TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automaticky řízený model vozu s přísavným systémem

Bakalářská práce

M10000175

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Petr Najman**
Vedoucí práce: Ing. Jan Koprnický, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Automatically controlled car model with suction system

Bachelor thesis

M10000175

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Petr Najman**
Supervisor: Ing. Jan Koprnický, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Najman**
Osobní číslo:	**M10000175**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Automaticky řízený model vozu s přísavným systémem**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s elektromechanickými částmi podvozku a řídicí jednotkou.
2. Zanalyzujte energetické požadavky přísavného systému pro model vozu.
3. Upravte hw systému, navrhněte algoritmus řízení a vytvořte odpovídající software pro řídicí jednotku modelu vozu.
4. Funkční model vyzkoušejte na laboratorní testovací dráze.

Rozsah grafických prací: **dle potřeby dokumentace**

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

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

Seznam odborné literatury:

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

[2] **Vít, Jakub. Samořízené elektrické auto. Liberec, 2012. Bakalářská práce. Technická univerzita v Liberci. Vedoucí práce Jan Koprnický.**

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

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

Technická univerzita v Liberci - 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen 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 samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14.5.2015

Podpis:

Abstrakt

Tato práce popisuje analýzu a postup pro vytvoření automaticky řízeného modelu auta s přísavným systémem, který umožňuje modelu jezdit po vertikálních plochách. Pro práci byl zakoupen model vozu ovládaný přes infračervené (IR) dálkové ovládání, jenž takovýto pohyb umožňoval. Tento model byl otestován a bylo analyzováno jeho technické vybavení. K vytvoření automaticky řízeného vozu byla instalována řídicí jednotka od firmy Arduino, která splňuje všechna kritéria daná rozměrem vozidla, jeho hmotností, výkonem přísavného systému a schopností ovládat tři stejnosměrné motory. Tyto parametry nejlépe splňovala řídicí jednotka Arduino Fio s procesorem Atmel ATmega328P. Řídicí jednotka byla doplněna o dvojitý H-můstek pro obousměrné řízení motorů. Na zprovozněném modelu bylo testováno několik senzorů umožňujících autonomní řízení. Po tomto testování byl vybrán optický senzor CNY-70. Pro takto osazený model byl napsaný algoritmus, psaný v prostředí Arduino IDE, jazykem vycházejícím z jazyka C/C++. Algoritmy využívají instalovaného optického senzoru, a díky tomu se vozidlo dokáže chovat autonomně.

Klíčová slova: jazyk C/C++, přísavný systém, H-můstek, Arduino, Atmel ATmega328P, CNY70

Abstract

This bachelor thesis describes the analysis and procedure of creation of an automatically driven model car with a suction system. This system enables the model to ride on vertical surfaces. An infrared operated car model that comply with the parameters was bought for this project. This model was tested and its technical equipment was analysed. To build automatically controlled car model a CPU from Arduino had to be used, as it fulfils every criteria given by the size of the car, its weight, the performance of the suction system and the ability to control three DC motors. The Arduino Fio unit with the Atmel ATmega328P processor matched these requirements. CPU was complemented by double H-bridge that enables bidirectional operating of motors. Couple of sensors, that enabls autonomous operation, were tested on the model in service. The optical sensor CNY-70 was chosen after this test. For a model of this construction an algorithm in Arduino IDE had to be compiled and written in C/C++. The algorithms use the installed optical sensor thanks to whitch it is capable of autonomous operation.

Key words: language C/C++, sunction system, H-bridge, Arduino, Atmel ATmega328P, CNY70

Poděkování

Tímto děkuji všem, kteří mi byli nápomocni při vypracovávání této bakalářské práce. Velké poděkování patří vedoucímu bakalářské práce Ing. Janu Koprnickému, Ph.D, který mi dal užitečné rady a obohatil mou práci o jeho cenné zkušenosti. A také bych chtěl poděkovat mé rodině a kamarádům, kteří mi při psaní této bakalářské práce a také po celou dobu studia vytvářeli příjemné prostředí a podporovali mě.

Obsah

Seznam obrázků

Seznam zkratek

PWM	Puzlně šířková modulace
HW	Hardware
SW	Software
USB	Universal Serial Bus (univerzální sériová sběrnice)
IDE	Integrated Development Environment (vývojové prostředí)
IR	Infračervené záření

1 Úvod

Hlavním cílem této bakalářské práce je sestrojení a naprogramování modelu vozidla s využitím přísavného systému. Tento model bude schopný jezdit po rovných horizontálních i vertikálních plochách. Přísavný systém je realizován vysokootáčkovým ventilátorem, který vytváří podtlak na ploše podvozku modelu.

Jako základ slouží podvozek osazený dvěma stejnosměrnými motory, realizující pohyb modelu, a jedním výkonnějším stejnosměrným motorem, který zprostředkovává již zmíněný přísavný systém [1]. Tento podvozek bude doplněn o řídicí jednotku, schopnou řídit všechny tři stejnosměrné motory podle potřeby, baterii, dostatečně silnou pro napájení všech částí, a další součástky důležité pro správné fungování a řízení všech částí modelu. Při vybírání těchto součástí je důležité brát na vědomí jejich velikost a hlavně jejich hmotnost, protože model by se měl pohybovat po vertikální ploše, a to záleží hlavně na jeho celkové hmotnosti. Výkon přítlačného systému v tomto modelu je značně omezený, takže je nutné udržovat co nejmenší celkovou hmotnost vozidla.

K příslušnému technickému vybavení vozidla bude následně vytvořen algoritmus, díky kterému se model bude pohybovat. Vozidlo bude moci opisovat naprogramované tvary (jezdit v kruzích apod.), přídadně se řídit pomocí informací dodaných z přidaných čidel nebo bude navrhnut systém dálkového ovládání (WiFi, bluetooth apod.). Takto sestavený a funkční model bude testován v různých podmínkách (např. různé druhy povrchů, rozdílné povětrnostní podmínky atd.), následně se může zvážit jeho rozšíření a vylepšení.

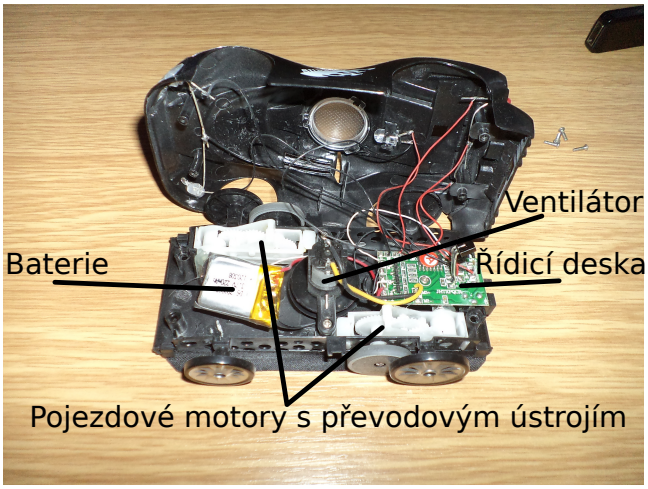
Struktura práce začíná rozborem koupeného podvozku, jeho nároky na napájení a možnosti jeho ovládání. Jako další krok práce následuje teoretický rozbor problému a zamyšlení se nad stylem řízení. Z tohoto vychází výběr hardwarových součástí pro sestrojení autonomního vozidla. Pro sestavené a pohybu schopné vozidlo jsou napsány testovací a řídící algoritmy. Takto softwarově a hardwarově vybavené vozidlo je testováno. Závěrem se řeší možné druhy rozšíření.

2 Teorie

2.1 Model vozidla

2.1.1 Seznámení s modelem

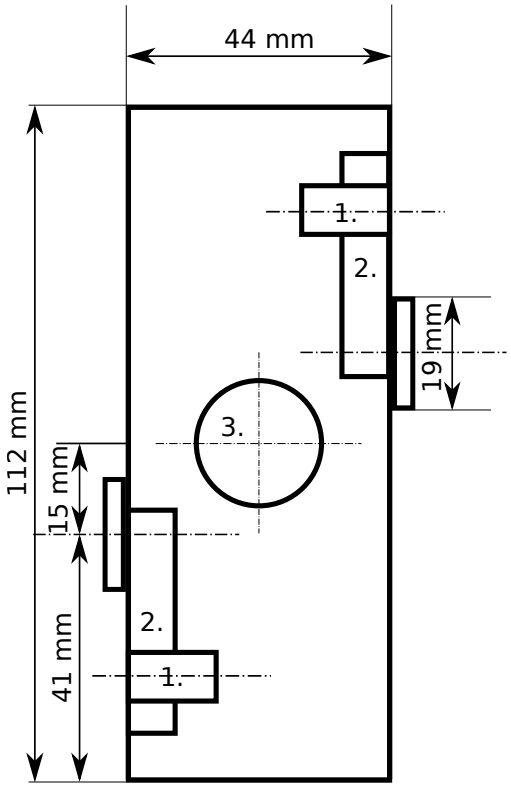
Jako základ mi posloužil model vozidla na dálkové ovládání schopného jezdit po horizontálních i vertikálních plochách (viz obrázek 2.1). Takovýto model se dá bez problému objednat z internetu [1]. Balení obsahuje samotný model a infračervené (IR) dálkové ovládání, jenž slouží jako úložiště energie (to je realizováno 6 AA bateriemi). Ovladač zároveň slouží i jako dokovací stanice pro dobíjení integrované Lithium-polymerové baterie vozidla. Samotné dobíjení probíhá přes konektor umístěný v zadní části modelu. Uvnitř vozu se nachází malá řídicí jednotka (zelený plošný spoj v pravé části vozu, viz obrázek 2.1), která přijímá IR signály z dálkového ovládání, tři stejnosměrné motory (dva po stranách a jeden uprostřed), malá Li-polymerová baterie s parametry 3,7 V a 200 mAh (levá část podvozku, viz obrázek 2.1), jež je schopná pohánět model vozidla 7–8 minut na vertikální ploše (testovaný údaj). Takto vybavený model má hmotnost 53 g. Testováním bylo zjištěno, že model je schopný se udržet (nepohybovat se) na vertikální ploše do vlastní hmotnosti 130 g.



Obrázek 2.1: Zakoupený model vozidla s přísavným systémem

2.1.2 Konstrukce modelu vozidla

Hlavní částí modelu je podvozek. Z části, které se dotýká plochy, po které má jezdit, je naprosto rovný a hladký, až na jeho středovou část, kde se nachází otvor pro ventilátor. Tento ventilátor je poháněn stejnosměrným motorem, jenž pracuje na napětí v rozmezí 1,2–6 V a vysává z pod podvozku vzduch, kde díky tomu dochází k podtlaku, a podvozek se tak udrží na vertikální ploše. Na každé straně podvozku se dále nachází další stejnosměrný motor se svým převodovým ústrojím. Tyto motory, navzdory své malé velikosti, dosahují až 20 000 otáček za minutu bez zátěže (katalogový údaj [2]), převodové ústrojí zde proto slouží ke snížení otáček na přijatelnou hodnotu. Tyto motory se starají o vlastní pohyb modelu, jejich umístění na podvozku je asymetrické (viz obrázek 2.2). Samotný podvozek s motory váží 28 g a má rozměry 112×44 mm.



Obrázek 2.2: Rozložení podvozku – 1. stejnosměrné motory, 2. převodové ústrojí, 3. ventilátor

2.2 Základní fyzika a využití podtlaku

Podtlak je záporný rozdíl mezi daným tlakem (nejčastěji v odděleném prostoru) a referenčním tlakem (atmosferický tlak). Podtlaku lze dosáhnout přesunem části hmoty (vzduchu) z místa, kde chceme podtlak vytvořit. Pro přesun je v našem

případě použit vysokootáčkový ventilátor. Podtlaku se využívá v praxi u mnoha případů, např. u sacích pump, u vysávání, v průmyslu – přísavné systémy pro manipulaci s tabulovým sklem nebo plechem atd. Využití tohoto jevu můžeme vidět i v přírodě u některých živočichů, např. chobotnic nebo gekonů.

Při základním uvažování pro výpočty fyzikálních sil, které musí působit na předmět (v našem případě model vozu), aby se dokázal udržet na vertikální rovině, jsem si vzal příklad z pohybu tělesa po nakloněné rovině [3]. S aplikací nakloněné roviny se setkáváme v mnoha praktických situacích, a proto je nutné správně popsat síly, které na takové těleso působí. V našem případě půjde trošku o extrémní úlohu, protože budeme uvažovat nakloněnou rovinu, jenž svírá s vodorovnou rovinou úhel $\alpha = 90^o$ v homogenním tíhovém poli. Na této nakloněné rovině tedy působí tíhová síla $\vec{F_G}$ (v našem případě $F_G = 0{,}5886\,\mathrm{N}$, toto je vypočítáno z hmotnosti vozu o 60 g). Na totéž těleso působí i reakční síla $\vec{F_R}$ nakloněné roviny, tato síla je kolmá na nakloněnou rovinu. Tíhovou silou působí na těleso Země a reakční silou působí nakloněná rovina. Pohybovou sílu $\vec{F_p}$, která uvádí těleso na nakloněné rovině do pohybu, můžeme získat také tak, že tíhovou sílu $\vec{F_G}$ rozložíme na dvě navzájem kolmé složky: normálovou sílu $\vec{F_n}$, která je kolmá k podložce, a pohybovou sílu $\vec{F_p}$, která je rovnoběžná s nakloněnou rovinou. Velikost sil $\vec{F_n}$ a $\vec{F_p}$ můžeme vyjádřit pomocí goniometrických funkcí k úhlu α. Tady se opět dostáváme k extrému mého problému, uvedu na dosazení :

$$F_p = F_G \cdot \sin\alpha \tag{2.1}$$

$$F_p = 0,5886 \cdot 1 = 0,5886\,\mathrm{N} \tag{2.2}$$

$$F_n = F_G \cdot \cos\alpha \tag{2.3}$$

$$F_n = 0,5886 \cdot 0 = 0\,\mathrm{N}. \tag{2.4}$$

Tady vidíme, že normálová síla $\vec{F_n}$ se rovná nule a pohybová síla $\vec{F_p} = \vec{F_G}$. Z toho vyplývá, že aby se naše vozidlo udrželo ve vertikální poloze, musí být normálová síla $\vec{F_n}$ minimálně rovna tíhové síle $\vec{F_G}$. Když víme, jaká je minimální síla pro udržení vozidla, budeme ještě potřebovat zjistit minimální sílu, která musí být vynaložena, aby se mohlo vozidlo začít pohybovat. V tomto případě musíme uvažovat ještě také se sílou smykového tření $\vec{F_t}$, která působí mezi tělesem a nakloněnou rovinou. Třecí síla působí vždy proti směru pohybu vozidla a leží ve styčných plochách obou těles. Velikost třecí síly lze psát ve tvaru

$$\vec{F_t} = f \cdot \vec{F_n}, \tag{2.5}$$

$$F_t = 0,87 \cdot 0,5886 = 0,512082\,\mathrm{N}, \tag{2.6}$$

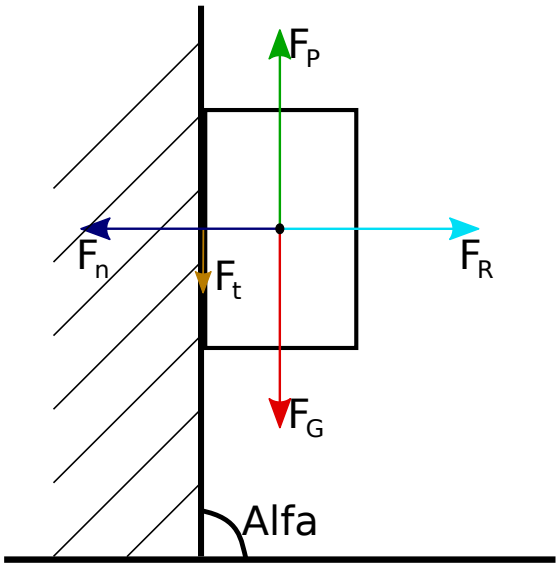
kde f, v našem případě budeme uvažovat, že $f = 0,87$ (součinitel smykového tření mezi sklem a gumou), je součinitel smykového tření mezi tělesem a nakloněnou rovinou. Vztah

$$\vec{F_p} - \vec{F_n} = m \cdot \vec{a} \tag{2.7}$$

je analogický ke vztahu

$$\vec{F_p} = m \cdot \vec{a}, \tag{2.8}$$

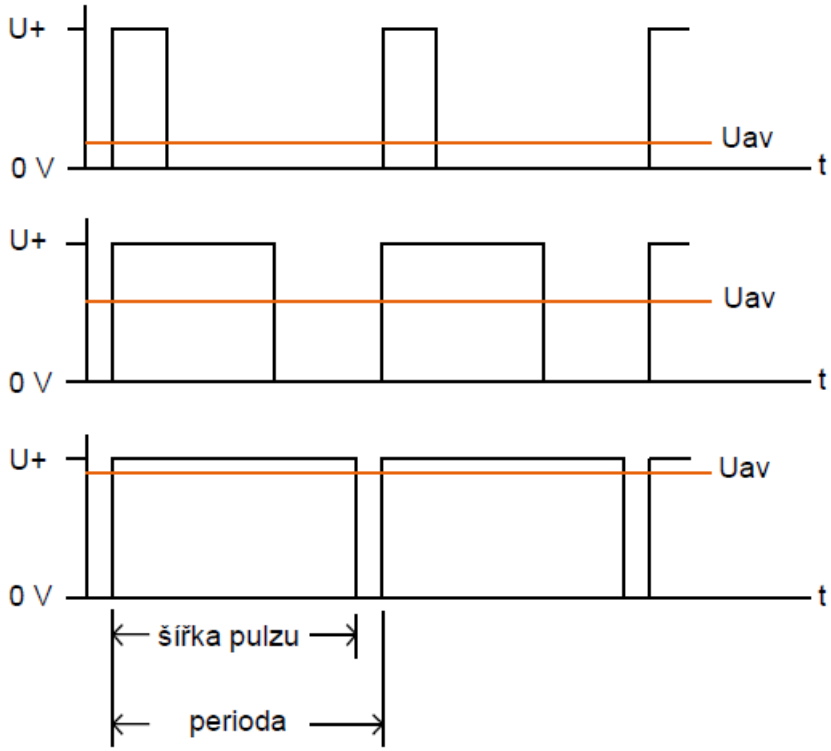
který platil pro případ zanedbatelného tření ($\vec{a} = 9,81\,m/s^2$). Po dosazení nám vyplývá $m = 0{,}0078\,\text{kg}$. Přítlačný systém musí vyvinout takovou sílu, aby zvládnul vytvořit dostatečnou sílu $\vec{F_n}$ a překonal ještě sílu smykového tření $\vec{F_t}$. Názorná ukázka všech sil je na obrázku 2.3. Z těchto výsledků můžeme vyvodit, že náš model vozu by se měl udržet ve vertikální poloze, protože se vypočtené hodnoty ani nepřiblížily otestované hmotnosti 130 g ($F_n = 1{,}275\,\text{N}$), jež je hraniční hmotnost pro udržení se ve vertikální pozici tohoto modelu. To je v našem případě splněno, protože vozidlo vážící 60 g musí, při vertikálním pohybu na skle, překonat působící síly které se rovnají 0,665 N.



Obrázek 2.3: Síly působící na vozidlo na vertikální ploše

2.3 PWM

V kapitole 2.5 se dozvíme, že budeme potřebovat řídicí jednotku s výstupy umožňující PWM [10]. Otáčky stejnosměrných motorů se řídí velikostí napětí, právě proto se využívá této modulace. PWM je zkratka z anglického pojmu Pulse With Modulation, česky pulzně šířková modulace. Je to metoda k dosažení analogového výstupu s digitálními prostředky. Digitální ovládání je využito k vytvoření obdélníkového signálu. To je dosaženo tak, že se výstup přepíná mezi zapnutým stavem a stavem vypnutým. Tento postup může simulovat napětí mezi plným zapnutím (v našem případě 3,3 V) a vypnutím (0 V). Změnou času na výstupu určujeme, kolik času stráví v zapnutém režimu oproti režimu vypnutému. Poměr těchto časů se nazývá *střída*. Perioda v PWM je čas, kdy dojde k přenesení jedné střídy. Podle velikosti střídy se pak řídí rychlost otáčení motoru. Když je střída malá, průměrné napětí je nízké a motor se otáčí pomalu, když je naopak střída vetší, zvětší se i průměrné napětí a motor se otáčí rychleji (viz obrázek 2.4).



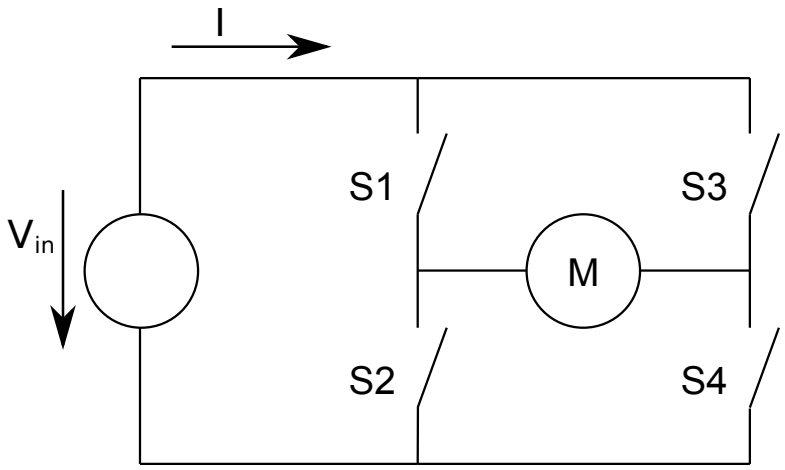
Obrázek 2.4: Pulzně šířková modulace (PWM)

2.4 H-můstek

V předešlé kapitole jsme se dozvěděli, co potřebuje k řízení stejnosměrných motorů. V této kapitole se dozvíme, co využijeme k tomu, aby se motory mohli točit v obou směrech. H-můstek [8] je elektronický obvod, který umožňuje využít napětí v obou směrech. V mém případě je tento obvod určen pro otáčení stejnosměrných motorů oběma směry. Termín H-můstek se používá díky typickému grafickému znázornění těchto obvodů. Toto zapojení se skládá ze čtyř spínačů (většinou tranzistorů). Pokud jsou spínače S1 a S4 (viz obrázek 2.5) sepnuty a spínače S2 a S3 rozepnuty, bude se motor otáčet jedním směrem. Když tuto situaci otočíme, tok proudu se obrátí a motor se bude otáčet opačným směrem.

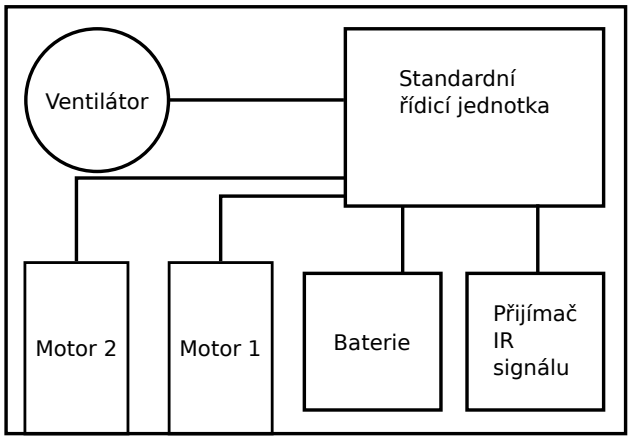
2.5 Požadavky na řídicí jednotku

Řídicí jednotka modelu musela splňovat následující požadavky. Vnitřní uspořádání modelu by se podle funkčnosti mělo lišit co možná nejméně od původního (viz obrázek 2.6). Nejdůležitější je její velikost, hmotnost, napájení pomocí baterie s napětím 3,7 V a schopnost řídit několik stejnosměrných motorů pomocí PWM výstupů. Jednotka by měla také obsahovat i další digitální a analogové vstupní/výstupní (I/O) piny pro další rozšiřování modelu. Velikost a hmotnost jsou velmi důležité, jednotka



Obrázek 2.5: Zapojení s využitím H-můstku

se musí vejít do podvozku vozu, jehož rozměry jsou značně omezeny, a nesmí mít moc velkou celkovou hmotnost (viz kapitola 2.1.1), aby byl model schopný udržet se na vertikální ploše a také se po ní pohybovat. Napájení z baterie poslouží k tomu, aby model nebyl omezený ve svém pohybu. Napětí baterie bude stejné, jako bylo ve standardním modelu (3,7 V), aby se dala převzít pohonná soustava.



Obrázek 2.6: Blokové schéma původního modelu

3 Použitý hardware

3.1 Řídicí jednotka

Pro účely práce se jako nejlepší jevila řídicí jednotka Arduino Fio (viz obrázek 3.1), jejím základem je 8bitový mikrokontrolér založený na architektuře RISC ATmega328P od firmy Atmel [5]. Společně s mikrokontrolérem je na této jednotce také obvod MAX1555 a obvod XBEE-1B3, o kterých se ještě zmíníme níže. Tato řídicí jednotka, s rozměry 27,9×66 mm a hmotností 9 g, má parametry:

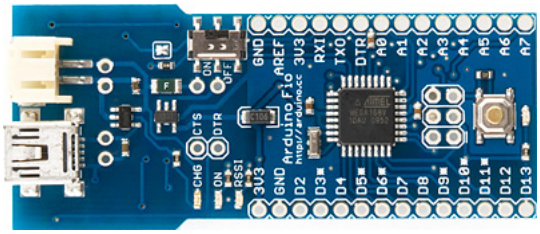
- 14 digitálních výstupů (z toho 6 umožňuje PWM),
- 8 analagových pinů,
- frekvence 8 MHz,
- paměť typu flash o velikosti 32 KB,
- operační napětí je 3,3 V,
- vstupní napětí 3,35–12 V,
- vstupní napětí pro nabíjení 3,7–7 V.

Arduino Fio může být napájeno dvěma způsoby, první z nich je pomocí rozhraní USB, druhá varianta je napájení z baterie o napětí 3,7 V. Je-li k Arduinu připojena baterie, tak zmíněné USB funguje jako port pro její nabíjení. Vybraná baterie váží 21 g. K nahrání programu do Arduina slouží rozšiřující plošný spoj, FTDI Basic Breakout adaptor [6], obsahující port USB (viz obrázek 3.2). Vždy, když se nahraje nový algoritmus, tento plošný spoj vyvolá přerušení a automaticky resetuje řídicí jednotku, nový program pak běží okamžitě bez zásahu uživatele.

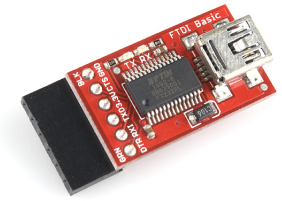
MAX1555
: Toto je malý integrovaný obvod, který na desce Arduina slouží k bezečnému a správnému nabíjení Li-polymerové baterie přes USB port. Maximální proud pro toto zařízení je 300 mA a nabíjecí napětí je mezi 3,7–7 V. Nevyžaduje téměř žádné externí komponenty a automaticky vybere mezi napájením pomocí USB nebo stejnosměrným zdrojem (baterie). Po dokončení nabíjení se sníží přívod proudu.

XBEE-1B3

Tento modul na desce Arduina se dá rozšířit o Wi-Fi modul se standardem IEEE 802.15.4. Pro řízení nebo připojení do malých bezdrátových sítí nebo pro využití bezdrátového ovládní.



Obrázek 3.1: Arduino Fio [4]



Obrázek 3.2: FTDI Basic Breakout adaptor [6]

3.2 Dvojitý H-můstek DRV8835

Pro řízení pojezdových kol, které jsou na stranách podvozku, potřebujeme H-můstek. Naším účelům nejlépe vyhovoval integrovaný duální H-můstek s označením DRV-8853 [11]. Tato součástka od výrobce Texas Instruments je určena pro řízení dvou stejnosměrných motorů v obou směrech při použitém napájení 0–11 V a proudu maximálně do 1,2 A (chvilkově dokáže snést i 1,5 A). Tento integrovaný obvod má 14 pinů, z nichž jsou dva určeny pro připojení na zem (GND), další dva pro napájení motorů a samotného obvodu, dva pro ovládání otáček motorů pomocí PWM, dva pro určování směru otáčení (logická 0 nebo 1), jeden pro mód obvodu (standardně log. 1) a čtyři pro samotné připojení dvou stejnosměrných motorů nebo jednoho krokového motoru. Tento obvod se prodává na rozšiřující desce [12], která s ním usnadňuje práci a je vybavena ochrannými prvky proti přepětí, nedostatečnému napětí a velkému proudu.

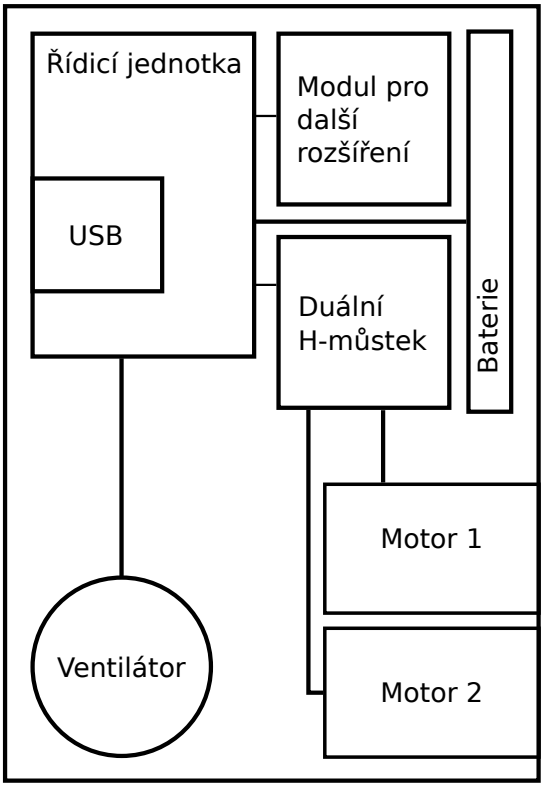
4 Sestavení a zapojení modelu

Původní myšlenka na přepracování modelu počítala s tím, že celý model bude řízen přes řídicí jednotku a napájen jednou baterií, jenž napájí i jednotku. Znázorněno na blokovém schéma (viz obrázek 4.1). Tato baterie s větší kapacitou než ta původní (3,7 V, 220 mAh) by měla zásobovat napětím jak řídicí jednotu, tak všechny 3 stejnosměrné motory. Toto řešení mělo mít za následek úsporu hmotnosti. Následoval návrh zapojení celého systému a jeho realizace. Prvotní schméma zapojení (v příloze C.1) obsahuje tři hlavní části, digitální výstupy řídicí jednotky Arduino Fio, zapojení spínacího tranzistoru pro řízení stejnosměrného motoru s ventilátorem a zapojení dvojitého H-můstku pro řízení pojezdových motorů.

Toto zapojení, ať vcelku jednoduché, se ukázalo dosti problematické. Baterie, která by měla napájet všechny části modelu najednou, je moc měkký zdroj. Měkké zdroje jsou zdroje s vnitřním odporem větším než 1 Ω. V praxi to znamená, že při zásobování řídicí jednotky a dalších indukčností (motorů) dojde k prudkému kolísání napětí a řídicí jednotka se resetuje. Bylo vyzkoušeno několik možných postupů, jak tento problém vyřešit např.: před stejnosměrné motory byly dány dva paralelně zapojené rezistory (toto řešení mělo za následek velký úbytek napětí a neefektivní řízení), předřadná indukce, která neměla žádný zlepšující výsledek atd. Nakonec se přistoupilo k nejschůdnějšímu a bezproblémovému řešení. Řídicí jednotka a všechny motory mají oddělené zdroje napájení. Zde našla využití baterie (3,7 V, 220 mAh), jež se nacházela v zakoupeném modelu, a využila se pro napájení motorů. Velká baterie (3,7 V, 1000 mAh), která měla napájet všechna zařízení modelu, byla nahrazena menší baterií o kapacitě 350 mAh. Schéma zapojení k tomuto řešení není o tolik rozdílné od původního (viz příloha C.2).

4.1 Ultrazvukový senzor

Nad takto sestaveným modelem vozidla se dále uvažovalo, jak ho rozšířit tak, aby bylo zajištěno jeho autonomní řízení. Jako elegantní řešení se nabídlo připojení ultrazvukového senzoru HC-SR04 [13]. Tento senzor vysílá ultrazvukový signál o frekvenci 40 kHz, který je vybuzen vstupním signálem z řídicí jednotky o minimální délce 10 mikrosekund. Pokud se tento signál setká s překážkou v rozmezí od 2–40 cm, tak se vrátí zpět k senzoru, a ten dá hodnotu o vzdálenosti v mikrosekundách. Takto vybavený model by byl schopný se vyhýbat překážkám do určené vzdálenosti. Popis algoritmu je v kapitole 5.2.1. Problémem se stalo napájení celého senzoru. Senzor HC-SR04 by měl být napájen napětím 5 V, některé zdroje ovšem udávaly, že funguje

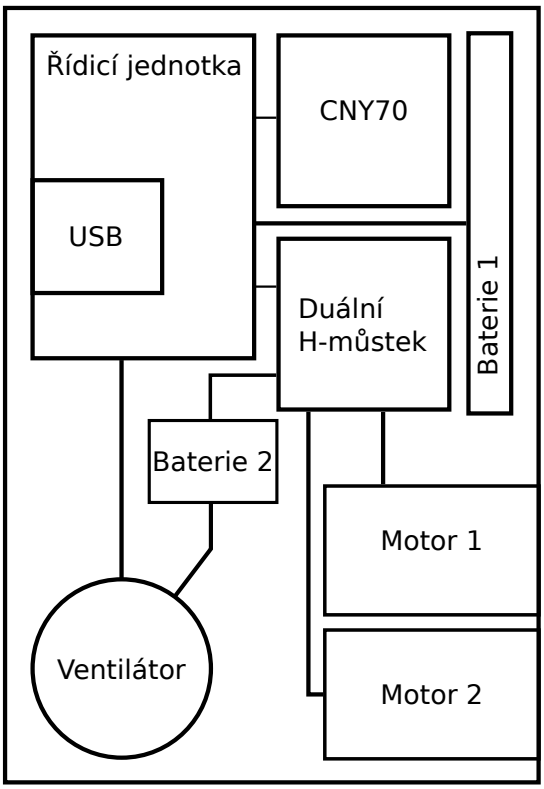


Obrázek 4.1: Blokové schéma přepracovaného modelu

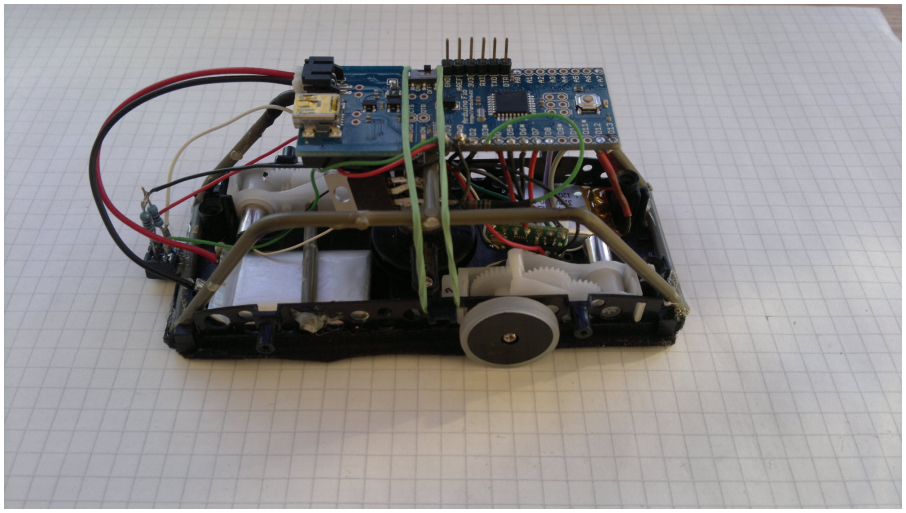
i na napětí 3,3 V, na kterém pracuje celé naše zařízení. Tato informace se ukázala jako chybná a senzor napájen napětím 3,3 V nevykazuje nic jiného než velmi zanedbatelný šum.

4.2 Optický senzor

K dispozici jsme měli také již vyzkoušené optické senzory CNY70 [14]. Tento reflektivní senzor obsahuje IR emitor a fototranzistor v pouzdře z neprůsvitného materiálu. Senzor CNY70 je schopný rozeznat bílou nebo černou barvu podkladu. Model vozidla s tímto senzorem by byl schopný sledovat černou (nebo bílou, záleží na barvě povrchu, na kterém se pohybuje) čáru nebo se volně pohybovat v ohraničeném poli. Aby se daly dobře číst informace z tohoto senzoru, byl jeho výstup připojen na analogový pin. U analogových pinů máme rozsah hodnot od 0 do 1023 a dá se s nimi lépe a přesněji pracovat. Senzor byl zapojen tak, jak je vidět na schématu zapojení C.2. Takto vypadá finální řešení našeho modelu, pro který bylo následně napsanáno několik řídicích algoritmů pro jeho autonomní pohyb. Tyto algoritmy jsou popsány v následující kapitole. Vzhled a blokové schéma finálně vylepšeného modelu s přísavným systémem vozu najdede na obrázku 4.2 a 4.3 .



Obrázek 4.2: Finální blokové schéma modelu vozidla s přítlačným systémem



Obrázek 4.3: Finální vzhled vytvořeného modelu vozidla s přítlačným systémem

5 Software

Mikrokontrolér ATmega328P na desce Arduino Fio má v sobě před-nahraný bootloader, který umožňuje nahrát nový kód bez externího zařízení komunikující mezi mikrokontrolérem a PC. Tento bootloader komunikuje pomocí originálního protokolu STK500 [15]. Existují dva postupy, jak nahrát do Arduina nové řídicí algoritmy. První je použitím FTDI USB se seriovým kabelem nebo, jako v našem případě, využitím seriového adaptoru s rozhraním USB. Druhá možnost je využitím XBee rozšíření pro přenos programu bezdrátově. Samotné programování pak probíhá v open-source Arduino Software (IDE) [16]. Pro programování řídicích algoritmů se využívá jazyk podobný C/C++. Programy v Arduinu mohou být rozděleny do tří hlavních částí struktury, hodnoty (proměnné a konstanty) a funkce.

5.1 Základní funkce

Program pro řídicí jednotku Arduino Fio je sestaven ze dvou hlavních částí. Jsou to dvě struktury, struktura **setup()** a **loop()**. První z uvedených, funkce **setup()**, je zavolána vždy na začátku programu. Využívá se pro inicializování proměnných, módů pro piny, začíná využívat knihovny apod. Funkce **setup()** je zavolána pouze jednou, pokaždé kdy je zapnuto napájení nebo po resetování obvodu. Poté, co proběhne funkce **setup()**, která vše inicializuje a nastaví hodnoty, funkce **loop()** dělá přesně to, co její název napovídá a je procházena ve smyčkách, tak nám umožňuje reagovat a měnit náš program. Uvedu jednoduchý příklad.

```
//Ovladani DC motoru s ventilatorem
int motorPin = 3;

void setup()
{
  pinMode(motorPin, OUTPUT);
}

void loop()
{
  analogWrite(motorPin, 255);
}
```

Jako první řádek programu máme stanovení globální proměnné, v tomto případě nastavuje pin který budeme využívat. Následuje již popisovaná funkce **setup()**, ve

které nastavuji pin 3 jako výstupní, a to funkcí **pinMode()**. První parametr v této funkci se zadává číslo pinu a jako druhý parametr se určuje, jestli bude vstupní INPUT nebo výstupní OUTPUT. Poté, co máme vše nastaveno, se spustí funkce **loop()**. Tady v této části určíme, co a jak chceme s daným pinem dělat. V tomto jednoduchém případě funkcí **analogWrite()** nastavujeme hodnotu PWM a číslem od 0 do 255 nastavujeme šířku pulzu. Funkce **analogWrite()** nám bude generovat stabilní obdélníkový výstup na určeném pinu a v určeném pracovním cyklu až do dalšího zavolání **analogWrite()**. Frekvence signálu PWM je pro časovač 0 (**timer0**) 488,28 Hz, tento časovač ovládá piny 5 a 6, u časovačů 1 a 2 (**timer1** a **timer2**) ovládajících piny 9, 10, 11 a 3 je frekvence 245,1 Hz. Každý z časovačů má přednastavený dělič systémových hodin pro generování PWM, zde máme přednastavené faktory 1, 8, 64, 256 a 1024. Standardně je nastaven dělicí faktor 64. Toto se dá softwarově změnit pomocí ovládání registrů čítačů. Zde je ukázka kódu funkce, jenž mění nastavení frekvence PWM.

```
void setPwmFrequency(int pin, int div) {
  byte mode;
  //vyber faktoru deleni pro casovace0 a 1
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(div) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      //nastaveni casovace 0
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      //nastaveni casovace 1
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  //vyber faktoru deleni pro casovace2
  } else if(pin == 3 || pin == 11) {
    switch(div) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
      case 1024: mode = 0x7; break;
      default: return;
    }
```

```
        //nastaveni casovace 2
        TCCR2B = TCCR2B & 0b11111000 | mode;
    }
}
```

Změnou frekvence PWM signálu se u některých motorů dá dosáhnout většího výkonu. V mém případě jsem tímto chtěl dosáhnout většího výkonu na ventilátoru, a tím i větší stability na vertikální ploše. Ukázalo se však, že u takto malých stejnosměrných motorů je změna minimální a nejdůležitější je šířka pulzu PWM.

5.2 Popis testovacích algoritmů pro senzory HC-SR04 a CNY70

5.2.1 HC-SR04

V kapitole 4.1 jsme se zmínili o ultrazvukovém senzoru HC-SR04. Senzor obsahuje 4 piny, a to napájecí, uzemňovací a pro nás nejdůležitější pin **trig** a **echo**. Ve funkci **setup()** si nastavíme oba tyto piny tak, že pin **trig** bude OUTPUT a pin echo bude INPUT. Přes pin **echo** ze senzoru přichází informace, jestli vyslaný signál přes pin trig, který je realizován 10 mikrosekundovým pulzem, narazil na nějakou překážku ve svém rozsahu, pokud ano, vrátí nám číslo. Na toto dobře využijeme funkci **pulseIn(pin, hodnota)**, jenž přečte příchozí pulz na pin. Např. pokud je nastavena hodnota HIGH (log. 1), funkce **pulseIn()** čeká, kdy se na pinu objeví log. 1 a začne počítat uplynulý čas do té doby než se na pinu objeví log. 0 a vrátí čas v mikrosekundách. S tímto časem je dál potřeba pracovat, abychom dostali výslednou vzdálenost překážky, nacházející se před senzorem. Rychlost zvuku v pokojových podmínkách je zhruba 340 m/s nebo 29,1 cm/µs. Vezeme tedy čas, jenž nám vrátil senzor HC-SR04 v mikrosekundách a vydělíme ho rychlostí šíření zvuku. Výsledné číslo opět vydělíme, a to číslem 2. Důvod je ten, že čas, který nám vrátí senzor, je počítaný za cestu tam a zpět. Nám tedy bude stačit poloviční údaj. Testovací algoritmus byl také doplněn o výpisy na seriový monitor, aby se dalo otestovat, zda měří vzdálenost v pořádku. Celý testovací kód je v příloze D.2.

5.2.2 CNY70

Testovací algoritmus pro optický senzor CNY70 je celkově jednoduší než v předešlém případě. Tento optický senzor, který má v sobě IR emitor a fototranzistor, snímá povrch, který je před ním a dokáže dobře rozeznat, jestli je povrch světlý (bílý), nebo černý. Pro čtení těchto hodnot využijeme funkci **analogRead(pin)**, před tím ovšem musíme nastavit tento pin na INPUT. Funkce **analogRead()** čte hodnoty z daného analogového pinu. Arduino obsahuje 8 kanálů s 10-bitovým A/D převodníkem. To znamená, že vstupní napětí mezi 0 a 3,3 V rozloží na hodnotu v integeru od 0 až do 1023. Z toho vychází, že máme rozlišení vstupu 3,2 mV na jednotku. Přečtení této jednotky trvá 100 µs. Takto nastavený testovací algoritmus byl doplněn o výpisy na seriový monitor pro možnost testování.

5.3 Popis řídicích algoritmů s využitím senzoru CNY70

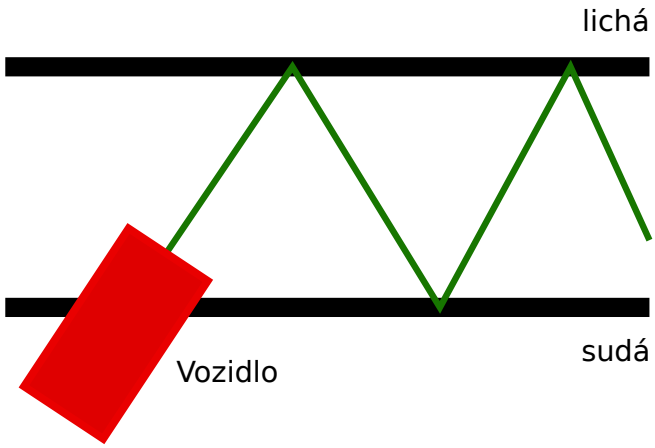
Pro finální verzi modelu s přísavným systémem bylo napsáno několik řídicích algoritmů, jenž pro autonomní řízení využívají optický senzor CNY70. Oba z těchto algoritmů mají stejné základní nastavení výstupů pro řízení všech motorů a ventilátoru v těle funkce **setup()** a jednu základní podmínku pro spuštění algoritmu až po 5 sekundách od spuštění napájení.

5.3.1 Algoritmus pro udržení vozidla v ohraničeném poli

První z algoritmů je napsán tak, aby se vozidlo na vertikální ploše pohybovalo v poli ohraničeném černým okrajem (realizováno černou páskou). Jak jsme zjistili pomocí testování (viz následující kapitola 6.3), senzor při detekci černé barvy vrací na analogový pin A0, pomocí funkce **analogRead(pin)**, hodnoty do čísla 200. Tohoto velmi jednoduše využijeme, a to tak, že do těla funkce **loop()** dáme podmínku. Pokud hodnota na pinu A0 klesne pod 200, znamená to, že předek vozu, kde je připevněn senzor, narazil na hraniční čáru pole, ve kterém se pohybuje, spustí se kód podmínky a vozidlo o kousek zacouvá a otočí se. Pokud tato podmínka není splněna, vozidlo se stále pomalu pohybuje směrem vpřed. Program obsahuje několik vlastních funkcí, které realizují kód pro jízdu vpřed, vzad, otáčení a zastavení motorů. Tyto funkce mají vstupní hodnotu integer, kde udáváme velikost PWM. Program se dá tak jednoduše měnit dle potřeby. Kód je také doplněn o výpisy na seriový monitor pro jeho odladění.

5.3.2 Algoritmus pro opsání a udržení se ve vyznačené dráze

Druhý algoritmus pro realizaci řízení využívá dráhy, která je ohraničena dvěma černými páskami, a vozidlo je schopné opsat tvar dráhy tím, že bude jezdit od jednoho kraje ke druhému, jak je znázorněno na obrázku 5.1. Počítá se s tím, že startovní pozice vozu je nakloněná k pravému okraji (čáře) označující trať. Opět využíváme hodnot na pinu A0, čtené funkcí **analogRead()**. V těle funkce **loop()** máme stejnou podmínku jako v předešlém případě, pokud hodnota na pinu A0, ukládaná do proměnné integer s názvem value, klesne pod hodnotu 200 vyvolá se další podmínka. V této podmínce využíváme dvě pomocné proměnné typu integer a to **temp** a **temp2**. Proměnná **temp2** se zvětší o hodnotu 1 pokaždé, kdy vozidlo narazí na hraniční čáru. Abychom zjistili u jaké čáry se vozidlo nachází, využijeme proměnnou **temp**, do které ukládáme výsledek výpočtu, kde proměnnou **temp2** dělíme modulem dvěma. Pravá čára z pohledu vozidla a také startovní strana má sudé hodnoty a levá strana má hodnoty liché. Z toho nám plyne, že pokud je hodnota **temp** nulová, vozidlo je u pravé čáry a otočí se směrem k levé čáře, a pokud je hodnota **temp** větší než nula (výsledek modula je zbytek po dělení dvou čísel), nachází se u levé čáry a zatočí vpravo. Pokud hlavní podmínka nen9 splněna, vozidlo se nachází mezi čárami a pohybuje se pomalu směrem kupředu. Tímto stylem jízdy je vozidlo schopno opsat libovolnou dráhu. Kód je opět doplněn o výpisy na sériový monitor pro lepší odladění a testování. Celý algoritmus najdete v příloze D.3.



Obrázek 5.1: Styl pohybu vozidla mezi ohraničením dráhy

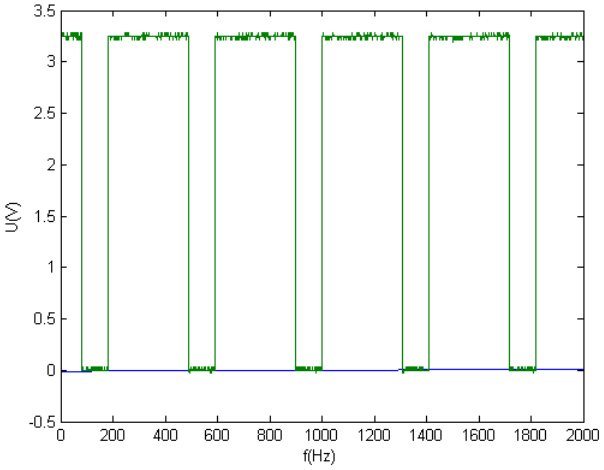
6 Testování

6.1 Testování zapojení ventilátoru a frekvence PWM

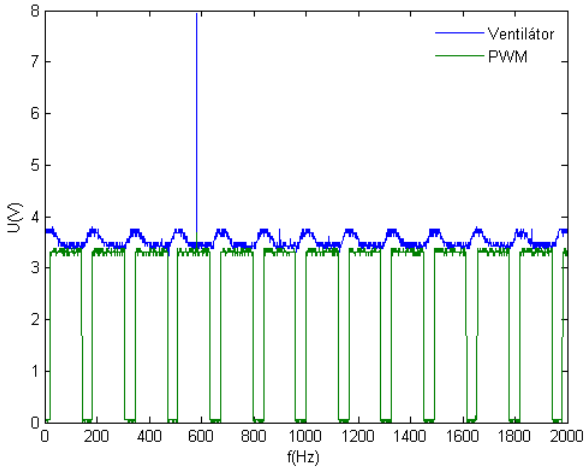
Původní zapojení celého systému vozidla počítalo s tím, že ho napájením bude zásobit pouze jedna baterie. Při řešení zrovoznění pojezdových stejnosměrných motorů jsme museli od této ideje odstoupit a samotné motory napájet zvlášť jinou baterií, než kterou je napájena řídicí jednotka. Pojezdové motory pak fungovaly bez problémů. Stejný problém se nakonec ukázal v zapojení a řízení ventilátoru. Napájení samotného motoru z řídicí jednotky způsobovalo její restartování kvůli kolísání napětí. Takto zapojený ventilátor se točil, ale než se roztočil na plný výkon, který je u něj vyžadován pro větší přítlak, řídicí jednotka se několikrát restartovala. Testovali jsme, jestli není problém v PWM, jenž přichází na spínací tranzistor. Připojili jsme tedy na výstupní pin D3* osciloskop, abychom zjistili, jak vypadá nastavené PWM (softwarově bylo nastavené PWM na 75 % výkonu). Výstup na osciloskopu (viz obrázek 6.1) nám ukázal, že v řídicí jednotce ani ve výstupním pinu problém není. Na těchto výsledcích jsme si také ověřili, že frekvence PWM je skutečně 245,1 Hz. Proto jsme přistoupili ke stejnému řešení jako u pojezdových motorů. Místo napájení ventilátoru přes řídicí jednotku jsme zvolili napájení další baterií (tou samou jako u pojezdových motorů). Toto řešení vyřešilo veškeré problémy s restartováním jednoty a ventilátor se točí na plný výkon bez větších ztrát napětí. Abychom si ověřili, jak motor přijímá řídicí signál PWM, opět jsme využili osciloskop. Výsledný graf vypadá následovně, viz obrázek 6.2.

6.2 Testování senzoru HC-SR04

Zde se jen okrajově vrátíme k tomu, co bylo zmíněno v minulé kapitole 5.2.1. Pomocí testovacího algoritmu napsaného pro tento senzor byla testována jeho funkčnost. Toto testování ukázalo, že senzor není schopen fungovat správně s napájecím napětím, jenž máme k dispozici (3,3 V). Pomocí výstupu na sériový monitor si můžeme ukázat zpracovávané hodnoty ze senzoru (viz tabulka 6.1). Jak je na této tabulce vidět, tak hodnoty **duration**, které by měly obsahovat vzdálenost překážky, jsou spíše jen šum zařízení, než reálné hodnoty. Po tomto zjištění bylo od použití tohoto senzoru ustoupeno.



Obrázek 6.1: PWM zobrazené na osciloskopu



Obrázek 6.2: PWM a signál na ventilátoru

6.3 Testování senzoru CNY70

V předešlých kapitolách bylo popsáno zapojení tohoto senzoru a o jeho testovacím algoritmus. V první verzi zapojení byl využit rozdílný odpor R3 (ze schématu C.2) s hodnotou 1,1 kΩ. Pomocí testovacího algoritmu a modelu propojeného s PC jsme dostali na seriový monitor tyto hodnoty (tabulka 6.2). Hodnoty pohybující se od 0 do 20 reprezentují, že se před senzorem nachází objekt z černé barvy. Hodnoty které se naopak pohybují kolem hodnoty 100 reprezentují barvu bílou. Abychom zvětšili rozsah hodnot pro přesnější využití senzoru, byl odpor R3 C.2 zvětšen na 47 kΩ. S tímto odporem se rozsah hodnot pohybuje od 0 do 1023, jak je vidět v tabulce 6.2. Na těchto hodnotách je vidět, že hodnoty pro černou barvu se pohybují od 0 do

Tabulka 6.1: Data z testovacího algoritmu pro senzor HC-SR04

distance [cm]	**duration [µs]**
0	0
0	2
0	2
0	5
0	2
0	0
0	2

200 (až 250) a pro bílou do maximální hodnoty 1023. Těchto poznatků jsme využili při programování autonomního řízení založeného na tomto optickém senzoru.

Tabulka 6.2: Ukázky dat ze zapojení senzoru CNY70

	hodnota A/D
Hodnota	109
Hodnota	102
Hodnota	103
Hodnota	102
Hodnota	7
Hodnota	12
Hodnota	0
Hodnota	3
Hodnota	4
Hodnota	4
Hodnota	4

	hodnota A/D
Hodnota	989
Hodnota	988
Hodnota	293
Hodnota	294
Hodnota	281
Hodnota	96
Hodnota	43
Hodnota	58
Hodnota	73
Hodnota	78
Hodnota	87

7 Možnosti dalšího rozšíření

Stávající prototyp modelu vozidla s přísavným systémem by se dal rozšířit o další prvky, jenž by zlepšily schopnosti jeho autonomního řízení. V této kapitole uvedu několik řešení, které by mohly projekt dále rozšířit. U všech těchto rozšíření se musí dát velký pozor na konečnou hmotnost modelu. Řídicí jednota ve stávajícím zapojení má volných 7 analogových pinů a 6 digitálních pinů, z toho 3 s možností PWM.

7.1 Senzorické

7.1.1 Akcelerometr

Akcelerometr je elektromechanické zařízení, které měří zrychlení. Pomocí tohoto zařízení můžeme určit v jakém úhlu je vozidlo k zemskému povrchu, a tím např. zapínat, vypínat ventilátor nebo korigovat výkon pro motory atd. Nabízí se nám mnoho různých řešení a také obvodů, které se dají využít. Uvedu zde například akcelerometr MMA7361 [18] nebo obvod IMU 5DOF IDG500/ADXL335 [19], jenž má v sobě zabudovaný akcelerometr i gyroskop (gyroskop měří úhlové zrychlení). Oba tyto uvedené obvody pracují s napětím 3,3 V a mají v našem projektu všestranné využití. Pro oba obvody se dají vyhledat knihovny pro řídicí jednotku Arduino Fio.

7.2 Komunikační

7.2.1 Rozhraní XBee

Řídicí jednotka Arduino Fio obsahuje připravené rozhraní pro modul XBee, jenž slouží pro příjem informací přes rozhraní Wi-Fi. Těchto modulů je několik typů a verzí. Kvalitní přehled nejpoužívanějších je na stránkách od firmy Sparkfun [17]. Rozšíření systému o tento modul by mohlo zajistit bezdrátové nahrávání programu nebo bezdrátový příjem dat za provozu modelu a jeho ladění. V neposlední řadě by se tento modul dal využít i pro dálkové řízení modelu vozidla.

7.2.2 Rozhraní Bluetooth

Bluetooth je další využívaný standard pro bezdrátovou komunikaci propojující dvě a více zařízení. Model by se dal obohatit i o toto rozhraní, např. o modul HC-05

[20], který je celkově dostupný. Model vozidla vybavený tímto rozhraním by se mohl bezdrátově ovládat přes PC nebo mobilní zařízení.

Závěr

V rámci této bakalářské práce byl vytvořen automaticky se řídicí model vozu s přísavným systémem, který je schopný jezdit po horizontálních a vertikálních plochách. Postup práce spočíval v teoretickém rozboru koupeného modelu vozu, jeho nároků na napájení a vlastností přísavného systému. Po tomto rozboru byl navrhnut nový systém řízení s využitím nového řídicího hardwaru realizovaného řídicí jednotkou Arduino Fio, která nejlépe vyhovovala všem potřebným parametrům.

Po několika verzích zapojení a testování všech částí vozu bylo vozidlo uvedeno do provozu. Vozidlo se bez sebemenších problémů pohybuje po hladkých vertikálních plochách (testováno převážně na skle).

Na funkčním modelu bylo testováno několik druhů senzorů pro zajištění autonomního řízení. Byly testovány ultrazvukové senzory pro možnost vyhýbání se překážkám a optické senzory pro umožnění pohybu ve vyznačeném poli. Výsledkem tohoto testování byl výběr optického senzoru, který splňoval omezení napájením, spotřebou a velikostí.

Pro autonomní model pak bylo vytvořeno několik testovacích a řídicích algoritmů. Testovacími algoritmy byla ověřena funkčnost všech hardwarových součástí vozidla. Tyto algoritmy pomohly vyřešit mnoho problémů při vývoji celého vozidla. Dále byl vytvořen řídicí algoritmus pro pohyb v ohraničeném poli a algoritmus pro udržení vozidla v ohraničené dráze a opsání jejího tvaru. Řídicí algoritmy zajišťují autonomní pohyb na horizontálních i vertikálních plochách.

V práci se podařilo splnit všechny body zadání a vytvořit funkční automaticky se řídící model s přísavným systémem.

Literatura

[1] RC AUTO WALL CLIMBER 866 © 2014 RC Profi [online]. [cit. 2015-04-02]. Dostupné z: http://www.rcprofi.cz/rcprofi/eshop/4-1-AUTA/8-2-MINI-auticka/5/58-RC-AUTO-WALL-CLIMBER-866-JEZDI-PO-STENACH

[2] Mini DC Motor © 2010 Gizmoszone [online]. [cit. 2015-17-01]. Dostupné z: http://www.gizmoszone.com/shopping/html/pages/gdc48.pdf

[3] Těleso na nakloněné rovině Encyklopedie fyziky © 2006–2015 [online]. [cit. 2015-15-05]. Dostupné z: http://fyzika.jreichl.com/main.article/view/1540-teleso-na-naklonene-rovine

[4] Arduino Fio © 2015 Arduino [online]. [cit. 2015-03-10]. Dostupné z: http://arduino.cc/en/Main/ArduinoBoardFio

[5] ATmega328P © 2015 Atmel Corporation [online]. [cit. 2015-15-03]. Dostupné z: http://www.atmel.com/devices/atmega328p.aspx

[6] FTDI Basic Breakout © 2009 SparkFun Electronics [online]. [cit. 2015-03-01]. Dostupné z: https://www.sparkfun.com/products/retired/8772

[7] Arduino FIO Tutorial © 2013 DXARTS [online]. [cit. 2014-06-12]. Dostupné z: http://wiki.dxarts.washington.edu/groups/general/wiki/ec15a/Arduino_FIO_Tutorial.html

[8] H-můstky – řízení stejnosměrných motorů © 2011 Gymšpit [online]. [cit. 2014-02-12]. Dostupné z: http://robot.gymspit.cz/new/cz/eurobot-2010/motor-controll-2010/hbridge/

[9] H-Bridges: Theory and Practice © 2006 Chuck McManis [online]. [cit. 2014-02-12]. Dostupné z: http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/

[10] PWM © 2015 Arduino [online]. [cit. 2014-05-12]. Dostupné z: http://arduino.cc/en/Tutorial/PWM

[11] DRV8835 © 1995–2015 Texas Instruments [online]. [cit. 2015-15-03]. Dostupné z: http://www.ti.com/product/drv8835

[12] DRV8835 Dual Motor Driver Carrier © 2001–2015 Pololu Corporation [online]. [cit. 2015-15-03]. Dostupné z: https://www.pololu.com/product/2135

[13] HC-SR04 © ELECFreaks [online]. [cit. 2015-29-04]. Dostupné z: http://www.micropik.com/PDF/HCSR04.pdf

[14] CNY70 © 2009 VISHAY [online]. [cit. 2015-27-04]. Dostupné z: http://www.vishay.com/docs/83751/cny70.pdf

[15] Protokol STK500 © Atmel Corporation 2003 [online]. [cit. 2015-14-04]. Dostupné z: http://www.atmel.com/Images/doc2525.pdf

[16] Arduino Software © 2015 Arduino [online]. [cit. 2015-14-04]. Dostupné z: http://arduino.cc/en/Main/Software

[17] XBee Buying Guide © 2015 SparkFun Electronics [online]. [cit. 2015-02-05]. Dostupné z: https://www.sparkfun.com/pages/xbee_guide

[18] MMA7361L © Freescale Semiconductor 2008 [online]. [cit. 2015-05-11]. Dostupné z: http://www.freescale.com/files/sensors/doc/data_sheet/MMA7361L.pdf

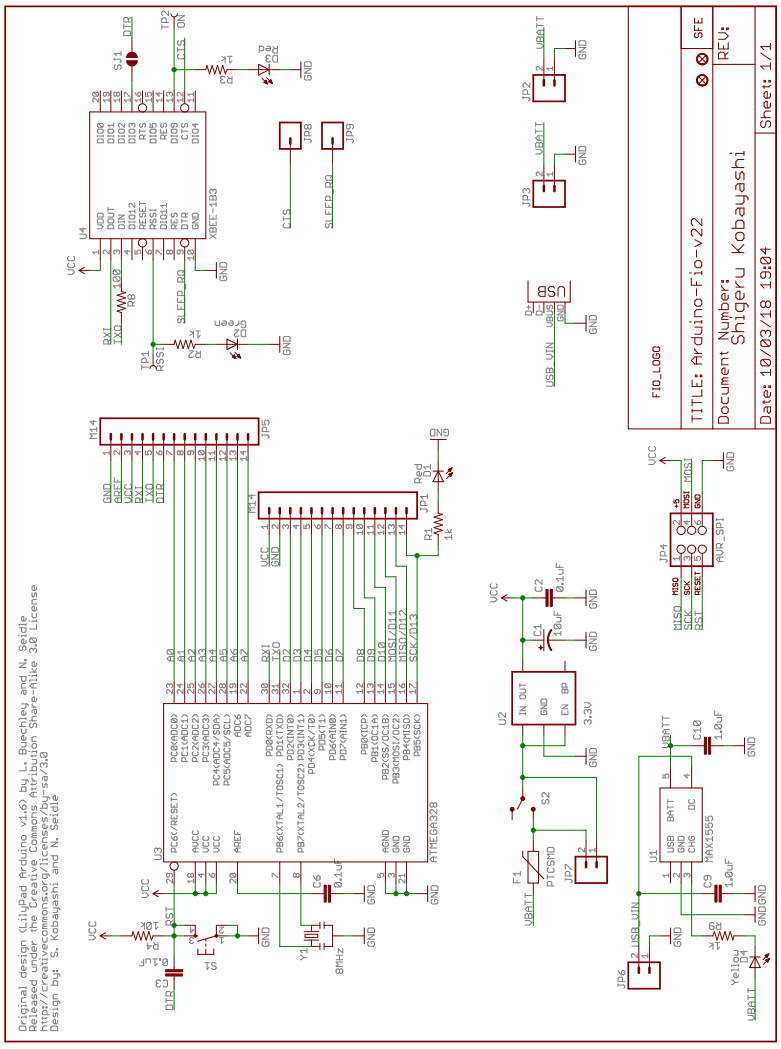
[19] IMU 5DOF © SparkFun Electronics [online]. [cit. 2015-05-11]. Dostupné z: https://www.sparkfun.com/products/retired/9268

[20] HC-05 – Bluetooth to Serial Port Module © ITead Studio 2010 [online]. [cit. 2015-05-11]. Dostupné z: ftp://imall.iteadstudio.com/Modules/IM120723009/DS_IM120723009.pdf

A Obsah přiloženého CD

- text bakalářské práce
  - bakalarska_prace_2015_Petr_Najman.pdf
  - kopie_zadani_bakalarska_prace_2015_Petr_Najman.pdf
- zdrojový kód programu
  - Testovací algoritmy
  - Řídicí algoritmy
- výkresová dokumentace
  - Arduino-Fio-schematic.pdf
  - prvotni_zapojeni.pdf
  - konecne_zapojeni.pdf
- katalogové listy použitých součástek
  - drv8835.pdf
  - cny70.pdf

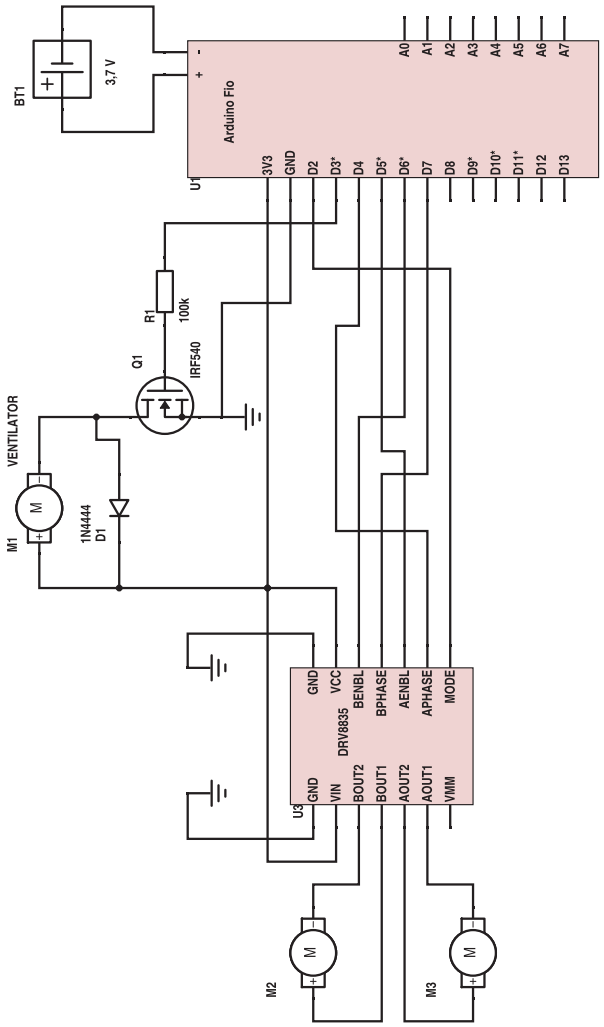
B Schéma zapojení desky Arduino Fio



Obrázek B.1: Schéma zapojení řidicí jednotky Arduino Fio [4]

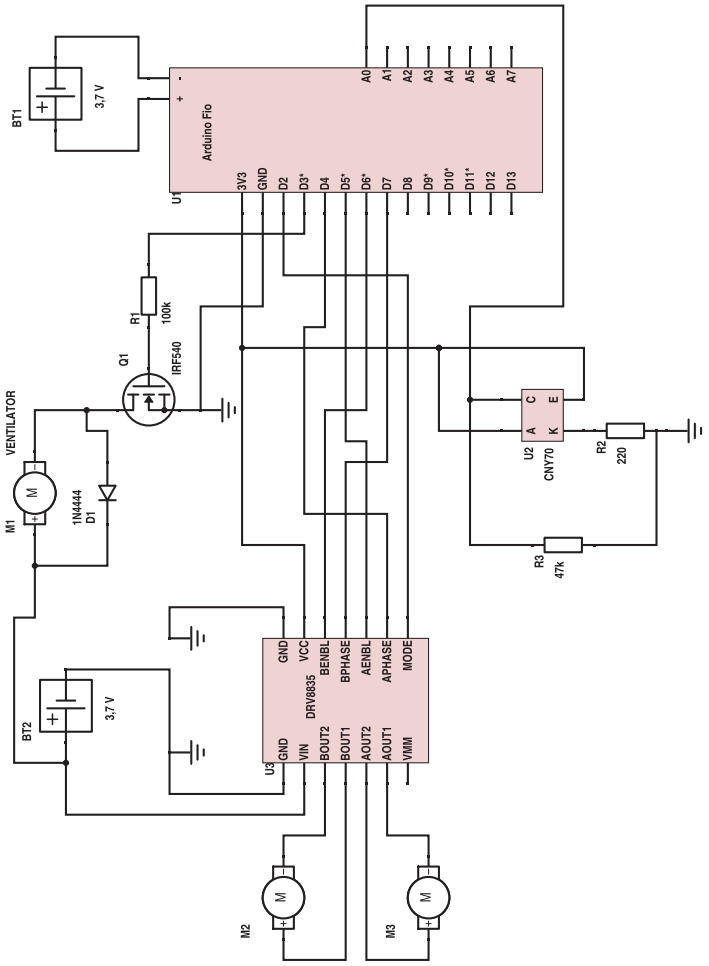
C Schéma zapojení upraveného modelu

C.1 Prvotní verze zapojení



Obrázek C.1: Prvotní návrh zapojení modelu

C.2 Konečná verze zapojení



Obrázek C.2: Finální návrh zapojení modelu s konečnými hodnotami

D Ukázky testovacích algoritmů

D.1 Rozsvěcení LED

```
int led = 12;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

D.2 Testovací algoritmus senzoru HC-SR04

```
//piny pro informace ze senzoru
const int trigPin = 8;
const int echoPin = 12;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  //kumunikace pres seriovy port
  Serial.begin(9600);
}

void loop() {

  long duration, distance;

  digitalWrite(trigPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  delay(5000);
  Serial.print(duration);
  Serial.print(" duration ");
  Serial.println();
  distance = microsecondsToCentimeters(duration);

  if(distance < 5) {
    Serial.println("Prekazka do 5 cm!");
  }

  Serial.print(distance);
  Serial.print(" cm ");
  Serial.println();

  delay(100);
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29,1 / 2;
}

}
```

D.3 Algoritmus pro opsání a udržení se ve vyznačené dráze

```
//log vstup pro pin MODE
\#define modePin 2
//pin pro rizeni ventilatoru
\#define ventPin 3
//log vstup pro motor 1
\#define motorPin1 4
//log vstup pro motor 2
\#define motorPin2 7
//piny pro rizeni motoru
\#define powerPin1 5
\#define powerPin2 6
```

```
//pin pro sledovani hodnot ze senzoru
\#define sensor A0
int value = 0;
int temp = 0;
int temp2 = 0;

void setup() {
  //nastaveni pinu na vstupni nebo vystupni
  pinMode(sensor, INPUT);

  pinMode(modePin, OUTPUT);

  pinMode(ventPin, OUTPUT);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);

  pinMode(powerPin1, OUTPUT);
  pinMode(powerPin2, OUTPUT);
  //nastaveni pinu mode na log. 1
  digitalWrite(modePin, HIGH);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:

  if (temp2 == 0) {
    delay(5000);
    temp2++;
  }
  analogWrite(ventPin, 255);
  value = analogRead(sensor);
  Serial.print(value);
  Serial.print("␣");
  //PWM 255-100\%, 191-75\%, 127-50\%, 64-25\%
  if(value < 400) {
    temp = temp2 \% 2;
    if(temp != 0) {
      motors_stop();
      delay(10);
      //////
      right(191);
      delay(160);
      //////
      motors_stop();
```

```
      delay(10);
      //////
      temp2++;
      Serial.print(temp);
      temp = 0;
      Serial.println("␣Otoceni␣na␣lichou␣vpravo.");
      //delay(1000);
    } else if(temp == 0) {
      motors_stop();
      delay(10);
      //////
      left(191);
      delay(180);
      //////
      motors_stop();
      delay(10);
      temp2++;
      Serial.print(temp);
      temp = 0;
      Serial.println("␣Otoceni␣na␣sudou␣vlevo.");
      //delay(1000);
    }
  }
  forward(60);
  Serial.println("Jizda␣rovne.");
  //delay(1000);
}

void motors_stop() {
  analogWrite(powerPin1, 0);
  analogWrite(powerPin2, 0);
}

void forward(int power) {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  analogWrite(powerPin1, power);
  analogWrite(powerPin2, power-5);
}

void right(int power) {
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  analogWrite(powerPin1, power);
  analogWrite(powerPin2, power-5);
```

```
}

void left(int power) {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, HIGH);
  analogWrite(powerPin1, power);
  analogWrite(powerPin2, power-5);
}
```