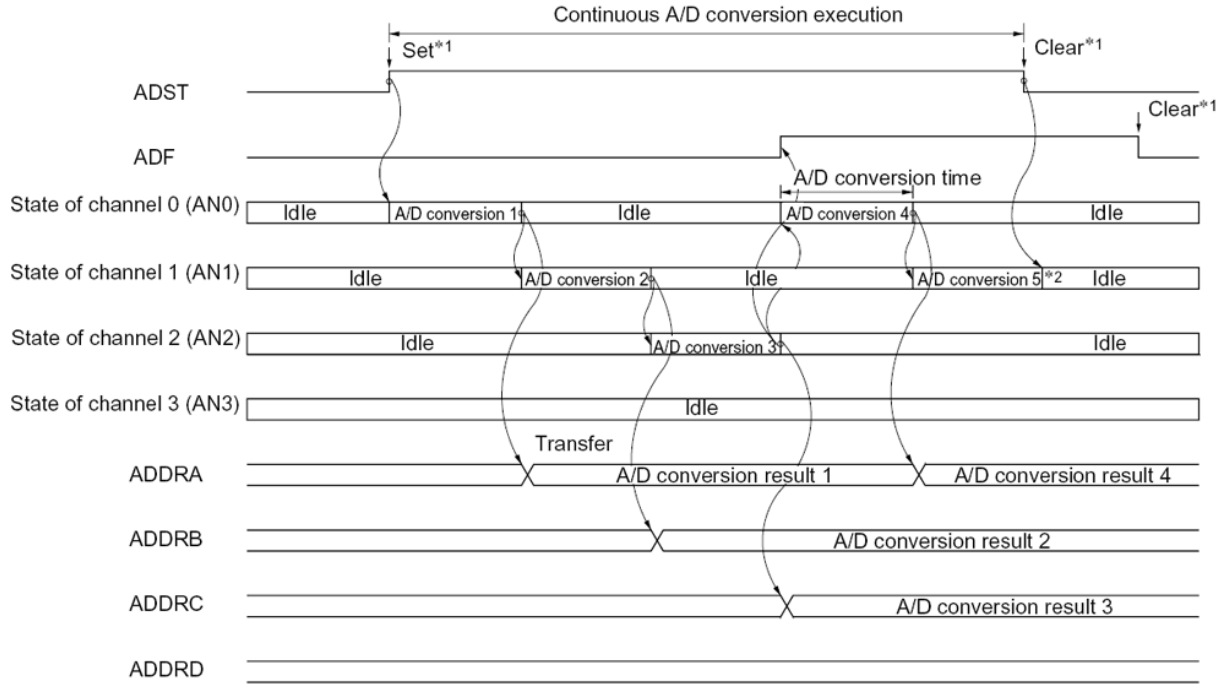
3.2.2 Scan Mode (SCAN=1)

Tento mód je použitelný pro monitorování analogových vstupů jako skupinu jednoho nebo více vstupů. Po spuštění A/D převodu (ADST = 1) začne převod od prvního definovaného vstupu ve skupině. Po skončení A/D převodu prvního vstupu se okamžitě začne A/D převod dalšího vstupu v řadě. A/D převod pokračuje cyklicky po zvolených kanálech dokud bit ADST není nastaven do 0. Výsledky A/D převodu jsou ukládány do datových registrů (ADDRA-ADDRD), které odpovídají jednotlivým vstupům. Jejich přiřazení je popsáno v tabulce 3-2.

Pokud chceme změnit operační mód nebo vstup, musíme, stejně jako v předchozím případě (SCAN = 0), nejdříve vypnout A/D převod nastavením ADST do 0. Po provedení nezbytné změny se nastaví ADST do 1 a převod se opět spustí od prvního zvoleného vstupu ve skupině. Příklad A/D převodu ve scan módu je vidět na obrázku 3-3.

Příklad postupu, kde jsou nastaveny tři vstupy (AN0-AN2)

1. Nastavení scan módu (SCAN = 1), výběr vstupů AN0-AN2 (CH3 = 0, CH2 = 0, CH1 = 1, CH0 = 0) je popsán v kapitole 4.1.2., přerušení zakázáno (ADIE = 0), spuštění A/D převodu (ADST = 1)
2. Po skončení A/D převodu prvního vstupu (AN0) je výsledek uložen do datového registru ADDRA. Poté se automaticky spustí A/D převod druhého vstupu (AN1).
3. Stejný postup je i pro třetí vstup (AN2).
4. Po provedení A/D převodu všech zvolených vstupů (AN0-AN2)je nastaven příznak ADF do 1 a A/D převod prvního vstupu (AN0) začíná znovu. Pokud je současně povoleno přerušení (ADIE = 1), tak je voláno po dokončení A/D převodu.
5. Kroky 2 až 4 jsou prováděny dokud bit ADST zůstává nastaven v 1. Pokud je bit ADST nastaven do 0, A/D převod je okamžitě ukončen a nedokončený převod je ignorován. Po opětovném nastavení bitu ADST do 1 je znovu od prvního vstupu (AN0) spuštěn A/D převod .



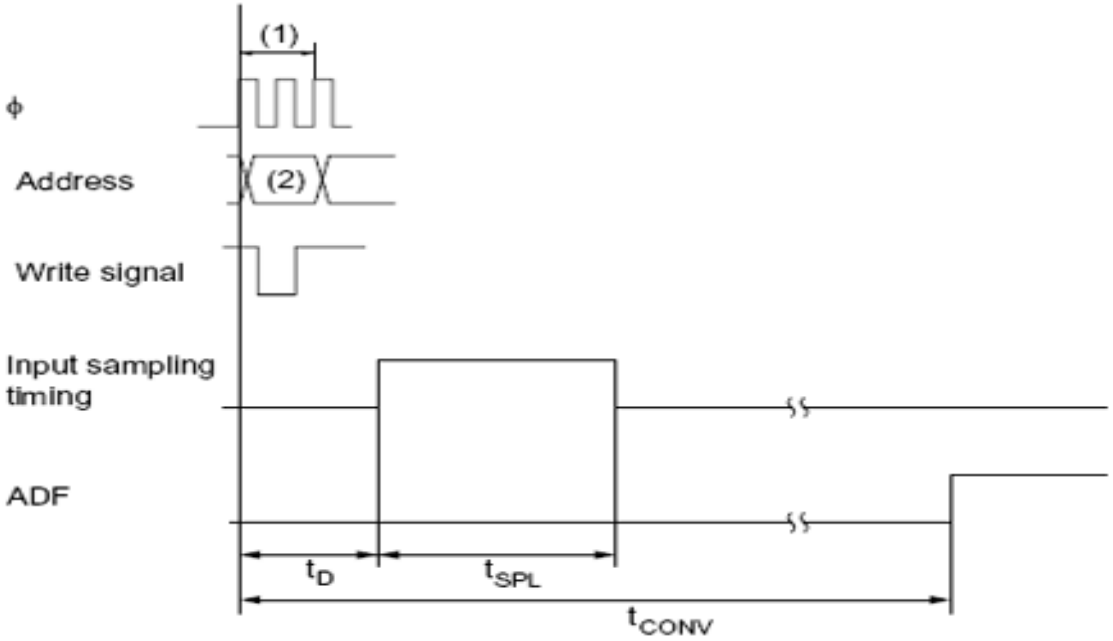
Poznámka: * kolmé šipky označují instrukce provedeny softwarově

Obrázek 3-3: Příklad A/D převodu ve Scan módu a vybraných kanálech AN0 až AN2

3.3 Vstupní vzorkování a doba A/D převodu

A/D převodník má v sobě vzorkovací obvod s pamětí. A/D převodník navzorkuje analogový signál za dobu t_D hned potom co je bit ADST nastaven do 1 a následně začne převod.

Jak je vidět z obrázku 3-4, doba A/D převodu se skládá ze zpoždění začátku A/D převodu t_D a ze vstupního vzorkovacího času $t_{SPL.}$ Délka t_D závisí na časovém nastavení délky A/D převodu. Nastavení délky A/D převodu vychází z nastavení bitů CKS1 a CKS0 registru ADCR a je přinejmenším 10 µs. Nastavení pro single mode (SCAN = 0) je zobrazeno v tabulce 3-3. Nastavení pro scan mode (SCAN = 1) je zobrazen v tabulce 3-4.



Legenda:

(1): Zapisovací cyklus ADCSR

(2): Adresování ADCSR

t_D: Zpoždění začátku A/D převodu

t_{SPL}: Vstupní vzorkovací čas

t_{CONV}: Celkový čas A/D převodu

Obrázek 3-4: Doba A/D převodu

		CKS1 = 0						CKS1 = 1					
		CKS0 = 0			CKS0 = 1			CKS0 = 0			CKS0 = 1		
Popis	Symbol	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
Zpoždění začátku A/D převodu (stavů)	tD	18	-	33	10	-	17	6	-	9	4	-	5
Vstupní vzorkovací čas (stavů)	tSPL	-	127	-	-	63	-	-	31	-	-	15	-
Celkový čas A/D převodu (stavů)	tCONV	515	-	530	259	-	266	131	-	134	67	-	68

Tabulka 3-3: Doba A/D převodu podle nastavení bitů CKS1 a CKS0 pro single mód (SCAN = 0)

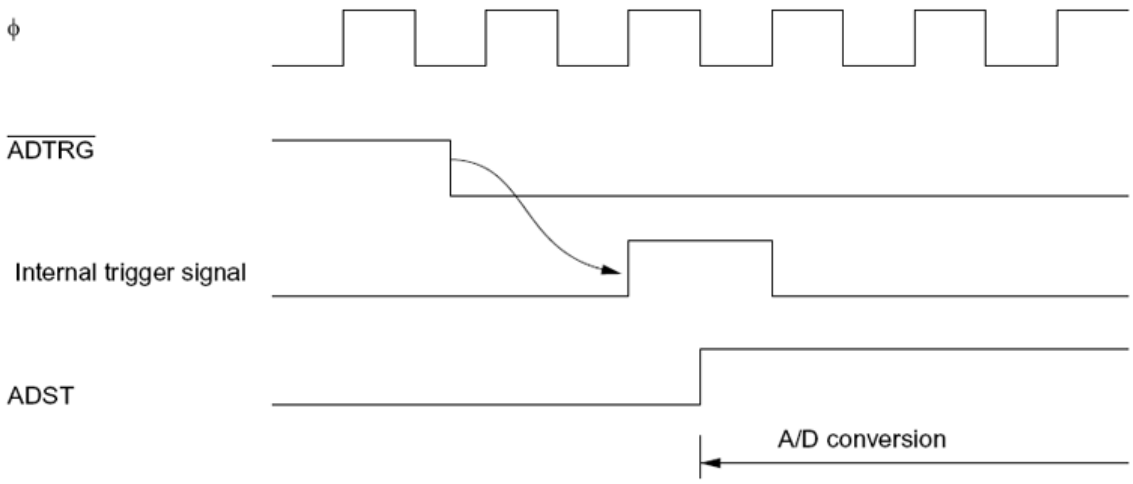
CKS1	CKS0	Doba A/D převodu (stavů)
0	1	512 (fixně)
0	1	256 (fixně)
1	0	128 (fixně)
1	1	64 (fixně)

Tabulka 3-4: Doba A/D převodu podle nastavení bitů CKS1 a CKS0 pro scan mód (SCAN = 1)

3.4 Externí spouštění A/D převodu

A/D převod může být spuštěn externím signálem na pinu $\overline{ADTRG}$. Rozhodují o tom bity TRGS1 a TRGS0 registru ADCR. Popis je uveden v kapitole 3.1.3..

Sestupnou hranou na pinu $\overline{ADTRG}$ se nastaví bit ADST do 1. Všechny ostatní operace jsou stejné jako když je A/D převodník spuštěn softwarově. Na obrázku 3-5 je zobrazeno nastavení bitu ADST externím signálem.



Obrázek 3-5: Externí spouštění A/D převodu

3.5 Přerušení

A/D převodník generuje přerušení (ADI) po skončení A/D převodu. Povolení přerušení může být povoleno nastavením bitu ADIE v registru ADCSR.

Kontrolér přenosu dat (DTC) může být aktivován přerušením (ADI).

3.6 Převedené měřená hodnoty

Při zpracovávání hodnot ze senzorů v řídícím programu, používáme hodnoty po A/D převodu. V tabulce 3-5 je uvedena závislost těchto hodnot na vzdálenosti od překážky. V tabulce 3-6 jsou hodnoty, které odpovídají detekci černá čáry, které jsou zaokrouhleny na celé stovky.

Vzdálenost překážky [cm]	10	15	20	25	30	35	40
Měřené hodnota (Přední senzor)	497	350	275	225	191	165	148
Měřené hodnota (Zadní senzor)	484	342	266	215	185	160	141

Vzdálenost překážky [cm]	45	50	55	60	65	70	75	80
Měřené hodnota (Přední senzor)	133	120	108	98	89	84	78	72
Měřené hodnota (Zadní senzor)	130	115	102	90	84	78	72	68

Tabulka 3-5: Závislost měřených hodnot na vzdálenosti od překážky

	Měřená hodnota		
Povrch podložky	QRD1114-1	QRD1114-2	QRD1114-3
Bílá	200	200	200
Černá	1000	1000	1000

Tabulka 3-6: Měřené hodnoty podle barvy podložky

3.7 Nastavení A/D převodníku v realizované aplikaci

V této sekci je uvedeno konkrétní nastavení jednotlivých registrů použitého A/D převodníku, jak bylo použito v realizovaném programu. Tyto registry jsou nastavovány pro vyčítání hodnot z použitých senzorů ve funkcích read_QRD1( ), read_QRD2( ), read_QRD3( ), read_GP1( ), read_GP2( ).

Registr ADCSR

V tomto registru se nastavují bity pro řízení A/D převodu. V programu je použito následující nastavení jednotlivých bitů.

ADF je automaticky po skončení A/D převodu nastaven do 1. Před dalším spuštěním A/D převodu je potřeba ho softwarově nastavit do 0, jinak A/D převod nezačne.

V programu je to provedeno takto:

```
*AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;
```

ADIE není nastaven, protože v programu nepoužíváme přerušení. Tedy ADIE = 0.

ADST je bit, kterým se spouští samotný A/D převod. Počáteční hodnotou je nastaven na 0. Před požadovaným začátkem A/D je softwarově nastaven do 1.

V programu je to provedeno takto:

```
*AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;
```

Po skončení A/D převodu je bit ADST automaticky nastaven do 0.

SCAN určuje nastavení skenovacího módu. V programu je použit Single mode, který převádí hodnoty jen z jednoho vstupu. Počáteční hodnota bitu SCAN je nastavena právě na Single mode, takže není potřeba ho softwarově nijak nastavovat.

CH3-CH0 určují, který vstup se bude zpracovávat. Nastavení je popsáno v kapitole 3.1.2 v sekci „Bity 0 až 2“.

Realizace v programu pro QRD114-1 na vstupu AN6:

```
*AD_ADCSR = ADCSR_CH2m | ADCSR_CH1m;
```

Registr ADDR (ADDRA - ADDRD)

V tomto adresovém registru jsou ukládány výsledky A/D převodu. Adresový registr ADDRA (ostatní registry analogicky) je 16-bitový a je rozdělena na dva 8-bitové registry ADDRAH (horních 8-bitů) a ADDRAL (8-dolních bitů). Jednotlivým vstupům do A/D převodníku odpovídají registry ADDRA-ADDRD podle tabulky 3-2.

Pro výstupy jednotlivých senzorů je přiřazení v programu následující:

Hodnoty senzoru QRD1114-1 na vstupu AN6 je uložen do registru ADDRC

Hodnoty senzoru QRD1114-2 na vstupu AN7 je uložen do registru ADDRD

Hodnoty senzoru QRD1114-3 na vstupu AN8 je uložen do registru ADDRA

Hodnoty senzoru GP2D12-1 na vstupu AN9 je uložen do registru ADDRB

Hodnoty senzoru GP2D12-1 na vstupu AN10 je uložen do registru ADDRC

Pro zpracování uloženého převodu je potřeba bitového posunu, protože výstup z A/D převodníku je 10-bitové číslo, které je uloženo na bitech 6-15.

Realizace v programu pro QRD114-1 na vstupu AN6:

```
AN6h = *AD_ADDRCH;
AN6d = *AD_ADDRCL;
AN6 = (AN6d >> 6) | (AN6h << 2);
```

Kde AN6h je horní byte, AN6d je dolní byte a AN6 je 10-bitové číslo A/D převodu.

Registr ADCR

V tomto registru se povoluje, nebo zakazuje externí spouštění A/D převodu pomocí bitů TRGS1 a TRGS0. V programu je použitou pouze softwarové spouštění A/D převodu, což odpovídá původnímu nastavení.

Dále se zde nastavuje čas A/D převodu. V programu je použito základní nastavení, tedy čas převodu 530 stavů, což přibližně odpovídá 10 µs.

Jelikož obě požadované nastavení odpovídají původnímu, není potřeba tento registr v programu měnit.

Registr MSTPCRA

V tomto registru se zapíná A/D převodník. Normálně je A/D převodník vypnut (nastaven do 1), kvůli úspoře energie. Pro zapnutí A/D převodníku musíme nastavit bit MSTPA1 do 0.

Realizace v programu:

```
*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;
```

3.8 Popis realizované aplikace

Ve vytvořeném programu jsou dvě řídící funkce pro pohyb robota. Jsou to funkce autonomni_pohyb( ) a sleduj_caru( ). Obě jsou volané z hlavní funkce main( ).

V současné době je program vytvořený tak, že pohyb robota je vykonáván pouze podle jedné řídící funkce. Pro řízení pohybu robota podle druhé řídící funkce je potřeba v hlavní funkci main( ) změnit volání řídící funkce, program přeložit a nový program nahrát do mikrokontroléru robota.

Funkce autonomni_pohyb( ) umožňuje jízdu po prostoru, kde jsou aktivní senzory vzdálenosti. Program řídí pohyb robota tak, aby se vyhýbal překážkám. Není zde implementována žádná složitá inteligence.

Tato řídící funkce obsahuje nekonečnou smyčku, ve které je vykonáváno řízení pohybu robota. Jsou zde použity příznaky stav, pro jízdu rovně nebo zatáčení, a směr, pro jízdu dopředu nebo dozadu. Neustále se testuje vzdálenost od překážky. Pokud je vzdálenost menší než 20cm, čemuž podle tabulky 3-5 odpovídá hodnota 275, tak se po dobu 5 - 12s, bude robot otáčet s 70% pravděpodobností doprava. Po otočení pokračuje robot v jízdě rovně. A postup se bude analogicky opakovat. Směr pohybu robota je změněn po šesti zatočeních.

Funkce sleduj_caru( ) řídí jízdu robota po čáře nakreslené na podlaze. Jako čáru jsme použili černou, elektrikářskou pásku o šířce 18mm. Testování jsme prováděli na bílé sololitové desce, dřevěné plovoucí podlaze a na šedivém PVC linoleu. Hodnoty těchto povrchů byly od 200 pro bílou až po 290 pro šedivé linoleum. Řídící program pak testuje hodnoty jednotlivých senzorů. Jako hraniční byla zvolena hodnota 500, která spolehlivě určuje přechod z okolního povrchu na černou čáru.

Základem této řídící funkce je prostřední senzor QRD114-3, který snímá černou čáru nakreslenou na zemi. Pokud je nad čarou, tak hodnota ze senzoru je kolem 1000. Pokud při jízdě robota tento senzor udává hodnotu nižší než 800, znamená to, že robot již nejede svým středem na čarou.V tuto chvíli již jeden z krajních senzorů zvýšil hodnotu svého výstupu nad 500, tedy detekoval černou čáru. Ihned je spuštěna funkce move( ), která provede potřebné zatočení. Dále se řízení opakuje ve stejné, nekonečné smyčce.

V příloze č.6 je přiložen zdrojový kód programu , ve kterém jsou řídící funkce detailněji okomentovány.

4 Zhodnocení a závěr práce

Práci vykonanou v rámci této bakalářské práce je možné rozdělit na dvě části.

První část je tvořena návrhem a realizací senzorové DPS, na které je umístěna elektronika potřebná ke správnému fungování zvolených senzorů. Při práci byly také odstraněny chyby na řídící desce. Výsledná senzorová DPS, která je výsledkem první části mé bakalářské práce, funguje ke dnešnímu dni správně. Dále byly zakoupeny dva akumulátory o kapacitě 650mAh, díky kterým se může robot pohybovat, podle zvoleného programu, 2,5 hodiny.

V druhé části byl vytvořen program, který čte data z A/D převodníku a na základě těchto hodnot řídit pohyb robota podle zvolených řídících funkcí.

Má práce měla rozšířit stávající stav robota o senzorickou část tak, aby bylo možné na robotovi dále testovat složitější, inteligentní algoritmy. Tento úkol jsem úspěšně splnil a robot je plně připraven k dalšímu, programovému vylepšení.

5 Použitá literatura

[1] Kovačik, Petr. *Robotičtí fotbaloví hráči,* 2006, Diplomová práce ČVUT FEL

[2] Punčochář, Josef. *Operační zesilovače v elektronice*, Vydavatelství BEN, 2002, ISBN 80-7300-059-8

[3] Záhlava, Vít. *OrCAD pro Windows*, Vydavatelství GRADA, 1999, ISBN 80-7169-876-8

[4] Tým autorů, *Sharp GP2D12/GP2D15*, http://sharp-world.com

[5] Tým autorů, *Fairchild QRD1113/1114*, http://www.fairchildsemi.com

[6] Tým autorů, *Fairchild KA7805/KA7805A*, http://www.fairchildsemi.com

[7] Tým autorů, *Texas Instruments TLC272, TLC272A, TLC272B, TLC272Y, TLC277 LinCMOS Precision dual operational amplifiers*, http://www.ti.com

[8] Herout, Pavel. *Učebnice jazyka C*, Vydavatelství Kopp, 2006, ISBN 80-7232-220-6

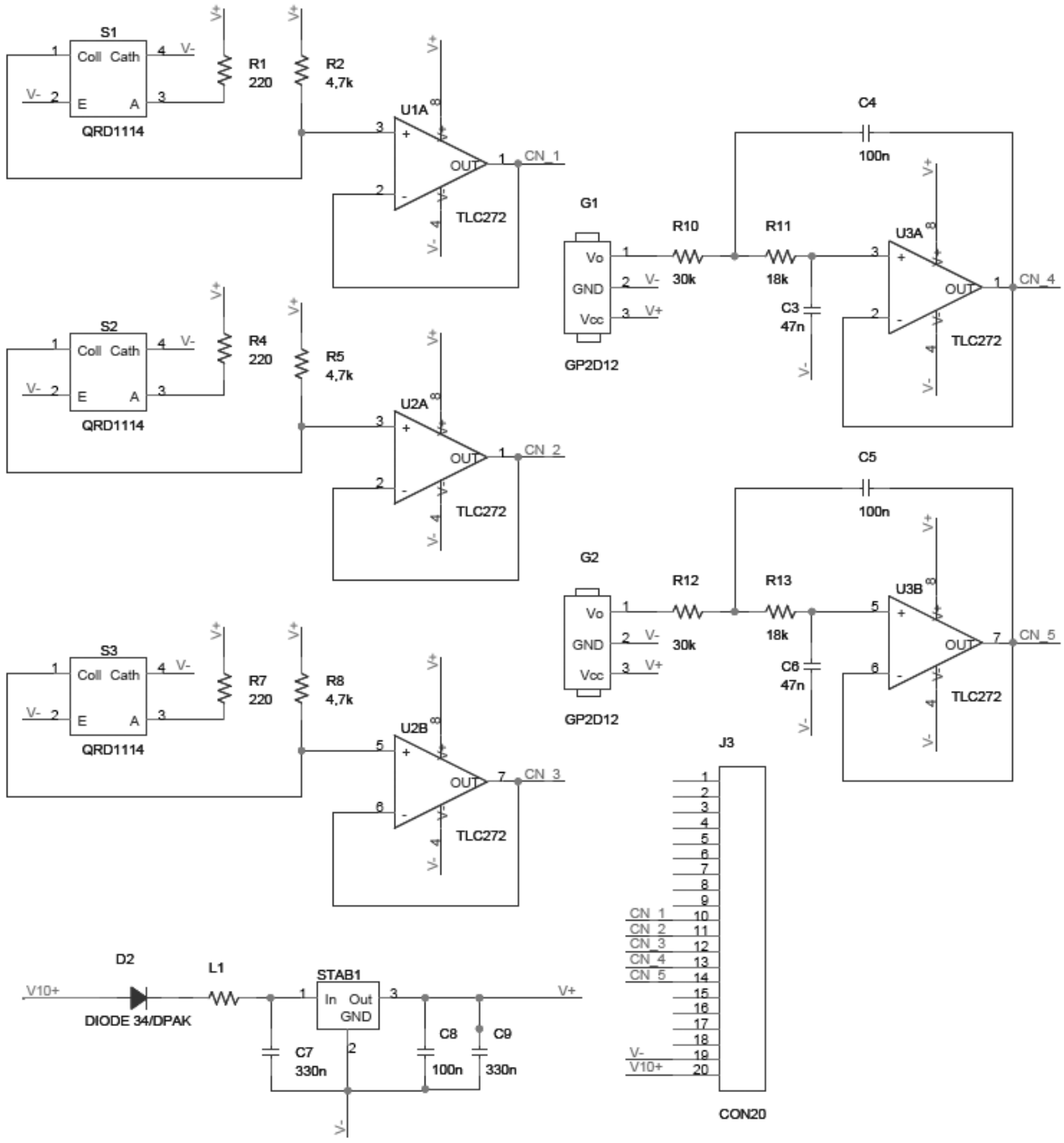
[9] Tým autorů, *Renesas 16-Bit Single-Chip Microcomputer Hardware Manual H8S/2639, H8S/2638, H8S/2636, H8S/2630 Group*, 2004

[10] oficiální web Ubuntu, www.ubuntu.cz

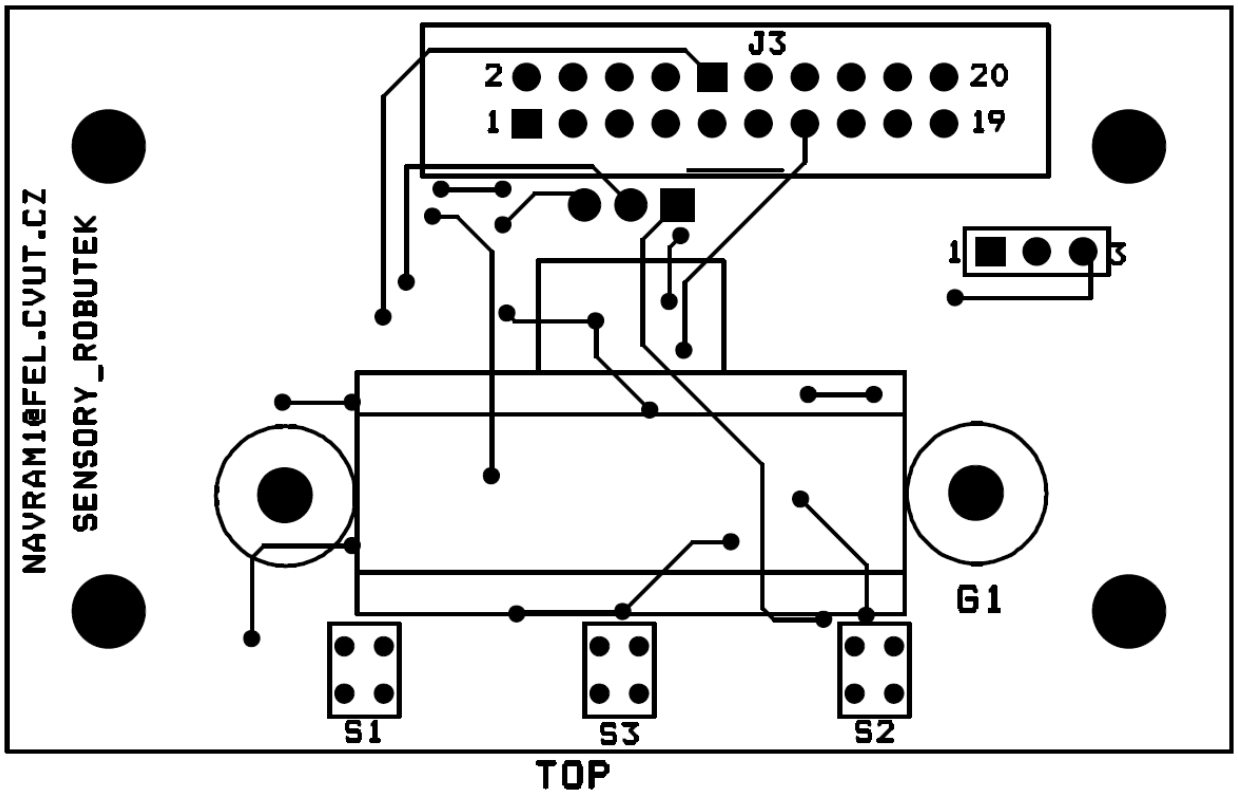
[11] web firmy Fulgurbattman, http://www.fulgurbattman.cz

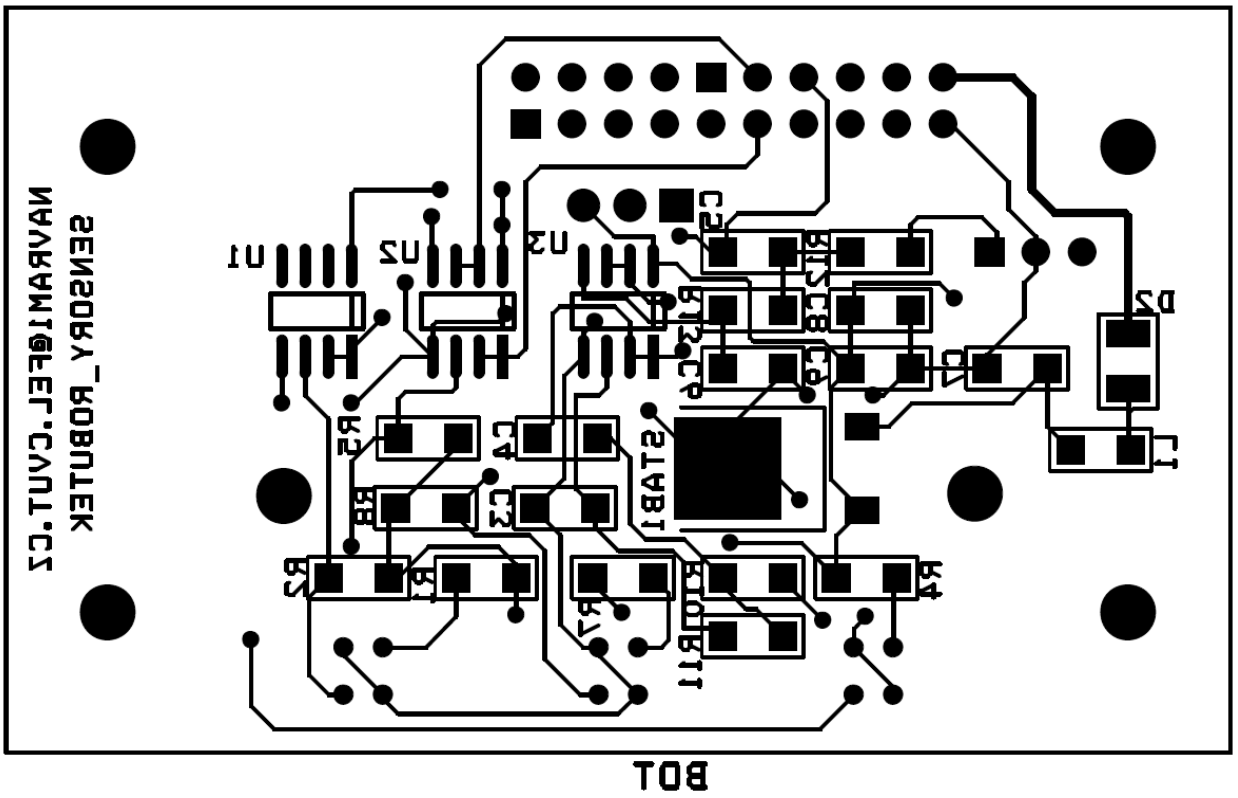
[12] web Battery University, http://www.batteryuniversity.com

Příloha č. 1 Schéma zapojení senzorové desky



Příloha č. 2 Návrh DPS strany TOP a BOTTOM





Příloha č. 3 Seznam součástek

Sum	Označení	Hodnota	Pouzdro
2	C3	47nF	SMD 1199
	C6	47nF	SMD 1200
3	C4	100nF	SMD 1201
	C5	100nF	SMD 1202
	C8	100nF	SMD 1203
2	C7	330nF	SMD 1204
	C9	330nF	SMD 1205
1	D2	BAS32SMD	SMD 1206
2	G1	GP2D12	
	G2	GP2D12	
1	J3	Konektor 14	MLWG14G
1	L1	10μH	SMD 1206
3	R1	220kΩ	SMD 1207
	R4	220kΩ	SMD 1208
	R7	220kΩ	SMD 1209
3	R2	4,7kΩ	SMD 1210
	R5	4,7kΩ	SMD 1211
	R8	4,7kΩ	SMD 1212
2	R10	30kΩ	SMD 1213
	R12	30kΩ	SMD 1214
2	R11	18kΩ	SMD 1215
	R13	18kΩ	SMD 1216
1	STAB1	KA7805	DPAK
3	S1	QRD1114	
	S2	QRD1114	
	S3	QRD1114	
3	U1	TLC272	SMD
	U2	TLC272	SMD
	U3	TLC272	SMD

Příloha č. 4 Seznam obrázků

Příloha č. 5 Seznam tabulek

Příloha č. 6 Zdrojový kód řídícího programu

```
/*
*
* Author funkci na pohyb motoru:   Petr Kovacik <kovacp1@feld.cvut.cz>, 2006
*
* Autor funkci na autonomni pohyb: Milan Navratil <navram1@fel.cvut.cz>, 2008
*
*
* Popis: main vola funkce autonomni_pohyb() , nebo sleduj_caru(),
* obe funkce pak volaji funkce na vycteni dat z 10-bitoveho A/D prevodniku
*
*/

#define _USE_EXR_LEVELS 1
#include <types.h>
#include <cpu_def.h>
#include <h8s2638h.h>
#include <periph/sci_rs232.h>
#include <system_def.h>
#include <stdlib.h>
#include <string.h>

#include <pxmc.h>
#include <pxmcbsp.h>

#include <cmd_proc.h>
#include "cmd_pxmc.h"
#include "timer3.h"
#include <stdio.h>

uint16_t read_GP1(void);                // deklarace funkce read_GP1
uint16_t read_GP2(void);                // deklarace funkce read_GP2
uint16_t read_QRD1(void);               // deklarace funkce read_QRD1
uint16_t read_QRD2(void);               // deklarace funkce read_QRD2
uint16_t read_QRD3(void);               // deklarace funkce read_QRD3



cmd_des_t const *cmd_rs232_default[];

/*struktury prikazu cmd*/
cmd_des_t const cmd_des_help={0, 0,"HELP","prints help for commands",
                        cmd_do_help,{(char*)&cmd_rs232_default}};

cmd_des_t const *cmd_rs232_default[]={

  &cmd_des_help,
  CMD_DES_CONTINUE_AT(cmd_pxmc_default),
  NULL
};
cmd_des_t const **cmd_bth=cmd_rs232_default;       /*cmd prikazy pro bth*/
cmd_des_t const **cmd_rs232=cmd_rs232_default;     /*cmd prikazy pro PC*/


/*struktury charakterizujici motor 0*/
pxmc_state_t mcsX0={
  pxms_flg:0,
  pxms_do_inp:0,
  pxms_do_con:0,
  pxms_do_out:0,
```

```
  pxms_do_deb:0,
  pxms_do_gen:0,
  pxms_ap:0, pxms_as:0,
  pxms_rp:155l*256,
  pxms_rs:0, //pxms_subdiv:8,
  pxms_md:8000l<<8, pxms_ms:5000, pxms_ma:5,    //pxms_ma akcelerace robota
  pxms_inp_info:(long)TPU_TCNT1,//TPU_TCNT1          /*chanel TPU A,B*/
  pxms_out_info:(long)PWM_PWBFR1A,              /*chanel PWM A,B*/
  pxms_ene:0, pxms_erc:0,
  pxms_p:40, pxms_i:0, pxms_d:1, pxms_s1:0, pxms_s2:0,
  pxms_me:0x1800,                               //6144
  pxms_ptirc:40,                          // 2000 irc per rev, 200/4 steps /
  pxms_ptper:1,
  pxms_ptptr1:NULL,
  pxms_ptptr2:NULL,
  pxms_cfg:PXMS_CFG_MD2E_m|PXMS_CFG_HLS_m|
      PXMS_CFG_HPS_m|PXMS_CFG_HDIR_m|0x1
};


/*struktury charakterizujici motor 1*/
pxmc_state_t mcsX1={
  pxms_flg:0,
  pxms_do_inp:0,
  pxms_do_con:0,
  pxms_do_out:0,
  pxms_do_deb:0,
  pxms_do_gen:0,
  pxms_ap:0, pxms_as:0,
  pxms_rp:155l*256,
  pxms_rs:0, //pxms_subdiv:8,
  pxms_md:8000l<<8, pxms_ms:5000, pxms_ma:5,
  pxms_inp_info:(long)TPU_TCNT2,                /*chanel TPU C,D*/
  pxms_out_info:(long)PWM_PWBFR1C,              /*chanel PWM C,D*/
  pxms_ene:0, pxms_erc:0,
  pxms_p:40, pxms_i:0, pxms_d:1, pxms_s1:0, pxms_s2:0,
  pxms_me:0x1800, //6144
  pxms_ptirc:40, // 2000 irc per rev, 200/4 steps /
  pxms_ptper:1,
  pxms_ptptr1:NULL,
  pxms_ptptr2:NULL,
  pxms_cfg:PXMS_CFG_MD2E_m|PXMS_CFG_HLS_m|     //FIXME: nastavit spravne
priznaky pro dalsi motorove struktur
      PXMS_CFG_HPS_m|PXMS_CFG_HDIR_m|0x1
};

pxmc_state_t *pxmc_main_arr[] = {&mcsX0,&mcsX1};

pxmc_state_list_t pxmc_main_list = {
  pxml_arr:pxmc_main_arr,
  pxml_cnt:sizeof(pxmc_main_arr)/sizeof(pxmc_main_arr[0])
};
```

```
//*****************************************************

void  unhandled_exception(void) __attribute__ ((interrupt_handler));

/*Interrupt routines*/
void  unhandled_exception(void)
{
};
//*****************************************************

extern cmd_io_t cmd_io_rs232_line;

extern void _print(char *str);

void
init(void)
{

/*************************************************************************
****/
  *DIO_PJDDR=0xff;      /*output gate*/
  *DIO_PEDDR=0xff;      /*output gate*/
  *DIO_PEDR=0x60;       /*0x0-LED - light all; 0x6 -ENA,ENB=1, LE33CD=0*/
  *DIO_PJDR=0x00; //rozsviceni vsech diod na */


  /*priority preruseni - SCI > TPU*/
/*   *SYS_SYSCR|=SYSCR_INTM1m; */
/*   *INT_IPRA=0x22; *INT_IPRB=0x22; *INT_IPRC=0x04;  */
/*   *INT_IPRD=0x40; *INT_IPRE=0x44; *INT_IPRF=0x55; */
/*   *INT_IPRG=0x55; *INT_IPRH=0x55; *INT_IPRJ=0x06; */
/*   *INT_IPRK=0x67; *INT_IPRM=0x66; */

  /*povoleni vsech preruseni atd...*/
  cli();
  excptvec_initfill(unhandled_exception, 0);

  /*nastaveni seriovych linek - Bth, PC*/
  //sci_rs232_setmode(RS232_BAUD_RAW | 3, 0, 0, 2);  // HCI - hardcoded
115200
  //sci_rs232_setmode(115200, 0, 0, 2);   // HCI
  sci_rs232_setmode(19200, 0, 0, sci_rs232_chan_default); //PC
  sti();

  _print("Start\n");

  pxmc_initialize();

  if (init_timer3((long long)100/*ms*/*1000*1000) < 0) {
    /* ERROR */
    DEB_LED_OFF(0);
  }
}


// Distance of wheels - d
#define WHEEL_DIST 74 /* mm */
#define MAX_R 300 /* mm */
#define MIN_R (WHEEL_DIST/2) /* mm */
```

```
void move(int speed, int r)        // funkce na pohyb motoru
{
  int sl, sr;
  int d = WHEEL_DIST;


  if (r != 0) {
    sr = speed + (long long)speed * (d/2) / r;
    sl = speed - (long long)speed * (d/2) / r;
  } else {
    sr = speed;
    sl = speed + 10;          // minus 100 je kompenzace zataceni
  }
  pxmc_spd(&mcsX0, +sl, 0);
  pxmc_spd(&mcsX1, -sr, 0);
  printf("speed=%5d, r=%5d, sl=%5d, sr=%5d\n", speed, r, sl, sr);
}

void autonomni_pohyb(void)
{
  long int now = get_timer();
  long int next = now;
  long int otoc_next = now;
  long int time_next = now;
  int speed = 3000;
  int radius = 100;
  int stav = 1;                                  // podle priznaku stavu
                                                 // se rozhoduje,zda pojede
                                                 // rovne, nebo zatoci
  int otoc = 0;                                  // priznak pro otoceni
  int smer = 1;                                  // smer = 1 robot jede
dopredu, smer = -1 robot jede dozadu
  uint16_t vzdalenost, vzdalenost_zpet, cas=0;

  while (1) {

    long int now = get_timer();

    vzdalenost = read_GP1();
    vzdalenost_zpet = read_GP2();

    if (now >= time_next){                       // podminka vypisu 1x za s
      time_next = now + 10;
      cas++;

      /*Vypsani vzdalenosti na terminal*/
      printf("Predni: %d\tZadni: %d\tSpeed: %d\tSmer: %d\tOtoc: %d\n",
      vzdalenost, vzdalenost_zpet, speed, smer, otoc);
    }                                           // konec if podminky na vypis za 1s

    if (stav == 1 && now >= next){              // podminka pro jizdu
      stav = 0;
      move(-speed,0);
      otoc++;
    }

    if (stav == 0 && vzdalenost >= 275 && smer == 1){
                        // podminka pro zatoceni,275 odpovida (mensi nez 20cm)

      if (rand() % 100 < 70)                    // 50 % predpoklad pro zatoceni doprava
move(-speed,radius);                            // zatoc rychlosti speed a polomerem
                                                // radius(doprava)
```

```
      else
        move(-speed,-radius);       // zatoc rychlosti speed a polomerem
                                    // -radius (doleva)
      stav = 1;

      next = now + ((rand() % 8)+5);// zataceni po dobu 5 - 12 s
    }                                          // konec if


    if (stav == 0 && vzdalenost_zpet >= 280 && smer == -1){            //
jizda zpet!!! podminka pro zatoceni 260 odpovida (mensi nez 20cm)

      if (rand() % 100 < 70)        // 50 % predpoklad pro zatoceni doprava
        move(-speed,radius);        // zatoc rychlosti speed a radiusem,
      else
        move(-speed,-radius);
      stav = 1;

      next = now + ((rand() % 8)+5);                       // zataceni po
dobu 5 - 12 (nahodne cislo)
    }                                          //konec if


  /* zmena_smeru */
    if (otoc >= 6){
      speed = -speed;
      otoc = 0;
      smer = -smer;
      move(-speed,0);
    }

  }                                            //konec while
}                                              //konec funkce


void sleduj_caru(void){
  long int now = get_timer();
  long int next = now;
  long int time_next = now;
  int speed = 2000;
  int radius = 100;
  uint16_t leva, stred, prava, cas=0;

  move(2000,0);
  while(1){

    long int now = get_timer();

    stred = read_QRD3();  // pohled zpredu, cteni senzoru odrazu QRD1114 1-3
    leva = read_QRD1();
    prava = read_QRD2();

    if (stred > 1000 && leva < 500 && prava <500 && now >= next){
      move(-speed,0);
    }

    if (stred < 800 && leva > 500){
      move(-speed,radius);
      next = now + 1;
    }
```

```
    if (stred < 800 && prava > 500){
      move(-speed,-radius);
      next = now + 1;
    }

    if (now >= time_next){          // podminka vypisu, jednou za vterinu
      time_next = now + 10;
      cas++;

      /*Vypsani vzdalenosti na terminal*/
      printf("V case %ds\tLeva: %d\tStredni: %d\tPrava: %d\n", cas, leva,
      stred, prava);                // vypisuje hodnoty povrchu po 1s

    }                                     // konec if podminklz na vypis za 1s
  }                                             // konec hlavni smycky
}                                               // konec funkce sleduj_caru()



uint16_t read_QRD1(void){ // funkce na vycteni informaci z AD prevodniku z
                         // ADDRC kde je nastaven AN6 tedy QRD114-1-leva zpredu

*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;      // Zapnuti AD prevodniku

*AD_ADCSR = ADCSR_CH2m | ADCSR_CH1m;     // nastaveni analog vstupu AN8, tedy
            // vystup ze senzoru QRD114-3-prostredni. CH3=0,CH2=1,CH1=1,CH0=0

  uint8_t AN6h, AN6d;
  uint16_t AN6;   // AN6h je horni byte AD prevodu, AN6d je dolni byte.
                  // Jsou cteny z ADDRC (AN6 je cele 10-bitove cislo,
                  // viz tabulka 4-2 [bp_2008_Navratil_Milan.pdf])

  int i;

  for(i = 0; i < 5; i++){       // for cyklus na 5 krat probehnuti smicky
    *AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;  // softwarove nastaveni ADF do 0
    *AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;  // spusteni AD prevodu

    while((*AD_ADCSR & ADCSR_ADFm) == 0) { // cekaci smycka dokud neskonci
                                           // AD prevod
      ;
    }

    /*Precteni vysledku AD prevodu z ADDRC*/
    AN6h = *AD_ADDRCH;        // Cteni horniho bytu
    AN6d = *AD_ADDRCL;        // Cteni dolniho bytu
    AN6 = (AN6d >> 6) | (AN6h << 2);
                  // Posun jednotlivych bitu, abychom dostali 10-bitove cislo

   }                                        // konec for

   return (AN6);                            // return pro sleduj_caru()

}                                           // konec funkce
```

```
uint16_t read_QRD2(void){ // funkce na vycteni informaci z AD prevodniku z
                  // ADDRD kde je nastaven AN7 tedy QRD114-2-prava zpredu

*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;      // Zapnuti AD prevodniku
*AD_ADCSR = ADCSR_CH2m | ADCSR_CH1m | ADCSR_CH0m;
// nastaveni analog vstupu AN8 - tedy vystup ze senzoru QRD114-3-prostredni.
//CH3=0,CH2=1,CH1=1,CH0=1

  uint8_t AN7h, AN7d;
  uint16_t AN7;   // AN7h je horni byte AD prevodu, AN7d je dolni byte.
                  // Jsou cteny z ADDRD   int i;

  for(i = 0; i < 5; i++){          // for cyklus na 5 krat probehnuti smicky
    *AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;  // softwarove nastaveni ADF do 0
    *AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;             // spusteni AD prevodu

    while((*AD_ADCSR & ADCSR_ADFm) == 0){
                             // cekaci smycka dokud neskonci AD prevod
      ;
    }

    /*Precteni vysledku AD prevodu z ADDRD*/
    AN7h = *AD_ADDRDH;                                 //Cteni horniho bytu
    AN7d = *AD_ADDRDL;                                 //Cteni dolniho bytu
    AN7 = (AN7d >> 6) | (AN7h << 2);     //Posun jednotlivych bitu, abychom
                                         // dostali 10-bitove cislo

   }                                           //konec for

   return (AN7);                               // return pro sleduj_caru()

}                                              // konec funkce


uint16_t read_QRD3(void)// funkce na vycteni informaci z AD prevodniku z
                       // ADDRA kde je nastaven AN8 tedy QRD114-3-prostredni
{
*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;      //Zapnuti AD prevodniku
 *AD_ADCSR = ADCSR_CH3m;      // nastaveni analog vstupu AN8 - tedy vystup ze
                  // senzoru QRD114-3-prostredni. CH3=1,CH2=0,CH1=0,CH0=0

  uint8_t AN8h, AN8d;
  uint16_t AN8;   //AN8h je horni byte AD prevodu, AN8d je dolni byte.
                  // Jsou cteny z ADDRA
  int i;

  for(i = 0; i < 5; i++){          // for cyklus na 5 krat probehnuti smicky
    *AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;  // softwarove nastaveni ADF do 0
    *AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;  // spusteni AD prevodu

    while((*AD_ADCSR & ADCSR_ADFm) == 0) {                 // cekaci smycka
dokud neskonci AD prevod
      ;
    }

    /*Precteni vysledku AD prevodu z ADDRA*/
    AN8h = *AD_ADDRAH;                                 // Cteni horniho bytu
    AN8d = *AD_ADDRAL;                                 // Cteni dolniho bytu
    AN8 = (AN8d >> 6) | (AN8h << 2);
                  // Posun jednotlivych bitu, abychom dostali 10-bitove cislo

   }                                           // konec for
```

```
   return (AN8);                                    // return pro sleduj_caru()

}                                                   // konec funkce


uint16_t read_GP1(void) { // funkce na vycteni informaci z AD prevodniku
                  // z ADDRB kde je nastaven AN9 tedy GP2D12 predni

*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;      // Zapnuti AD prevodniku
*AD_ADCSR = ADCSR_CH3m | ADCSR_CH0m;     // nastaveni analog vstupu AN9, tedy
                  // vystup ze senzoru GP2D12-predni. CH3=1,CH2=0,CH1=0,CH0=1

  uint8_t AN9h, AN9d;
  uint16_t AN9;   // AN9h je horni byte AD prevodu, AN9d je dolni byte.
                  // Jsou cteny z ADDRB

  int i;

  for (i = 0; i < 5; i++){    // for cyklus na 5 krat probehnuti smicky , aby
                              // se ustalily hodnoty ze senzoru a nedavaly
                              // nahodnou hodnotu

*AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;     // softwarove nastaveni ADF do 0
*AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;     // spusteni AD prevodu

while((*AD_ADCSR & ADCSR_ADFm) == 0) {
            // cekaci smycka dokud neskonci AD prevod
      ;
    }

    /*Precteni vysledku AD prevodu z ADDRB*/
    AN9h = *AD_ADDRBH;                                  // Cteni horniho bytu
    AN9d = *AD_ADDRBL;                                  // Cteni dolniho bytu
    AN9 = (AN9d >> 6) | (AN9h << 2);
                  // Posun jednotlivych bitu, abychom dostali 10-bitove cislo

   }                                                // konec for

   return (AN9);                                    //return pro autonomni_pohyb()

}                                                   // konec funkce


uint16_t read_GP2(void){// funkce na vycteni informaci z AD prevodniku
                        // z ADDRC kde je nastaven AN10 tedy GP2D12 zadni

*SYS_MSTPCRA = *SYS_MSTPCRA & ~MSTPCRA_MSTPA1m;      // Zapnuti AD prevodniku
*AD_ADCSR = ADCSR_CH3m | ADCSR_CH1m;// nastaveni analog vstupu AN10, tedy
                  // vstup ze senzoru GP2D12-predni. CH3=1,CH2=0,CH1=1,CH0=0

  uint8_t AN10h, AN10d;
  uint16_t AN10;        //AN10h je horni byte AD prevodu, AN10d je dolni byte.
                        // Jsou cteny z ADDRC)
  int i;

   for(i = 0; i < 5; i++){    // for cyklus na 5 krat probehnuti smicky

*AD_ADCSR = *AD_ADCSR & ~ADCSR_ADFm;     // softwarove nastaveni ADF do 0
*AD_ADCSR = *AD_ADCSR | ADCSR_ADSTm;                // spusteni AD prevodu
```

```
    while((*AD_ADCSR & ADCSR_ADFm) == 0) {
                  // cekaci smycka dokud neskonci AD prevod
      ;
    }

    /*Precteni vysledku AD prevodu z ADDRC*/
    AN10h = *AD_ADDRCH;                              // Cteni horniho bytu
    AN10d = *AD_ADDRCL;                              // Cteni dolniho bytu
    AN10 = (AN10d >> 6) | (AN10h << 2);
           // Posun jednotlivych bitu, abychom dostali 10-bitove cislo*/

   }                                                 //konec for

   return (AN10);

}


int main()
{
  _print("Snowforce for life ! ! !\n");

  init();  /* TODO initialize rand accoring to voltage read from AD6. */

  autonomni_pohyb();    // volani hlavni funkce autonomniho pohybu robota

  //sleduj_caru();      // volani funkce na sledovani cary

  for (;;); // nekonecna smycka, pro odchyceni nahodneho skoku zpet do main

  return 0;

}
```

Příloha č. 7 Struktura přiloženého CD

Struktura přiloženého CD je na obrázku P1.

Obrázek P-1: Struktura adresářů na přiloženém CD

Adresář „Dokumentacni video“ obsahuje dvě video nahrávky funkčního robota se senzorickou deskou realizovanou jako část této bakalářské práce.

- autonomni_pohyb.avi
- sleduj_caru.avi

Adresář „Katalogove_listy“ obsahuje potřebné katalogové listy.

- 7805_DS.pdf
- H8S_2638.pdf
- QRD1114.pdf
- Sharp_GP2D12.pdf
- TLC272.pdf

Adresář „Obvodovy_navrh“ obsahuje soubory k výrobě senzorické DPS.

- SENSORY_ROBUTEK.GTD
- SENSORY_ROBUTEK.MAX
- SENSORY_ROBUTEK.OPJ

Adresář „Text_bakalarske_prace“ obsahuje tuto práci ve formátu pdf.

- BP_2008_Navratil_Milan.pdf

Adresář „Zdrojove_kody“ obsahuje všechny použité zdrojové kódy. Je potřeba stáhnou balík gcc-h8300-coff-3.4.3-bin, potřebný k přeložení zdrojových kódů. Ten je ke stažení na http://rtime.felk.cvut.cz/hw/index.php/H8S/2638.

- cmd_pxmc.c a cmd_pxmc.h
- Makefile a Makefile.omk
- navratil_ad.c
- timer.c a timer.h