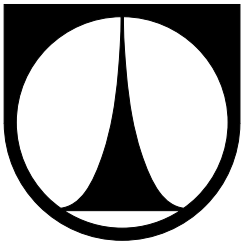
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií



Model autonomního vozu

Bakalářská práce

Liberec 2013 Filip Procházka

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B 2612 – Elektrotechnika a informatika

Obor: 2612R011 – Elektronické informační a řídicí systémy

Model autonomního vozu

Autonomous Car Model

Bakalářská práce

Autor práce: **Filip Procházka**

Vedoucí práce: Ing. Jan Koprnický, Ph.D.

Konzultant práce: –

V Liberci 16. května 2013

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Filip Procházka**
Osobní číslo:	**M10000110**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Model autonomního vozu**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s elektromechanickými částmi podvozku a řídicí jednotkou.
2. Zprovozněte hardwarovou část systému.
3. Navrhněte algoritmus řízení a vytvořte odpovídající software pro řídicí jednotku modelu vozu.
4. Funkční model otestujte na závodní dráze.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30–40 stran**

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

Seznam odborné literatury:

[1] **The Freescale Cup Knowledge Center. FREESCALE. The Freescale Cup Knowledge Center [online]. 2012 [cit. 2012-10-04]. Dostupné z: https://community.freescale.com/docs/DOC-1284**

[2] **Ďaďo, S.; Kreidel, M.: Senzory a měřicí obvody. Praha : ČVUT, druhé vydání, 1999, ISBN 80-01-02057-6.**

[3] **FREESCALE. MPC560xB Controller Board User's Guide. MPC560XBMCBUG, Rev. 0, 08/2012. 2012.**

Vedoucí bakalářské práce: **Ing. Jan Koprnický, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2012**

Termín odevzdání bakalářské práce: **17. května 2013**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2012

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/ 2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Janu Koprnickému, Ph.D. za jeho podporu a cenné rady, které mi poskytl při tvorbě bakalářské práce. Také bych rád poděkoval své rodině, která byla ochotná mě při studiu podporovat.

Abstrakt

Tato bakalářská práce se zabývá inteligentním vozidlem sledující čáru, které se pohybuje po neznámé dráze a má za úkol tuto dráhu zajet za co nejrychlejší čas. Práce je zpracována od sestavení vývojového kitu, zakoupeného u společnosti Freescale, přes seznámení se s prvky vozidla až k vytvoření řídicího algoritmu a následném vyzkoušení vozidla na dráze při interní školní soutěži.

Klíčová slova: Autonomní řízení, Mikrokontrolér, H-můstek, Lineární kamera

Abstract

This Bachelor's thesis deals with the intelligent vehicle pursuing a line that moves along an unknown path and it's task is to do it as fast as possible. This thesis is made from the assembly development kit purchased from Freescale company through the meeting with the elements of the vehicle to the creation of the controling algorithm followed by the testing of the vehicle on the track at internal school competition.

Keywords: Autonomous control, Microcontroller, H-bridge, Line Scan Camera

Obsah

Seznam obrázků

1 Úvod

Úkolem této bakalářské práce je seznámit se s elektronikou zakoupenou u společnosti Freescale Semiconductor a sestavit funkční autonomní auto, které bude schopné zajet co nejrychleji trať podle černého pruhu na bílém podkladu podle pravidel soutěže Freescale Cup, která se tento rok uskutečnila již po druhé i pro oblast Evropy.

Práce nás ze začátku seznámí s cílem bakalářské práce a základními pravidly, které by mělo splňovat inteligentní vozidlo. V další části zprávy se seznámíme se základními součástkami, jako je vývojový kit a jeho částmi, nebo plošný spoj, který nám bude ovládat stejnosměrné motory sloužící k pohonu auta. Po této části bude popsáno sestavení dodaných dílů a jejich zprovoznění, a nakonec tu popíšeme vznik řídicího algoritmu pro sledování čáry za pomocí lineární kamery.

Zakončení bakalářské práce je úspěšné zajetí tratě pokud možno v co nejkratším čase a porovnání s ostatními bakalářskými či diplomovými projekty, které mají stejné téma práce.

2 Freesclae Cup 2013

2.1 O soutěži

Freescale Cup je soutěž mezi inteligentními autíčky, která jsou většinou podobná RC modelům. Vítězem této soutěže se stane ten tým, kterému se podaří navrhnout takové vozítko, které zajede (bez vychýlení z trati) nejrychleji neznámou trať (ukázková trať obr. 1), která je vyznačena černým pruhem.

Tato soutěž už má několikaletou tradici ve světě, především v Asii. V Evropě se poprvé objevila v loňském roce, kdy se konala u nás v Praze. Při sestavování vozidla je nutné dodržovat některá pravidla, která určuje pořadatel soutěže, což je firma Freescale Semiconductor, jež je hlavním dodavatelem hardwaru pro Freescale Cup 2013. Evropské finále se konalo 26.–27. března 2013 v Paříži. Naše fakulta se účastní pouze závodu mezi místními studenty na konci letního semestru.

Obrázek 1: Ukázková trať (Freescale Cup Mexiko 2009) [7]

2.2 Základní pravidla pro rok 2013

Výčet základních pravidel [1]:

1. V návrhu musí být použity následující součástky (nezměněné)
   (a) DC motor

(b) Servomotor

i. Nezahrnuje propojení Servomotoru s přední nápravou

(c) Baterie (NiMH 7,2 V, 2400 mAh)

i. Použití pouze jedné baterie

(d) Při poškození některé součásti, nutné nahradit stejným modelem

2. Podvozek může být upraven s určitými omezeními

(a) Nesmí se měnit vzdálenost mezi koly

(b) Konstrukce auta nesmí překročit rozměry 250 mm × 400 mm

(c) Povoleno vrtání a umisťování stojanů

3. Povoleno přidání externích obvodů (např. senzory)

4. Může být použit pouze jeden z následujících mikrokontrolérů[1] MPC5604B [2], K40X256

5. Je zakázáno bezdrátové připojení k vozidlu

6. Zakázán DC-DC měnič

7. Pro řízení motoru musí být použita jedna z následujících možností

(a) Freescale H-Bridge jako MC33931 or MC33932

(b) Diskrétní analogové komponenty

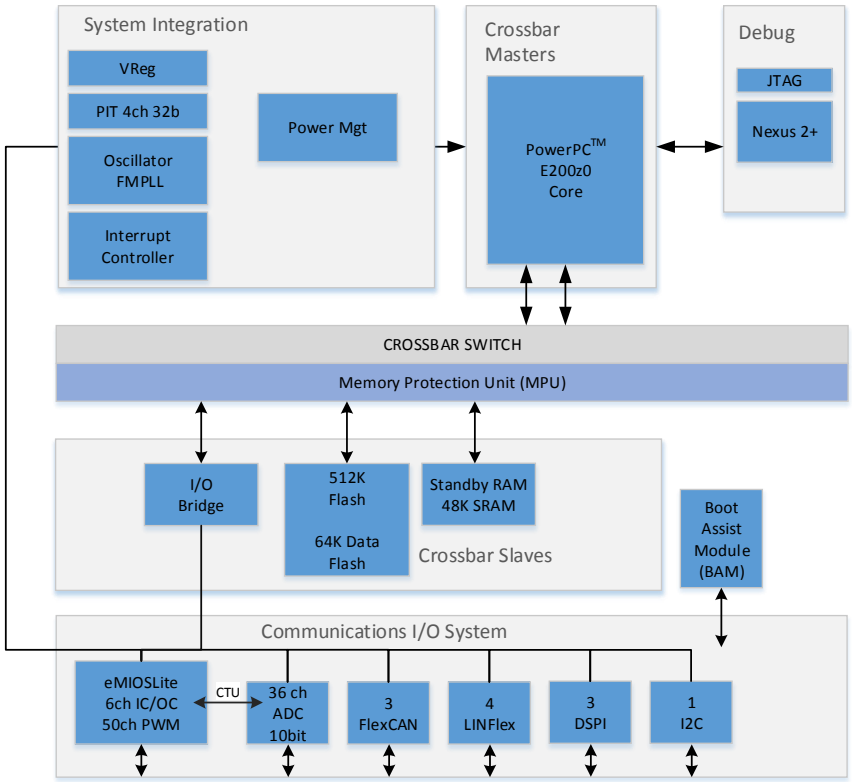
3 Seznámení s Hardwarem

3.1 Mikrokontrolér MPC5604b

Srdcem našeho vozidla je mikrokontrolér MPC5604b od firmy Freescale. Je to 32-bitový mikrokontrolér z rodiny Qoriwa, který se specializuje na automobilový průmysl, a z tohoto důvodu tomu odpovídají i jeho vlastnosti, což jsou např. sběrnice CAN či LIN.

[1]Mikrokontrolér je jednočipový mikropočítač, který obsahuje jádro procesoru, paměť a programovatelné vstupně/výstupní periférie (např. A/D, D/A převodníky, čítače, časovače a další).

Jádro mikrokontroléru PowerPC e200z0 dokáže pracovat na frekvenci 48–64 MHz, díky tomu dokážeme vytvořit periodické přerušení času (PIT[2]) přibližně každých 16 ns, což využijeme při získávání jednotlivých pixelů z lineární kamery, kde je nutné mít určitý minimální časový úsek pro získání správných hodnot (viz kap. 3.7). Další vlastnosti jsou 512 kB vestavěné Flash paměti, 64 kB EEPROM, 96 kB RAM. Základní blokové schéma můžeme vidět na obrázku níže (obr. 2).



Obrázek 2: Blokové schéma MPC5604b

[2]Periodic intterupt timer

3.1.1 Paměť RAM

RAM jsou považovány za nejrychlejší paměti [12]. Problém u těchto pamětí nastává v okamžiku, kdy dojde k odpojení napětí. Pokud nastane tato možnost, tak dojde ke ztrátě dat a po zapnutí mají nedefinovaný stav, neboli řečeno jsou volatilní. Paměti dělíme dále na DRAM a SRAM. U našeho mikrokontroléru se nachází 32 kB typu SRAM.

U statické paměti RAM je typická buňka tvořena klopnými obvody a skládá se z šesti MOSFET tranzistoru. Na rozdíl od DRAM není nutné data průběžně obnovovat a z tohoto důvodu jsou podstatně rychlejší. Objevují se vetšinou o velikosti do 1 MB. Využití najdou například tam, kde je nutná velká rychlost bez nutnosti vetší kapacity (např. vyrovnávací paměť).

3.1.2 Paměť FLASH

Jedná se o nevolatilní elektronickou paměť s libovolným přístupem [11]. Technologie je odvozena od pamětí typu EEPROM. Odlišnost je v morfologii paměťových buněk a propojovací síti mezi buňkami. Programování probíhá po jednotlivých slovech, mazat lze pouze celé bloky.

Dělí se podle technologie výroby. Přesněji se jedná o typ NAND a NOR. Každý má své výhody i nevýhody, a proto se hodí každá pro jiný účel. U NAND se využívá při úschově většího množství dat z důvodu rychlého zápisu a čtení dat. Nedostatek je ve složitém zápisu dat po bytech. Technologii NOR použijeme např. při uložení firmwaru, či programu mikrokontroléru. Výhodou je možnost zápisu po bytech. V našem případě se pro ukládání programu využívá FLASH paměť typu NOR o velikosti 512 kB.

3.1.3 eMIOS

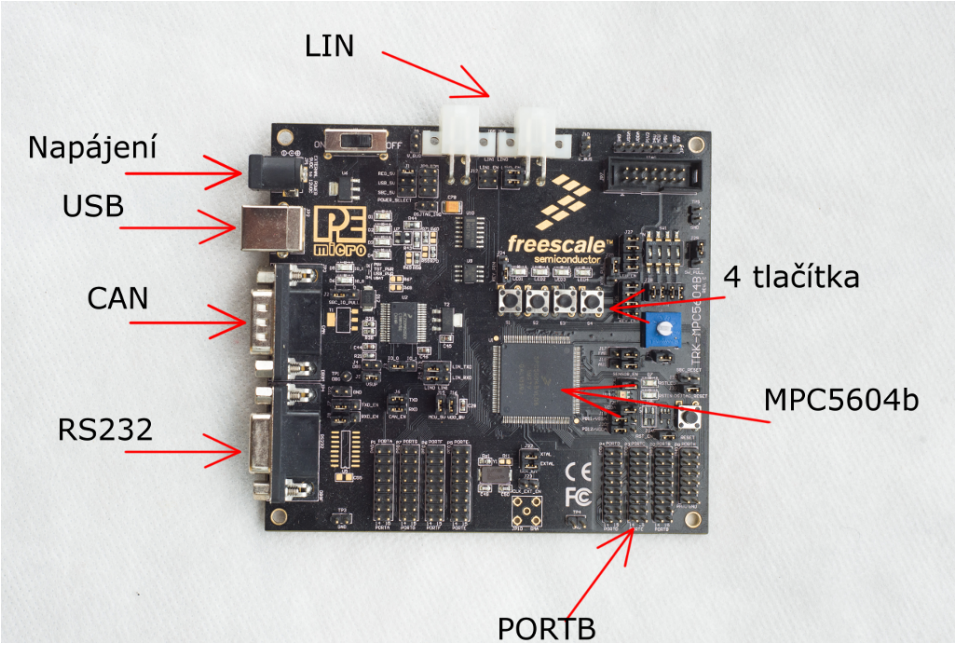
Abychom dostali auto do pohybu, je nutné ovládat nějakým způsobem motory. Dva stejnosměrné motory pro dopředný/zpětný pohyb a jeden servomotor pro možnost natáčení vozidla doleva či doprava a tím pádem sledování definované dráhy. Oba typy motorů ovládáme pomocí PWM[3], kde nastavujeme napětí na motoru pomocí střídy a tím určujeme požadovanou rychlost či natočení.

[3]Pulse Width Modulation

V tomto momentě přichází místo pro modul mikroprocesoru, který je znám pod zkratkou eMIOS[4]. S jeho pomocí můžeme jednoduše nastavit vlastnosti jednotlivých vstupně/výstupních portů, jestli chceme číst data nebo například využívat porty pro ovládání motorů pomocí PWM. eMIOS poskytuje funkce pro vytváření časových událostí. Používá časovače kanálu, kde každý kanál poskytuje podmnožinu funkcí dostupných v jednotném kanálu, při rozlišení 16 bitů.

3.2 Vývojový kit

Mikrokontrolér je součástí vývojového kitu TRK-MPC5604B (obr. 3), který využívá maximálně schopnosti mikrokontroléru, jako jsou už jednou zmiňované sběrnice CAN a LIN, dále tam najdeme sériovou linku RS232, SPI, LED, tlačítka a několik vstupně/výstupních programovatelných portů a další.

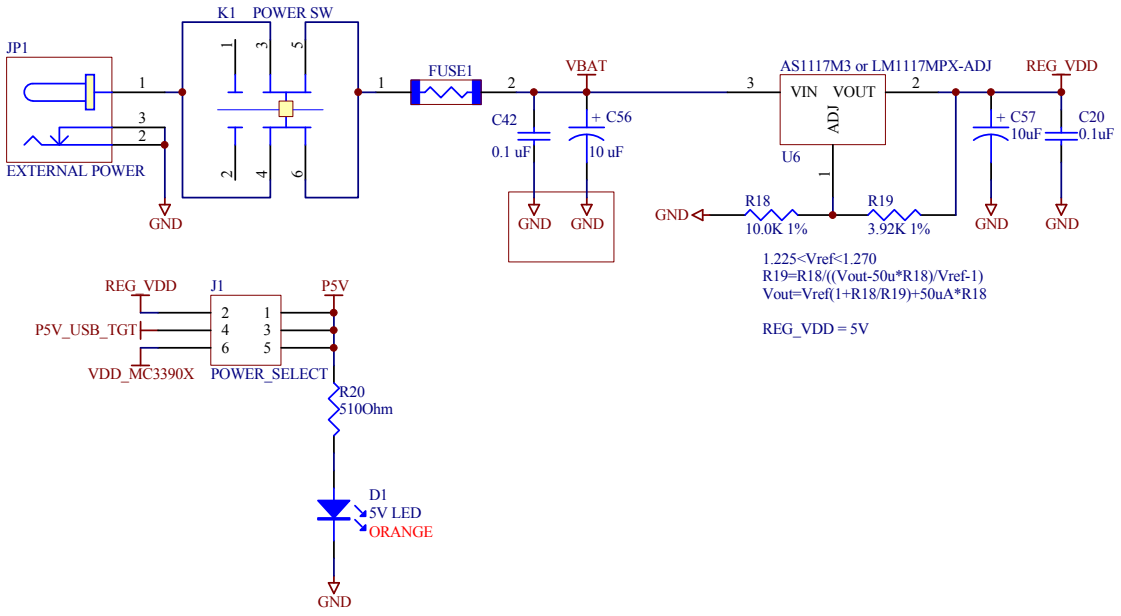


Obrázek 3: Vývojový kit TRK-MPC5604B.

Vývojový kit může být napájen z USB (5 V), což je taky jedna z možností, jak do mikroprocesoru nahrát námi vytvořený program. Další možností napájení je přivedení externího napájení(obr. 4) na konektor JP1 (2,5 mm × 5,5 mm). Rozmezí tohoto napětí je 9 až 12 V,

[4]enhanced Modular I/O Subsystem

které se následně pomocí stabilizátoru AS1117M3 stabilizuje na potřebných 5 V. Z jakého zdroje chceme vývojový kit napájet rozhodneme nastavením správného přepínače J1.



Obrázek 4: Napájení kitu TRK-MPC5604B [4]

3.2.1 CAN

Je datová sběrnice, využívající se dříve převážně v automobilovém průmyslu, postupem času se uplatnila i v průmyslové automatizaci. Důvodem pro použití této sběrnice je především vysoká spolehlivost, přenosová rychlost (až 1 Mbit/s) a odolnost při zhoršených podmínkách (teplota, rušení apod.).

Sběrnice je tvořena dvouvodičovým vedením, kde vodiče jsou označeny jako CAN-Low, CAN-High a paralelně k nim jsou přidány zakončovací odpory (120 Ω). Přenáší se dva logické stavy recesivní a dominantní. Dominantní stav reprezentuje log. 0, recesivní log. 1. Pokud bude rozdíl napětí CAN-L a CAN-H nenulové, dosáhneme dominantního stavu. Signálové vodiče jsou konstruovány, tak aby při CAN-H bylo napětí v rozmezí 3,5 až 5 V, u CAN-L 0 až 1, 5 V.

Protokol CAN je definován čtyřmi typy rámců, jsou to datový rámec (Date frame), žádost o rámec (Remote frame), chybový rámec (Error frame) a rámec přeplnění (Overload frame). V datovém rámci (viz tab. 1) je posílán obsah zprávy a její priorita. Pokud uzel zjistí, že

na sběrnici je vysílán dominantní bit, zatímco on vysílá recesivní, tak okamžitě přestane vysílat a umožní rychlejší doručení důležitější zprávy. Tím docílíme toho, že se např. airbagy vyřídí předněji než např. ve stejný okamžik vysílané ABS. Ostatní rámce mají podobnou strukturu jako popisovaný datový rámec. Žádost o rámec, žádá druhý uzel o vysílání datového rámce se shodným identifikátorem. Chybový rámec zjišťuje chybu v přijímaných bitech. Rámec přeplnění vysílá pokud potřebuje nějaký čas na zpracování předchozí zprávy.

START OF FRAME	ARBITRATION FIELD	CONTROL FIELD →
→ DATA FIELD	CRC FIELD	ACKNOWLEDGE FIELD →
→ END OF FRAME	INTERMISSION FIELD	

Tabulka 1: Formát datového rámce

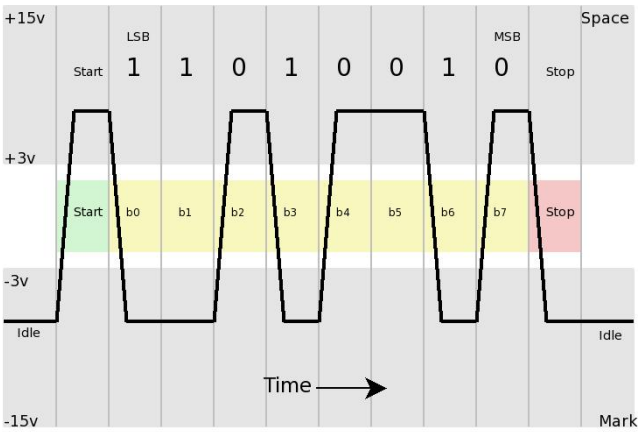
3.2.2 RS232

Je komunikační rozhraní využívající se ke komunikaci dvou zařízení [8]. Maximální vzdálenost vedení činí 15 m, při přenosové rychlosti 19200 bd [8]. Snížení přenosové rychlosti můžeme dosáhnout vzdálenosti až 900 m (při 2400 bd). Pro větší odolnost proti rušení se využívá většího napětí na propojovacích vodičích, než je standardních 5 V. Nejčastěji se využívá pro logické úrovně přenosové napětí +10 V (log. 0) a −10 V (log. 1). Přenos probíhá asynchronně, kde je pevně daná přenosová rychlost, která může být až 115200 bd.

Komunikace se odstartuje start bitem, pomocí něhož dokážeme zjistit, že začíná přenos. Následně se přenese 8 bitů, kde první bit je LSB a poslední MSB. Dříve byla varianta pro 7 bitů, aby se ušetřilo místo. Dnes se prakticky nevyužívá. Po přenesení námi zaslaných bitů, může být přenos doplněn o paritní bit a následně je ukončen stop bitem, který zajistí určitou prodlevu pro přijímač, aby mohl zpracovat přijatý byte. Příklad takovéto komunikace můžeme vidět na níže uvedeném obrázku (obr. 5).

3.3 Řídicí deska motorů

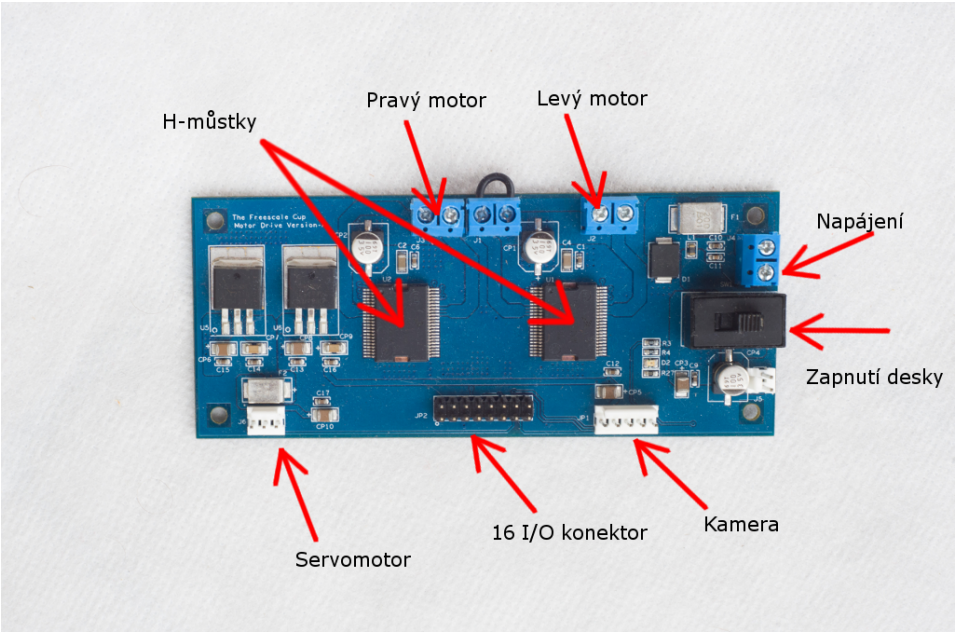
Řídicí deska (obr. 6) neslouží pouze k řízení motorů, ale využíváme ji také k připojení kamery a připojení hlavního zdroje napájení, což je v našem případě baterie o kapacitě 3000 mAh. Pokud se podíváme na pravidla o baterii (viz 1c), tak si můžeme všimnout, že jsme zde



Obrázek 5: Přenos bytu po RS232, ASCII 75 [8]

v rozporu s pravidly. Tato situace nastala, když jsem hledal doporučení na baterii a článek využíval jiná pravidla než byla v letošním roce. Pokud bychom se zúčastnili v příštím roce Freescale Cupu, tak by to nevadilo, jelikož naše auto momentálně odebírá proud přibližně 400 mA, což by zvládla i baterie v souladu s letošními pravidly.

Komunikace mezi mikrokontrolérem a řídicí deskou probíhá v našem případě pomocí 16-pinového vstupně/výstupního portu B.



Obrázek 6: Řídicí deska motorů

3.4 Řízení pomocí PWM

Pulzně šířková modulace je využívaná v robotice pro řízení motorů či serv. Pomocí použití vnitřních čítačů, mikrokontrolér moduluje střídu obdélníkového průběhu, které dovoluje kontrolovat množství energie vstupující do zařízení. Střída je udávána poměrem, který popisuje délku intervalu logické jedničky.

3.4.1 Konfigurace časovače

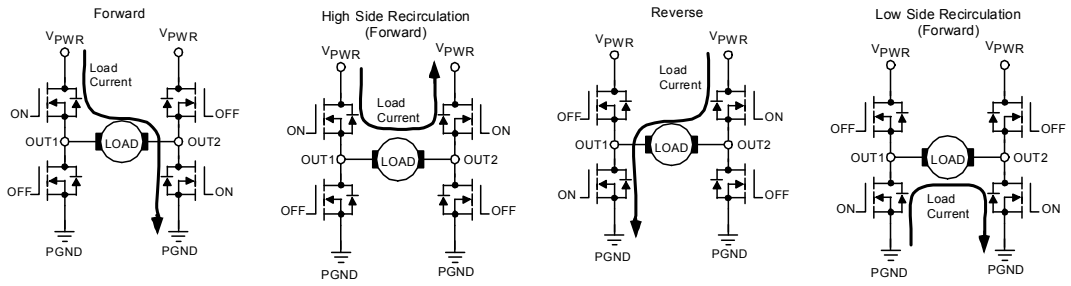
Generování PWM signálu je založeno na neustálém hardwarovém porovnáním hodnot registru s volně běžícího čítače, který čítá podle hodinového signálu. Pokud nastane shoda, může hardware vyvolat přerušení a zavolat obslužnou rutinu.

3.5 DC Motor

Je stroj převádějící elektrickou energii na mechanickou. Motor se skládá ze statoru a rotoru. Na statoru je navinuto budicí vinutí, produkující při průchodu stejnosměrného proudu podélné magnetické pole, proto je nutné motor vždy nabudit. Polarita hlavních a komutačních pólů se po obvodě střídá.

3.5.1 H-můstek

Obsahuje dvojici spínacích prvků, mezi kterými je umístěn motor. Příklad H-můstku je zobrazen na níže uvedeném obrázku (obr. 7). Spínače jsou sepnuty v párech, kde je vždy sepnut po jednom na každé straně. Pokud by byly aktivní dva přepínače na jedné straně můstku, došlo by ke spojení zdroje napětí na plusu a mínusu, což by vedlo ke zkratu.



Obrázek 7: Řízení směru motoru pomocí H-můstku [3]

Řízení směru otáčení motoru se určuje pomocí toku proudu, který se určí správným sepnutím spínačů. Tímto můžeme docílit změnu směru otáčení motoru. Na obrázku výše (obr. 7) jsou vidět typy směru proudu a jejich orientaci otáčení motoru.

Jako přepínače mohou být použity různé typy spínacích prvků, jako je např. relé, tranzistory, přepínače apod. V našem případě najdeme v zapojení čtyři MOSFET tranzistory, které jsou spínány řídicím napětím.

3.5.2 Ovládání motoru

Model obsahuje dva stejnosměrné motory, které jsou ovládány dvěma H-můstky (viz 3.5.1) MC33931, kde ovládání směru je podmíněno tabulkou níže (tab. 2). Řízení otáček se provádí pomocí PWM regulace, kterou nám generuje mikroprocesor.

Tabulka 2: Ovládání motoru

Device State	Input Conditions				Status	Outputs	
	$EN/\overline{D2}$	D1	IN1	IN2	$\overline{SF}$	OUT1	OUT2
Forward	H	L	H	L	H	H	L
Reverse	H	L	L	H	H	L	H
Freewheeling Low	H	L	L	L	H	L	L
Freewheeling High	H	L	H	H	H	H	H
Disable 1 (D1)	H	H	X	X	L	Z	Z
IN1 Disconnected	H	L	Z	X	H	H	X
IN2 Disconnected	H	L	X	Z	H	X	H
D1 Disconnected	H	Z	X	X	L	Z	Z
Under-voltage Lockout)	H	X	X	X	L	Z	Z
Over-temperature	H	X	X	X	L	Z	Z
Short-circuit	H	X	X	X	L	Z	Z
Sleep Mode $EN/\overline{D2}$	L	X	X	X	H	Z	Z
$EN/\overline{D2}$ Disconnected	Z	X	X	X	H	Z	Z

3.6 Servomotor

Je specializovaný stejnosměrný motor, u kterého je možné nastavit přesnou polohu natočení osy. Většina dnešních servomotorů je propojena třemi vodiči, kde dva z nich jsou napájení motoru a třetím vodičem se určuje otočení osy, tak je to i u našeho serva.

Ovládání se zde provádí stejně jako u stejnosměrného motoru, tedy PWM regulací, kde pokud bude střída 50 %, tak servo bude v základní poloze, snižování se bude natáčet postupně doleva, zvyšováním naopak doprava. Maximální úhel natočení ze základní polohy u našeho serva je 90°. Většina servomotorů je ovládána impulsy o frekvenci 5 Hz, kde délka impulsu určuje natočení, pro 1 ms 0°, pro 2 ms to je 180°.

3.7 Lineární kamera

Je založena na čipu TSL1401CL od firmy TAOS. Skládá se z čidla CMOS, jenž obsahuje 128 fotodiod uspořadaných v lineárním poli. To znamená, že řádková kamera nám poskytuje rozlišení o velikosti 1×128.

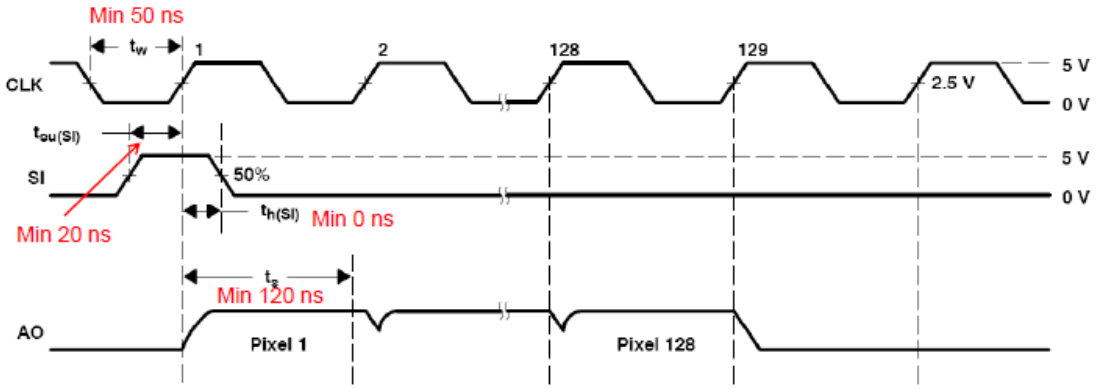
Pro získání dat lineární kamery využíváme tři signály, jsou označeny zkratkami CK (clock), SI (serial input to sensor) a AO (analog output). Dosažení správných hodnot na analogovém výstupu je nutné splnit pár podmínek, které můžeme vidět na časovém diagramu (obr. 8). Čtení dat zahájíme spuštěním signálu SI, který může být vytvořen např. pomocí GPIO[5]. V tomto okamžiku se zahájí výstupní cyklus kamery, kde každá z fotodiod je připojená k integrátoru a jejich přepínání nám provádí 128-bitový posuvný registr. Následně nám stačí generovat správný hodinový signál, abychom nepřesáhli maximální povolenou frekvenci, která u naší kamery činí 8 MHz. Poté už se nám na analogovém výstupu zobrazí požadovaná data. Teoretický čas pro získání všech 128 hodnot, můžeme zjistit z níže uvedeného vzorce. Po dosazení maximální možné frekvence (8 MHz), nám vyjde čas 33,75 μs.

Na výstupu je napětí v závislosti na osvětlení. Pokud není fotodioda osvětlena měli bychom mít na výstupu 0 V, při klasickém světle dostaneme kolem 2 V, pokud se dostaneme do saturace budeme mít 4,8 V. V saturaci je možné výstup ovlivnit pomocí změny integračního času integrátoru. Když budou na vstupu 4,8 V, tak víme, že je čip příliš osvětlován a je možné snížit integrační čas a tím zpřesnit námi získávané hodnoty. Naopak pokud

[5]General Purpose Input/Output

bude na výstupu nízké hodnoty, můžeme očekávat, že světelné podmínky nejsou dobré. Stačí prodloužit čas integrace, abychom získali věrohodnější výsledky.

$$T_{int(min)} = \frac{1}{F_{max}} \cdot (n - 18) + 20 \tag{1}$$

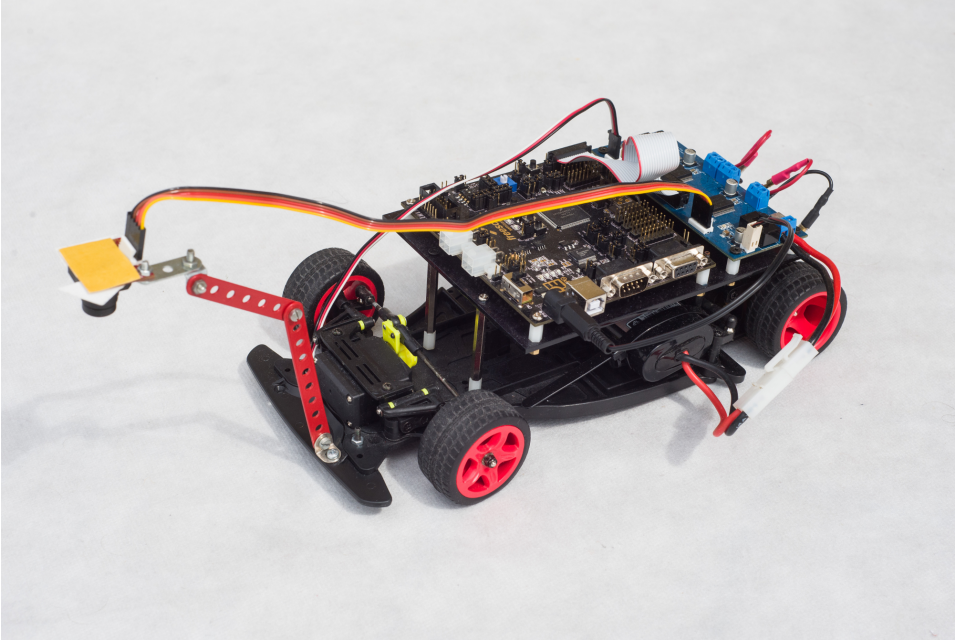


Obrázek 8: Časový diagram pro získání dat z kamery [5]

3.8 Sestavení modelu

Dodané součástky zakoupené u firmy Freescale (podvozek, vývojový kit, řídicí deska motorů, kamera a servo) bylo nutné upevnit nějakým způsobem k podvozku vozidla.

Rozhodli jsme se pro vytvoření plastové desky, kde jsou vyvrtány otvory pro správné uchycení kitu, řídicí desky, či pro protáhnutí kabelů. Námi vyrobená deska byla k podvozku připevněna pomocí distančních sloupků, které byly přilepeny dvousložkovým lepidlem. Servomotor bylo nejprve nutné složit a poté ho propojit s přední nápravou [6]. Řádkovou kameru jsme upevnili k přední části vozidla pomocí stavebnice Merkur, je proto možné jednoduše měnit polohu nebo natočení kamery. Můžeme tím docílit lepších výsledků z lineární kamery. Nakonec byla na podvozek připevněna baterie. Zajištěna byla stahovacími pásky, aby nedocházelo k pohybu baterie při jízdě auta. Složené vozidlo můžeme vidět na obrázku níže (obr. 9).



Obrázek 9: Složené vozidlo

4 Vývojové prostředí

Zdrojový kód naší aplikace je psán programovacím jazykem C. Jedná se v dnešní době o jeden z nejpoužívanějších programovacích jazyků. Využití najde především při psaní ovladačů OS nebo pro program našeho mikrokontroléru. Je přehlednější než dříve používaný assembler. Mnoho dnešních moderních programovacích jazyků přebírá syntaxi od tohoto jazyka.

Program je vyvíjen softwarem CodeWarrior od firmy Freescale. Přesněji se jedná o verzi v2.10, která je určená pro řady MPC55xx/MPC56xx. Při nahrávání programu máme na výběr ze dvou možností. A to buď nahrát program do paměti SRAM či FLASH. Při nahrávání do SRAM je nutné mít vývojový kit neustále pod napětím, jinak bychom o uložený program přišli (viz kap. 3.1.1). Proto ve většině případů jsme program ukládali do FLASH paměti.

4.1 Ladění aplikace

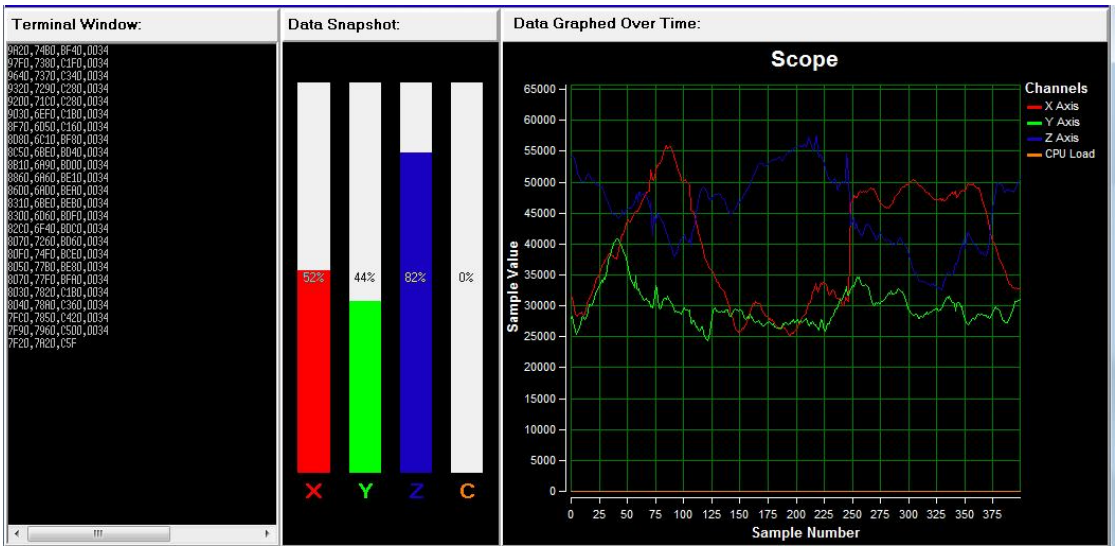
Výhodou našeho vývojového kitu je možnost ladit program při spuštění. To znamená možnost sledovat námi vytvořený program, jeho proměnné a různé registry mikrokontroléru za chodu aplikace. Můžeme tím dosáhnout daleko rychlejšího vývoje algoritmu, jelikož si můžeme ověřit, že se nacházíme na místě v programu tam, kde očekáváme. Není problém změnit

hodnoty našich proměnných v režimu ladění bez nutnosti překládání aplikace a její znovu nahrání.

4.2 Komunikační terminál

Při prvotním psaní programu jsme využívali hojně analytický software od firmy P&E Microcomputer Systems. Pomocí jejich programů bylo možné zobrazovat získávaná data na notebooku a pracovat s nimi, jak jsme uznali za vhodné. Např. pomocí SerialGrapheru (obr. 10) jsme mohli v reálném čase zobrazovat průběh, který nám zobrazuje řádková kamera, pokud jsme neměli zrovna po ruce osciloskop.

Terminál komunikuje přes virtuální RS232, která je vytvořena na našem vývojovém kitu, a posílá data přes USB rozhraní. Bylo sice možné zvolit, jestli využijeme virtuální port nebo klasickou fyzickou RS232. Jelikož náš notebook nedisponoval RS232 ani převodníkem USB-RS232, rozhodli jsme se využít právě téhle nabízené možnosti.

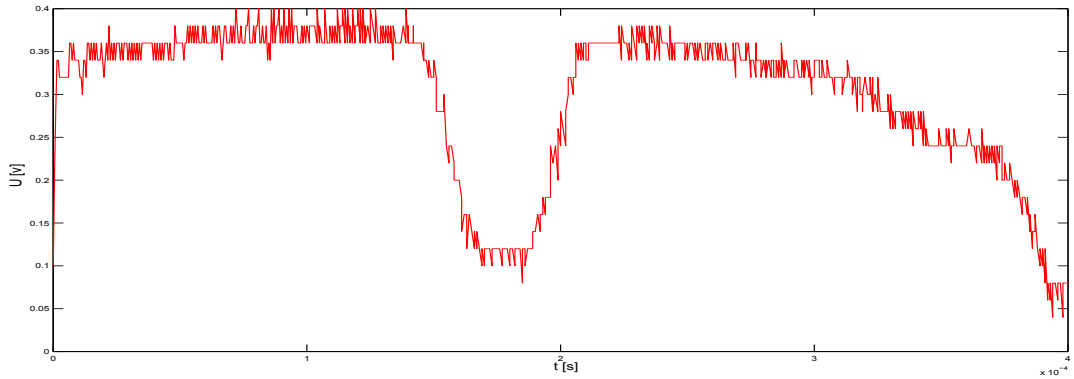


Obrázek 10: Ukázka aplikace SerialGrapher

5 Návrh řídicího algoritmu

Před návrhem správného algoritmu pro řízení našeho auta, jsme se museli první seznámit s ovládáním mikrokontroléru. Využili jsme ukázkové aplikace, která je dostupná z komunit-

ního fóra Freescalu. V aplikaci je ukázána práce s kamerou a motory. Pro motory stačilo pouze nastavit správně registry modulu eMIOS, abychom získali na výstupu požadovanou střídu PWM. Řádková kamera je řízena GPIO signály, kde jsme nastavovali signály CLK, SI a na analogovém vstupu mikrokontroléru jsme četli požadovaný signál. Data jsou ukládána do pole o počtu 128 prvků typu integer, jelikož naše kamera má rozlišení 1×128. Prvek pole obsahuje hodnotu v rozmezí 0 až 256, kde jsou hodnoty převedeny AD převodníkem z přijímaného signálu kamery. Nula reprezentuje nulové osvětlení, 256 naopak maximální.



Obrázek 11: Získané data z kamery

Abychom mohli přemýšlet nad vývojem programu, bylo nutné nejdříve zjistit, jaká data nám posílá kamera. K tomu jsme využili osciloskop. Připojili jsme tedy kameru k řídicí desce, která byla propojena s naším kitem. Následně stačilo přiložit sondu osciloskopu na analogový signál a triger jsme nastavili s náběžnou hranou signálu SI, který zahajoval přenos analogového signálu. Ze získaného průběhu (obr. 11) můžeme vidět, že při měření byla kamera nastavena přibližně na střed čáry. Mapu trati jsme měli složenou z vytištěných předloh zatáček či rovinek Freescalu.

Pokud se pozorněji podíváme na složené vozidlo z kap. 3.8 (obr. 9), tak si můžeme povšimnout, že máme stranu, kde není umístěn čip kamery, přelepenou páskou. Zjistili jsme totiž, že výstupní výsledky jsou ovlivněny také dopadajícím světlem právě na tuto část desky. Rozhodli jsme se proto zamezit ovlivnění výsledku, přelepením nějakým neprůsvitným materiálem.

5.1 Vývoj programu

Hned na začátku je nutno poznamenat, že jsme počítali s tím, že bude trať vždy dostatečně osvětlena, aby bylo možné rozlišit černou od bílé. Z tohoto důvodu jsme se nerozhodli pro osvětlení tratě přiděláním LED světel k přední části vozidla, která jsme mohli vidět u konkurenčních vozidel z loňských let. Při zjišťování zobrazovaných dat při různých osvětlení jsme usoudili, že by musela být trať umístěna v hodně špatně osvětlených podmínkách, aby nebylo možné sledovat trať.

5.1.1 Určení prahové hodnoty

U vývoje řídicího algoritmu jsme vycházeli z obrázku předchozí kapitoly (obr. 11). V prvé řadě bylo nutné určit, jaká oblast z analogového signálu je považovaná za černou barvu, nebo jestli ji můžeme považovat za bílou. Definovali jsme si proto proměnnou, která v sobě nesla hodnotu prahu. Ten určoval, jaký prvek bude brán za bílou nebo černou. Poté jsme převedli hodnoty do nového pole, které nabývalo pouze hodnot jedna a nula. Jednička nám reprezentovala tmu, nula světlo. Proměnná prahu byla nastavena při inicializaci a určovali jsme jí podle toho, jak hodně byla osvětlena námi testována trať.

5.1.2 Zjištění polohy čáry

Po úspěšném naprogramování vyhodnocení tmy nebo světla, bylo nutné určit ze 128 bitového pole, jaké hodnoty jsou správné a kde se nám zanesla určitá chyba. Myšlenka byla taková, že jsme se snažili najít největší místo počtu jedniček vedle sebe. Ze středu největšího počtu jedniček už bychom mohli jednoduše určit polohu sledované čáry. Proto tato část algoritmu byla nejdůležitějším ze všech bodů programu. Zjistili jsme proto maximální počet jedniček vedle sebe a to jsme určili jako výslednou polohu námi sledované čáry.

Jakmile jsme našli místo s nejvyšším počtem jedniček, tak jsme si uložili hodnotu levého a pravého indexu první a poslední jedničky. Zjištění polohy čáry už stačilo pouze zjistit průměr těch dvou hodnot. Výsledné umístění jsme zvolili jako absolutní hodnotu z 64, jelikož máme 128 rozlišení. Znamená to, že např. pro -64 je čára umístěná maximálně vlevo, 0 střed, 64 maximálně vpravo.

Když už jsme znali správnou polohu námi sledované čáry, nebyl už problém nastavit výslednou polohu serva. Definovali jsme si proto funkci, která nám přepočítala hodnotu polohy na potřebnou hodnotu pro natočení motoru ve směru čáry.

5.1.3 První jízda

Po navrhnutí první verze našeho algoritmu, jsme mohli vyzkoušet jeho funkčnost na dráze. Poprvé jsme testovali řídicí program bez zapnutých motorů, abychom viděli, jestli se servo natáčí ve směru posouvané čáry. Výsledky byly z našeho pohledu nad očekáváním, jelikož se motor otáčel přesně tak, jak jsme si představovali.

Rozhodli jsme se tedy pro zapnutí motorů nižší rychlostí a vpuštění na jednodušší trať (mírná zatáčka). Auto zvládlo jízdu na jedničku, proto jsme neváhali a postavili si ovál s menším poloměrem zatáčky než v předchozím případě. Když vozidlo přijelo k prvnímu oblouku tratě, tak se sice začal servomotor správně natáčet, ale po chvilce se otočil na druhou stranu a auto nám vyjelo z trati.

Zjišťování příčiny problému nám zabralo minimum času, jelikož jsme zjistili z ladicích nástrojů, že se na analogovém výstupu kamery objevovaly určité chyby. Ty se projevovali tím způsobem, že nám prahová hodnota vyhodnocovala hodnotu, kterou neměla. To mohlo být způsobeno např. trochu odlišným osvětlením než jsme měli nastavenou hodnotu pro práh. Zapříčinilo nám to, že jsme našli 4 jedničky na více místech v poli a nebylo možné určit, jaká hodnota je správná. Bylo proto nutné nějakým způsobem tuto chybu odstranit.

5.1.4 Křížení trati

Trať může být vedena i přes jiný pruh, tím může vzniknout křížení. Aby vozidlo pokračovalo stále ve stejném směru a neovlivnilo ho křížení, umístili jsme do programu podmínku. Ta zahrnuje pravidlo, že pokud bude počet jedniček nalezených vedle sebe větších jak 15, tak auto bude pokračovat ve směru, který měl nastavený v předchozím kroku.

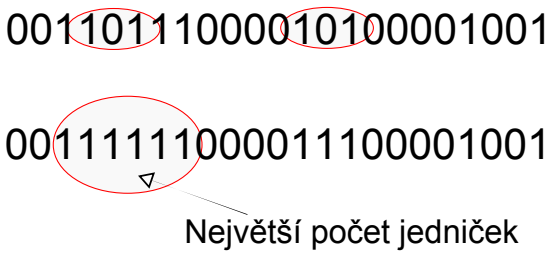
5.1.5 Eliminace chyby prahové hodnoty

Část problému jsme vyřešili způsobem, že jsme dali podmínku programu, že minimální počet jedniček v řadě za sebou musí být roven nebo větší aspoň hodnotě 8. Pokud tato podmínka

nebude splněna, tak se tato hodnota zahodí a vezme se předchozí hodnota. Vyřešíme tím také ten problému, že pokud auto začne sledovat zatáčku a kamera je umístěna o nějaký kousek dál, že nevidí čáru, i když jede správně, tak pojede stále stejným směrem.

Při testování navrženého algoritmu jsme zjistili, že nám výše uvedenou podmínkou program ani jednou neprojde. Bylo to způsobeno tím, že počet jedniček nikdy nepřesáhl vyšší číslo jak 5. Problémem bylo to, že se v řadě po sobě jdoucích jedniček, objevila náhodně nějaká nula a rozhodilo to vyhodnocení polohy.

Vyřešili jsme to tím, že jsme zjistili, jestli před a za nulou byla jednička, tak jsme vyhodnocovanou nulu změnili na jedničku. Ukázku zpracovaných dat můžeme vidět na níže uvedeném obrázku (obr. 12). Z obrázku je patrné, kde jsme hodnoty programově zaměnili. Předpokládáme, že přepsané nuly, jsou pouze výsledkem nějakého odrazu tratě.



Obrázek 12: Převod chybných bitů

Testování probíhalo na stejné trati, kde nám auto v předchozím případě vykolejilo. Sledování definované dráhy probíhalo správně a nedocházelo k žádným problémům. Základní verze řídicího algoritmu byla tedy hotová a mohli jsme se pustit do jeho zlepšení.

5.2 Vylepšení algoritmu

5.2.1 Automatické určení prahu

Abychom nemuseli pokaždé nastavovat hodnotu prahu v programu ručně a tím se zbytečně zdržovat, tak jsme měli v plánu to automaticky určit. První verze byla taková, že jsme vzali prostředních 10 hodnot pole s výsledky kamery. Následně jsme vypočítali jejich průměr, přičetli nějakou konstantu a nastavili jsme jej jako prahovou hodnotu. Toto řešení fungovalo pouze z poloviny. Už v předchozí kapitole jsme se zmiňovali o občasných chybách, které

se objevovaly z důsledku nerovnoměrného osvětlení dráhy. To způsobovalo i náš problém s průměrováním hodnot, jelikož se mohlo objevit velké číslo a narušit správné vyhodnocení.

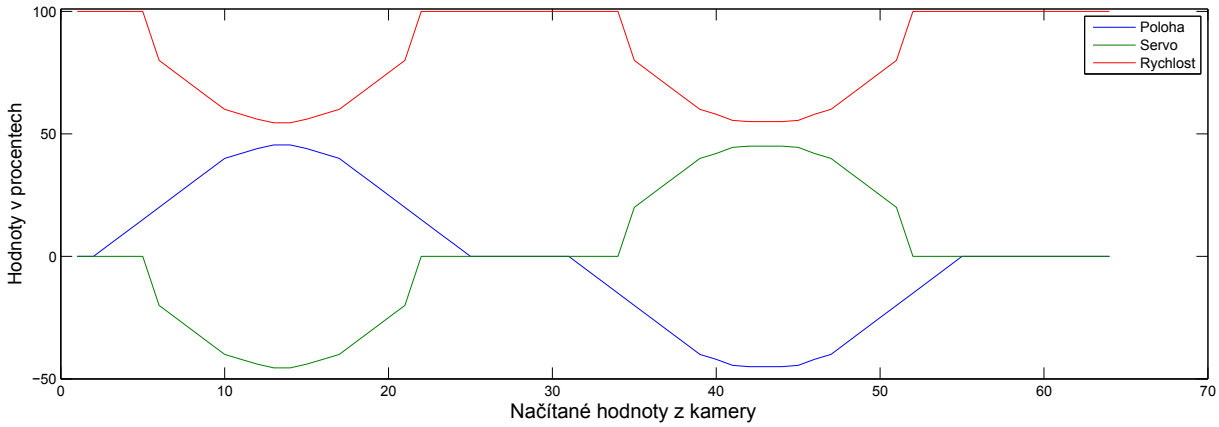
Druhá verze určení prahu byla už z větší části správně navržená. Tentokrát jsme neprůměrovali žádné hodnoty, ale vzali jsme nejmenší hodnotu ze všech prvků pole. Tím jsme docílili toho, že jsme vybrali nejtemnější místo ze snímaných fotodiod a dalo se očekávat, že to bude právě námi sledovaná dráha. Bylo nutné přičíst určitou hodnotu, jelikož se další vzorek může o něco málo lišit. Navrhli jsme si proto funkci, která spočítala, jak velkou hodnotu má přičíst k nalezené minimální hodnotě. Tato hodnota se lišila v závislosti na osvětlení. Pokud byly světelné podmínky více než dobré, bylo k prahu přičteno velké číslo. Je to z toho důvodu, že tam jsou větší rozdíly mezi světlem a tmou. Např. pro bílou dostaneme 256 a pro černou 150. Naopak při špatném osvětlení budeme mít u bílé 40, u černé 20. Z příkladů je vidět, že je nutné správné zvolit, kolik přičteme k nalezené minimální hodnotě, abychom se nedostali k blízkosti maximální hodnoty pole. Pokud by se tak stalo, řídicí algoritmus by nám nemohl fungovat.

Může se stát, že se zrovna při prvním získání vzorků z kamery objeví nějaká chyba, což může být například to, že bude hodnota nižší než by měla být. Následkem toho by bylo, že by se přičetlo, jiné číslo než je potřeba a hodnota prahu by nebyla zvolena správně. Tento problém není nijak vyřešen, ale je možné to obejít restartem procesoru. Zapříčiní nám to, že se po restartování najde znova minimální hodnota, která bude tentokrát správná. Ověřit si to můžeme posouvání vozidla mimo čáru a sledovat, jestli se servomotor natáčí očekávaným směrem. Pokud ne, můžeme se pokusit opět restartovat procesor.

5.3 Řízení rychlosti

Když nám vozidlo sledovalo úspěšně dráhu, mohli jsme začít experimentovat s proměnlivou rychlostí motorů. Náš princip řízení byl poměrně jednoduchý. Vypočítali jsme si polohu čáry, poté jsme nastavili servomotor a z úhlu natočení vypočítali rychlost motoru. Znamená to, že pro úhel 0° jsme měli nastavenou stoprocentní rychlost, s postupnou změnou natočení, jsme od základní hodnoty odečítali určitou hodnotu. Snáze to pochopíme z vyobrazeného grafu níže (obr. 13). Jedná se pouze o ilustrativní graf, který byl ručně vytvořený v Excelu.

Při pohledu na graf můžeme vidět, že servo se nezačne natáčet jakmile se poloha trochu



Obrázek 13: Graf závislosti na poloze

vychýlí. Zvolili jsme to z toho důvod, jelikož čára má určitou šířku, a auto by se cukalo při sebemenší změně polohy. Proto bereme střed v určitém rozmezí, u nás to je 54 až 74.

5.4 Aplikace filtru

Náš servomotor je možné nastavovat každých 20 ms. Za tuto dobu získáme několik vzorků z kamery. Abychom nebrali pouze poslední vzorek, rozhodli jsme se pro aplikování Kalmanova filtru. Důvod byl ten, že jsme ho využili v bakalářském projektu [13], kde se nám velice osvědčil, takže nebyl problém s naprogramováním. Kalmanův filtr se využívá v reálném čase, protože je velice rychlý, a neprojevuje se tam větší zpoždění, které by bylo v našem případě nežádoucí.

5.5 Integrační čas

V popisu řádkové kamery jsme se zmiňovali, že je možné ovlivnit výstupní hodnoty dobou integračního času čipu (viz kap 3.7). Řekli jsme si proto, že toho využijeme, abychom dosáhli optimálních výsledků v co nejrychlejším čase. V místě kde vyhledáváme nejnižšího hodnotu pro práh, jsme přidali podmínku, že pokud bude minimální hodnota v poli menší než 30 dekadicky, tak zvýšíme integrační čas. Docílíme toho, že pokud bude trať špatně osvětlena, tak vyčkáme delší čas pro zvýšení hodnot na analogového výstupu. Rychlost zpracování bude menší, ale omezíme riziko, že bychom dostávali špatné výsledky. Naopak při velkých

hodnotách na výstupu můžeme integrační čas snížit a docílit rychlejšího zpracování dat, při zachování správnosti měření.

5.6 Využití tlačítek

Abychom nemuseli restartovat mikrokontrolér při chybném vyhodnocení prahu, rozhodli jsme se využít tlačítka, která jsou dostupná na našem vývojovém kitu. Definovali jsme si proto pro ně určité funkce. Při stisku prvního tlačítka nám provede opětovné najití minimální hodnoty ze získaných dat kamery a přenastavení prahu. Pokud i poté není hodnota optimální, tak můžeme použít další dvě tlačítka. U nich je nastaveno, že pokud budou stisknuta, tak se inkrementuje, či dekrementuje hodnota prahu. Je možné, tak dosáhnout optimálního nastavení prahové proměnné.

Závěr

Na konci práce byl vyvinut algoritmus, který byl schopen řídit námi složené vozidlo po definované dráze, čímž jsme splnili podmínky zadání. Při návrhu jsme se neobešli bez problémů, hlavně co se týče stránky softwaru. Největší z nich byl zanesení určité chyby, či šumu výstupu kamery. Každou změnu byť sebemenší, kterou jsme provedli v programu bylo nutné otestovat, jelikož by bylo možné, že bychom se dostali situace, kdy bychom už se jen těžko vraceli k funkčnímu algoritmu.

Samotné ověření síly našeho programu mělo proběhnout mezi třemi týmy vytvořenými místními studenty. Jelikož žádný z dalších týmu nevytvořil řídicí algoritmus schopný ovládat auto před odevzdáním naší práce, nemohli jsme si ověřit, že náš návrh šel správným směrem. Můžeme pouze uvažovat, že jsme postupovali dobře, jelikož nám vozidlo zajelo neznámou trať bez větších problémů.

Na konečném algoritmu je ještě hodně věcí, které je možné vylepšit. Byla vytvořena základní kostra programu, která je schopná ovládat vozidlo. Největší cílem pro další rok je zlepšit a zrychlit ovládaní rychlosti motoru. V momentální fázi se servomotor ovládá pro každou rychlost stejně, což je samozřejmě chybně, jelikož se dostane ke vzorku pokaždé v jinou dobu. Dalo by se to vyřešit např. přidáním senzoru otáček. Z umístění výšky kamery by bylo poté možné zjistit, kdy přesně se dostanou přední kola ke snímanému vzorku a tím regulovat čas ovládání serva. V letošním roce Freescale Cupu se objevil nový objekt a tím byl tunel. U tohoto případu nám kamera nijak nepomůže. Bylo by proto nutné vybavit vozidlo nějakými postranními senzory vzdálenosti, či osvětlovacími LED.

Literatura

[1] Freescale Semiconductor: *The Freescale Cup 2012-2013 Season Rules.*[online]. rev. 1. 6. 2012 [cit. 19. 12. 2012] Dostupné z: <https://community.freescale.com/servlet/JiveServlet/download/93225-5-249658/TFCGlobalRules2013Final.pdf>

[2] FREESCALE. *Datasheet mikrokontroléru MPC5604B* [online]. rev 11/2012 [citováno 20. 12. 2012]. Dostupné z: <http://www.freescale.com/files/32bit/doc/ref_manual/MCF51JM128RM.pdf>

[3] FREESCALE. *Datasheet H-můstku MC33931* [online]. Rev. 4.0, 10/2012 [citováno 15. 05. 2013]. Dostupné z: <http://www.freescale.com/files/analog/doc/data_sheet/MC33931.pdf>

[4] Freescale Semiconductor: *Schéma vývojového kitu TRK-MPC5604B*[online]. [cit. 23. 12. 2012] Dostupné z: <http://cache.freescale.com/files/microcontrollers/hardware_tools/schematics/TRKMPC5604BSCH.pdf>

[5] TAOS: *Datasheet TSL1401*[online]. [cit. 23. 12. 2012] Dostupné z: <www.w-r-e.de/robotik/data/tsl1401.pdf>

[6] Servo and Steering Assembly Directions[online]. [cit. 25. 04. 2013] Dostupné z: <https://community.freescale.com/docs/DOC-1015>

[7] The Race Track - Freescale Cup Mexico 2009[online]. [cit. 01. 05. 2013] Dostupné z: <http://www.flickr.com/photos/freescale/4541008332/>

[8] HW server představuje - Sériová linka RS-232[online]. [cit. 01. 05. 2013] Dostupné z: <http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html>

[9] Aplikování sběrnice CAN[online]. [cit. 01. 05. 2013] Dostupné z: <http://www.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html>

[10] Qorivva MPC560xB Home Page[online]. [cit. 15. 05. 2013] Dostupné z: <https://community.freescale.com/docs/DOC-1283>

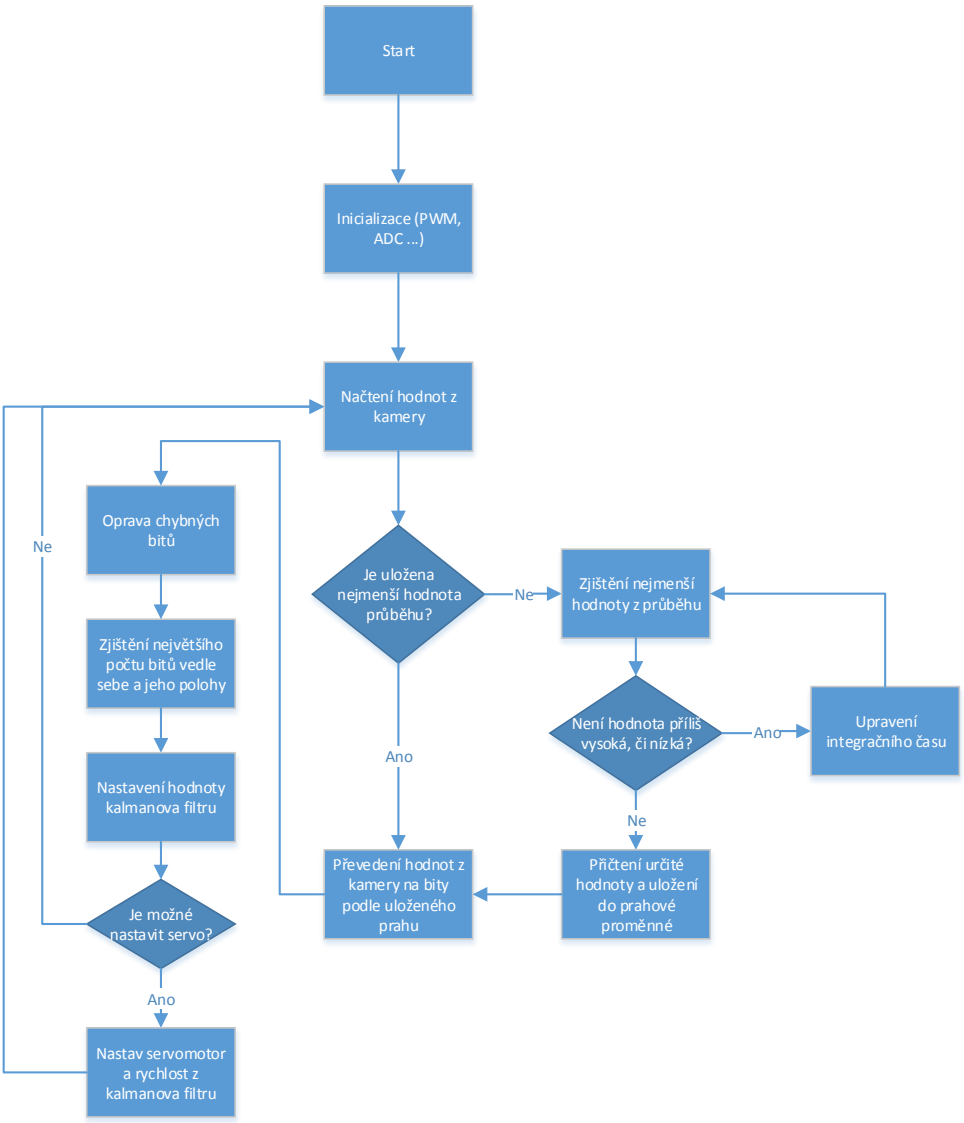
[11] Technologie flash pamětí a způsoby jejich využití[online]. [cit. 15. 05. 2013] Dostupné z: <http://www.root.cz/clanky/technologie-flash-pameti-a-zpusoby-jejich-vyuziti/>

[12] Statické a dynamické paměti[online]. [cit. 15. 05. 2013] Dostupné z: <http://www.root.cz/clanky/staticke-a-dynamicke-pameti/>

[13] Procházka, F. - Hofman, L.: Samořízené auto na autodráhu. Bakalářský projekt, MTI FM TUL, 2012. [cit. 15. 05. 2013]

A Přílohy

A.1 Vývojový diagram



Obrázek 14: Vývojový diagram

A.2 Obsah přiloženého CD

- Seznam adresářů
  - **Bakalářská práce:** Práce ve formátu .pdf.
  - **Zdrojový kód:** Vyvíjený program v prostředí CW.