

**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA ELEKTROTECHNICKÁ
A INFORMAČNÍCH TECHNOLOGIÍ BRNO**



AUTONOMNÍ PÁSOVÉ VOZIDLO

JAN MACKŮ
L4CR

PROFILOVÁ ČÁST MATURITNÍ ZKOUŠKY
MATURITNÍ PRÁCE

BRNO 2015

Prohlášení

Prohlašuji, že jsem maturitní práci na téma Autonomní pásové vozidlo vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že:

- Beru na vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na internetu Střední průmyslové školy elektrotechnické a informačních technologií, Brno, Purkyňova 97.
- Beru na vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠEIT Brno, dostupná k prezenčnímu nahlédnutí. Škola zajistí, že nebude pro nikoho možné pořizovat kopie jakékoli části práce.
- Beru na vědomí, že SPŠEIT Brno má právo celou moji práci použít k výukovým účelům a po mé souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.
- Beru na vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

V Brně dne:

Podpis:

Poděkování

Chtěl bych poděkovat vedoucí práce Ing. Miroslavě Odstrčilíkové za vedení této maturitní práce, za pomoc při její tvorbě a realizaci.

Anotace

Tato práce se zabývá výrobou a vývojem Autonomního pásového vozidla. Vozidlo je vybaveno třemi ultrazvukovými snímači a dvěma stejnosměrnými motory, z nichž každý pohybuje jedním pásem. Řídící jednotkou je Arduino Leonardo doplněno o desku řídící motory a desku ovládající LCD display. Vytvořené Autonomní pásové vozidlo se dokáže samostatně pohybovat v prostoru.

Obsah

1	Úvod	1
1.1	Využití	1
1.2	Zadání	1
1.3	Teoretický rozbor	1
2	HW	3
2.1	Arduino	3
2.2	Ardumoto	5
2.3	LCD	5
2.4	Pojezd	5
2.5	Senzor	6
3	SW	8
3.1	Arduino IDE	8
3.2	Wiring	8
3.3	Multitasking	9
3.3.1	Kooperativní multitasking	9
3.3.2	Preemptivní multitasking	10
3.4	Řízení	10
4	Výroba	13
4.1	Konstrukce	13
4.2	Program	19
4.3	Ladění	24
4.3.1	Rovná jízda	24
4.3.2	Otáčení	25
4.3.3	Zastavování	25
5	Závěr	26
6	Zkratky a veličiny	27
	Literatura	29

KAPITOLA 1

Úvod

Využití

Využití autonomních vozidel je široké. Autonomní vozidla posíláme na místa, kam člověk nemůže, např.: do vesmíru, nebo na místa, kde je to pro člověka nebezpečné, např.: do rozpadených budov, kde hrozí zřícení. Také je využíváme na práce, které jsou pro člověka nudné, nebo jinak nepříjemné např.: robotický vysavač. Jestliže mluvíme o autonomním vozidle, myslí se tím vozidlo, které na základě hodnot ze senzorů se dokáže samostatně pohybovat v prostoru a případně řešit různé situace.

Zadání

Zadáním této maturitní práce je vytvořit autonomní pásové vozidlo dále jen APV, které se bude pohybovat samostatně v prostoru vymezeném svislými stěnami (bludiště).

Teoretický rozbor

Řízení pásových vozidel je velice složitá záležitost. Správné řízení závisí na mnoha aspektech. Pásová vozidla se řídí změnou rychlostí jednotlivých pásů. Pokud chceme správně řídit pásové vozidlo musíme brát v potaz nejen rychlosti pásů, ale i tření, terén a další aspekty, které by mohly ovlivnit směr a rychlosť pohybu vozidla. Z tohoto důvodu je řízení pásových vozidel velice náročné. Hlavně proto, že některé z aspektů nejsou ani pořádně prakticky popsané a pouze o nich uvažujeme jako například o tření. Řízením autonomních vozidel se zabývají např.: [\[Cre85\]](#), nebo [\[FR14\]](#).

Pro správné řízení vozidla je potřeba nejen znát jednotlivé vzdálenosti od stěn, ale i úhel natočení vůči nim. Díky tomuto úhlu je schopno vozidlo správně a včas zareagovat na přiblížování ke stěně (viz. obr. 1.1). Tento úhel je možno vypočítat pomocí následujících vzorců:

$$\alpha \approx (v_1 - v_2) t$$

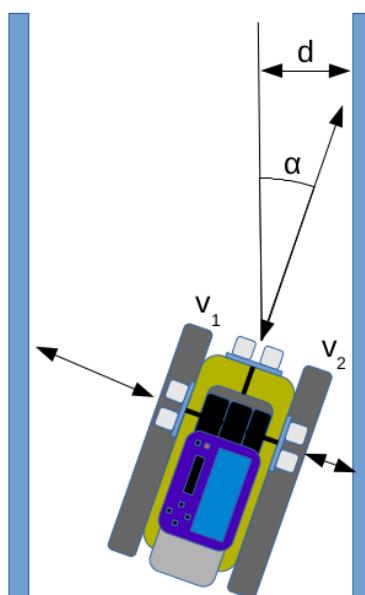
Úhel natočení vůči stěně (α) odpovídá rozdílu rychlostí jednotlivých pásů (v_1 ; v_2) za dobu za kterou se tyto rychlosti uplatní (t).

$$\Delta d = v \Delta t \sin \alpha$$

Přírůstek vzdálenosti od stěny za přírustek času.

$$d = d_0 + \sin \alpha \int_{t_1}^{t_2} v \, dt$$

Vzdálenost od stěny (d) lze vypočítat jako součet vzdálenosti původní (d_0), k níž se přičte $\sin \alpha$ vynásobený integrálem. Integrál představuje měnící se vzdálenost od stěny v závislosti na čase (t_1 ; t_2). Pomocí těchto vzorců lze vypočítat vzdálenost i úhel natočení vozidla vůči stěně.



Obr. 1.1: Úhel natočení

KAPITOLA 2

HW

Popis a specifikace použitého hardwaru pro realizace a výrobu autonomního pásového vozidla.

Arduino

Arduino jsem zvolil díky velkému množství rozšířujících desek, které jsem použil. Také z důvodu využití open source² licence pro software a hardware. Arduino má veřejné zdrojové kódy a schémata a každý je může upravit jak sám potřebuje. Také díky tomu se Arduino rychle rozšířilo po celém světě, je pro něj k dispozici velké množství dokumentace a příkladů použití.

Dle [Vod14] začal vývoj prvního Arduina v roce 2005, když se lidé z italské firmy Interaction Design Institute Ivrea rozhodli vytvořit jednoduchý a levný vývojový kit pro studenty, kteří neměli možnost si pořídit drahé kity jiných výrobců. Arduino se mezi studenty uchytilo a tak se tvůrci rozhodli poskytnout Arduino celému světu.

Řídicí jednotkou autonomního pásového vozidla je Arduino Leonardo s mikrokontrolérem ATmega32U4 firmy Atmel (viz. obr. 2.1). Leonardo samotné neobsahuje pouze procesor, ale i řadu dalších podpůrných obvodů např.: převodník z USB na rozhraní RS232 v napěťových úrovních 5 V a obvod pro stabilizaci napětí.

Leonardo má vstupy a výstupy dostupné na dutinkových lištách po obvodu (viz. obr. 2.2). Na Leonardu je také resetovací tlačítko, několik signalizačních LED diod: pro signalizaci připojeného napájecího napětí a pro signalizaci komunikace po sériové lince. Leonardo poskytuje celkem 14 digitálních pinů a 6 pinů analogových. Šest z digitálních pinů lze také použít na hardwarové PWM¹.

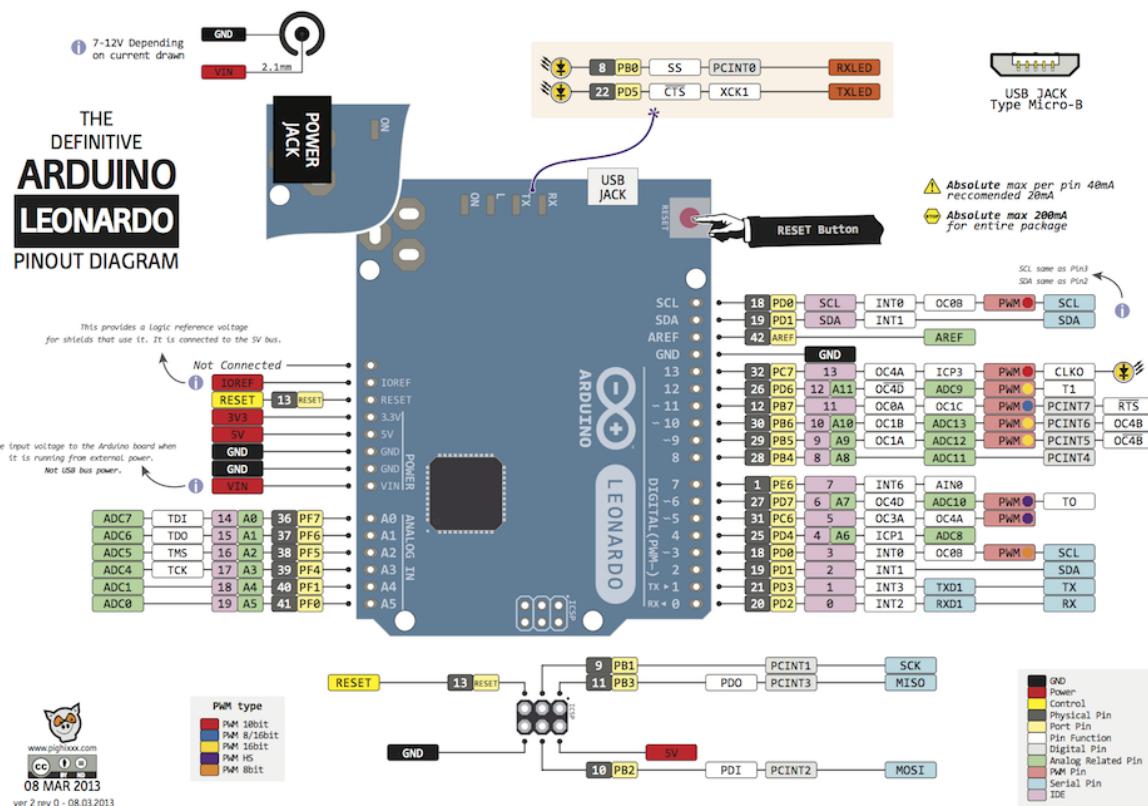
Arduino Leonardo lze napájet napětím od 6 V do 20 V, ale doporučené napájecí napětí je od 7 V do 12 V. Arduino lze napájet pomocí Micro USB, nebo pomocí Power Jack konektoru.

² Veřejné zdrojové kódy i schémata

¹ Pulzně šířková modulace



Obr. 2.1: Arduino Leonardo

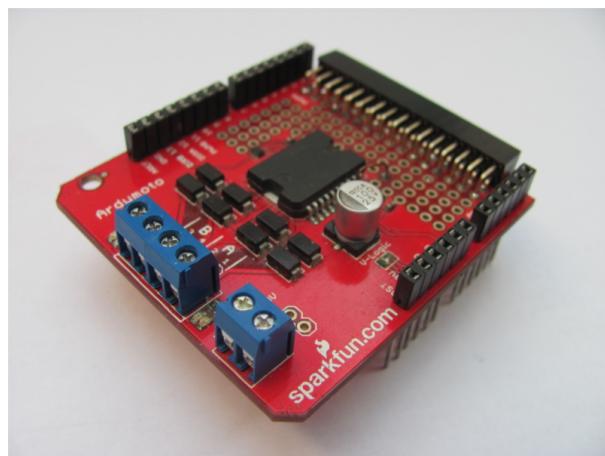


Obr. 2.2: Zapojení vstupů a výstupů desky Arduino Leonardo [Pig13]

Ardumoto

Jedná se o desku od firmy Sparkfun určenou pro Arduino k řízení motorů (viz. obr. 2.3). Tato deska dokáže řídit dva DC motory. Jádrem desky je dvoukanálový motorový driver L298P, díky tomu můžeme do každého kanálu (motoru) dodat až 2 A. Deska obsahuje dva páry LED (žlutá a modrá), které slouží na indikaci směru.

Deska umožňuje napájení buď z 3,3 V nebo z 5 V logiky. Ovládání směru motorů je připojeno na digitální výstupy 12 a 13. Ovládání rychlosti je na výstupech 3 a 11 (PWM).



Obr. 2.3: ArduMoto

LCD

Pokud potředujete ovládat LCD¹ 16x2 je většinou potřeba 6 výstupů na ovládání LCD a potom další 3 pro ovládání RGB². To je celkem 9 výstupů + napájení a zem, což je takřka polovina výstupů, které má k dispozici Arduino Leonardo.

Tohoto jsem se chtěl vyvarovat, protože bych potom neměl dostatek výstupů na ovládání motorů a připojení senzorů. Z tohoto důvodu jsem zvolil LCD desku od Adafruit (viz. obr. 2.4). Tato usnadňuje použití LCD 16x2. Díky LCD desce stačí pouze dva výstupy místo zmiňovaných devíti. To umožňuje komunikace přes standartní I2C sběrnici.

Pojezd

Pojezd je soustava dílů, které umožňují vozidlu se pohybovat. V tomto případě se jedná o pásový pojezd.

¹ Liquid crystal display

² Barevný model red-green-blue



Obr. 2.4: LCD shield od Adafruit

Pojezd se skládá ze základní desky na které vše stojí a je na ní vše upevněno. Samotný pohyb zajišťuje třístupňová převodovka na které se zařazen nejpomalejší převod. Z převodovky vedou dvě hnací šestihranné hřídele, které jsou zakončeny ozubenými koly, která pohání pásy.

Na základní desce pojezdu je společně s převodovkou upevněn rámeček na šest baterii AA. Také jsou k základní desce upevněny tři ultrazvukové senzory. Nejdůležitější je nástavba z Merkuru¹ připevněná k základní desce pro upevnění Arduina.

Senzor

Senzor HC-SR04 je ultrazvukový senzor určený k měření vzdálenosti. Senzor vyvolá vysokofrekvenční zvukové vlny a zachytává jejich odraz. Z časového intervalu mezi vyvoláním a přijmutím vln se za pomocí vzorce se vypočítá vzdálenost.

$$s = \frac{t}{58}$$

Ve výše uvedeném vzorci je s měřená vzdálenost a t doba odezvy na vyslaný ultrazvukový signál. Senzor HC-SR04 dokáže změřit vzdálenost od 3 cm do 400 cm. Senzor vysílá ultrazvukové vlny o frekvenci 40 Hz s rozptylem 15°. Senzor je napájen 5 V a jeho odběr je 15 mA. Příklad ovládání senzoru je popsán v [Gom11].

¹ Tradiční česká kovová stavebnice



Obr. 2.5: HC-SR04

KAPITOLA 3

SW

Popis a specifikace použitého softwaru pro řízení autonomního pásového vozidla.

Arduino IDE

Arduino má svoje vlastní vývojové prostředí tzv. IDE. Toto prostředí je naprogramované v jazyce Java. Toto prostředí podporuje programovací jazyk Wiring.

Arduino IDE je dostupné pro operační systémy Linux, OSX a Windows. Arduino IDE slouží k programování a nahrávání programu do vývojových desek Arduino. Arduino a jeho vývojové prostředí popisuje [\[Vod14\]](#).

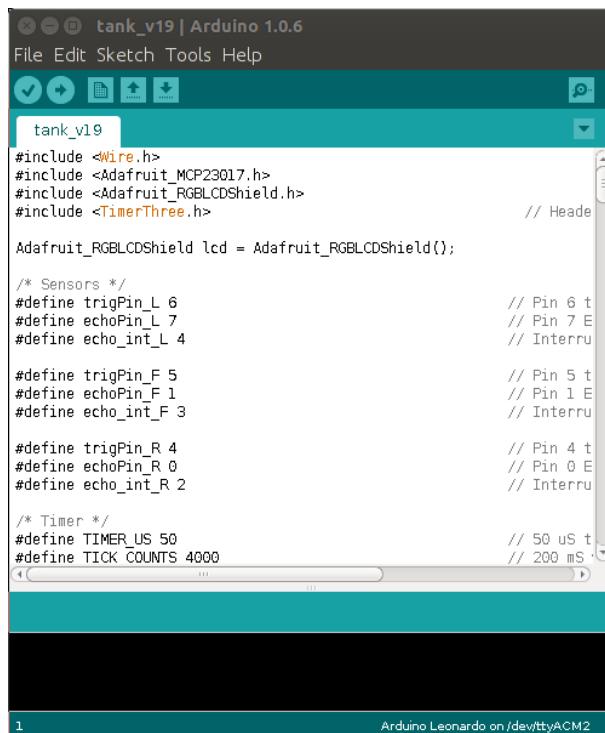
Wiring

Arduino lze programovat v jazyce C nebo C++. Výborným zdrojem informací o programování v jazyce C je [\[JB89\]](#) nebo [\[BWK06\]](#). Nejjednodušší je ale používat knihovnu Wiring. Tato knihovna je při programování Arduina tak rozšířená, že se o ní přestává mluvit jako o knihovně, ale začíná se říkat, že je to vlastní programovací jazyk.

Jako příklad se dají uvést dvě nejdůležitější funkce.

```
void setup() {  
    //Inicializace  
}
```

Funkce void *setup()* se volá pouze jednou, a to při připojení napájení k Arduinu. Do této funkce se píší většinou deklarace a nastavení.



Obr. 3.1: Arduino IDE 1.0.6 v Ubuntu 14.04

```
void loop() {
    //Hlavní smyčka
}
```

Funkce void *loop()* je funkce, která se volá hned po void *setup()*. Tato funkce běží pořád dokola, dokud má arduino napájení.

Obě výše uvedené funkce musí obsahovat každý program napsaný v jazyce Wiring.

Multitasking

Multitasking je schopnost systému provádět několik procesů současně. U autonomního vozidla je multitasking zapotřebí kvůli tomu, že potřebujete současně získávat hodnoty, zpracovávat hodnoty, rozhodovat se na základě získaných hodnot a podle rozhodnutí se pohybovat a případně ještě tyto parametry zobrazovat. Existují dva druhy multitaskingu.

Kooperativní multitasking

Kooperativní multitasking je založen na dostatečně rychlém volání jednotlivých úloh za sebou. Každá úloha je nucena po malé chvíli předat kontrolu systému a ten zase spustí jinou úlohu, která se musí spustit atd. Výhodou je jednodušší implementace systému a nižší požadavky na hardware. Nevýhodou je, že chybně naprogramovaná úloha způsobí úplné zastavení systému i ostatních úloh.

Preemptivní multitasking

V preemptivním multitaskingu rozhoduje o spouštění úloh (procesů) výhradně systém. V pravidelných intervalech pohybujících se okolo 100x až 1000x za sekundu dostane procesor přerušení od časovače a přeruší aktuálně prováděnou úlohu. Procesor nyní zváží priority úloh a jiné aspekty a na základě nich se rozhodne, zda spustí jinou úlohu, či nechá pokračovat stávající. Hlavní výhodou je, že i když se úloha zacyklí, tak po příznaku přerušení od daného časovače může tuto úlohu pozastavit a spustit jinou.

Více informací se můžete dozvědět na [\[Wik14\]](#).

Řízení

Řízení je činnost, kterou může provádět automatický regulátor. Regulátor má za úkol podle hodnoty na vstupu udržovat hodnotu regulované veličiny (teploty, hladiny, rychlosti, atd.).

V případě mého pásového vozidla je nutno toto aplikovat na bludiště. Bludiště má pouze jeden začátek a jeden konec a pouze pravoúhlé zatáčky. To znamená, že v bludišti mohu nastat pouze 4 situace:

- Jízda vpřed
- Otočení doleva
- Otočení doprava
- Zastavení

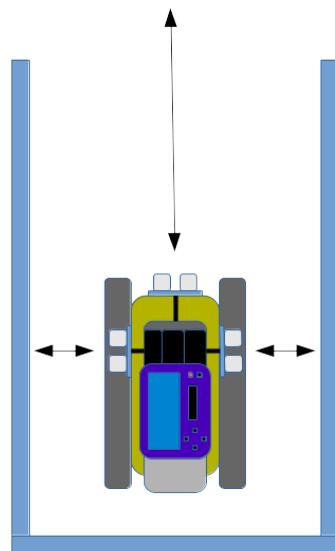
Po startu je důležité udržovat pásové vozidlo ve středu bludiště, nebo v konstantní vzdálenosti od stěny (viz [obr. 3.2](#)). Toto má na starosti regulátor přímé jízdy. Je to velice důležité, aby nedošlo ke kolizi se stěnou bludiště. Je to potřeba také proto, že nic není dokonalé ani motory a vozidlo nemusí jezdit přímočaře. Také je tím ošetřeno špatné nastavení vozidla před startem.

Po startu se vozidlo dříve nebo později dostane do situace, kdy už nebude moci jet rovně. V takovéto situaci musí zjistit, jaké má možnosti. Vyhodnotit ji může například tak, že na některé ze stran má dostatek místa k dalšímu postupu a další pohyb bude směrovat tím směrem (viz [obr. 3.3](#) nebo viz [obr. 3.3](#)). Po zjištění vzdálenosti a vyhodnocení se vozidlo otočí daným směrem o 90°.

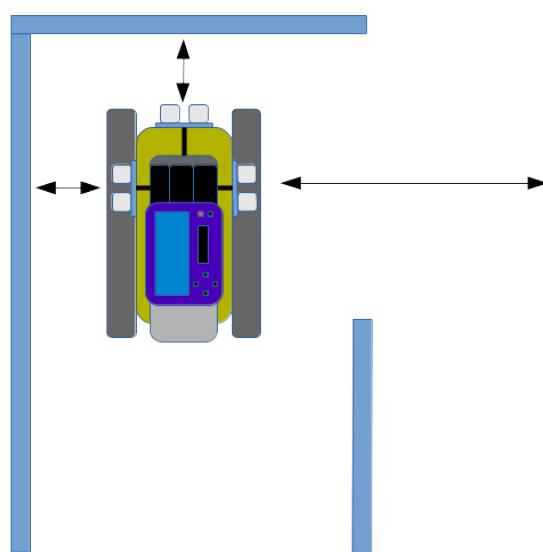
Po otočení přebírá opět řízení regulátor přímé jízdy, který se opět snaží udržet vozidlo uprostřed bludiště. Tento proces se může neustále opakovat. Změna nastane v okamžiku, kdy už vozidlo nebude mít dostatek prostoru k pokračování v jízdě (viz [obr. 3.5](#)). To znamená, že vozidlo je na konci bludiště a zastaví.

Regulátor přímé jízdy reguluje rychlosť otáček motorů na základě hodnot získaných ze senzorů. Regulátor je nastaven tak, že se snaží APV udržet uprostřed bludiště. Podle toho, na jakou stranu se reguluje, tak z té strany se odečítají hodnoty senzoru. Hodnoty senzorů se pro větší přesnost filtrovají, aby se odstranily nežádoucí hodnoty. Výsledný rozdíl vzdálenosti se K-krát zvětší a přičte se k rychlosći daného motoru.

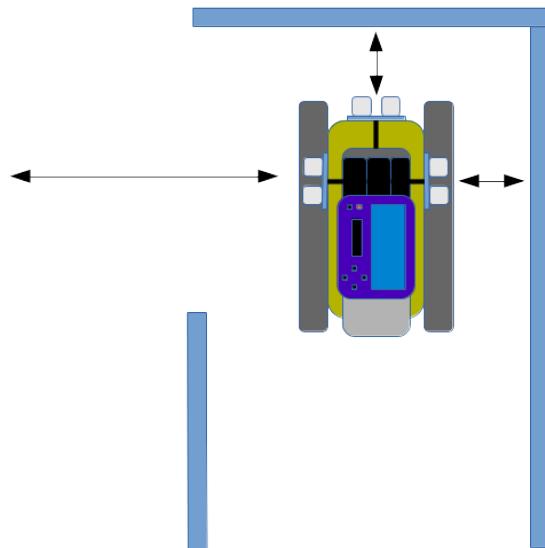
Velice důležité je znát také úhel natočení vůči straně, podle které regulátor reguluje (viz [Teoretický rozbor](#)). Díky tomu regulátor dokáže rychleji reagovat na vychýlení se ze středu bludiště.



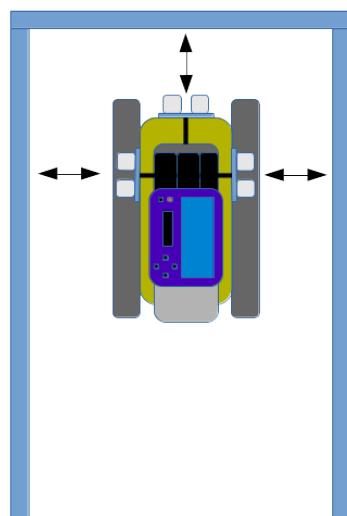
Obr. 3.2: Dopředu



Obr. 3.3: Doprava



Obr. 3.4: Doleva



Obr. 3.5: Stop

KAPITOLA 4

Výroba

V následující kapitole popisují konstruování a sestavování pásového vozidla. Také zde popíší jednotlivé části programu a jeho ladění.

Konstrukce

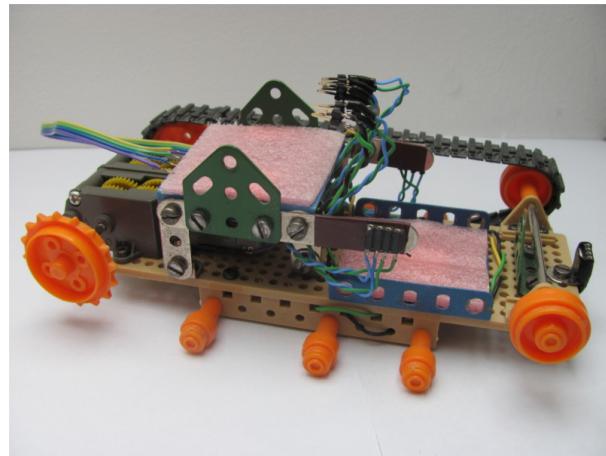
Konstruování je jedna z nejdůležitějších činností na většině projektů. Konstrukci pásového vozidla jsem přepracovával několikrát, většinou kvůli nevyhovujícímu umístění. Základem celé konstrukce Autonomního pásového vozidla je univerzální plastové pole 31x11 otvorů podobné stavebnici Merkur. K této základně je připevněno 5 os, z nichž je 6 hladkých a jedna hnací šestihranná. Všech pět os je připevněno plastovými díly z Universal plate set od firmy Tamiya, inc. Na koncích každé osy jsou dvě plastová kola určená k vedení a držení pásu a na koncích šestihranné osy jsou ozubená hnací kola určená k převodu otáček převodovky na pásy.

Na jeden okraj tohoto plastového skeletu je pomocí dvou šroubků připevněna převodovka Twin-motor gearbox také od firmy Tamiya, inc (viz obr. 4.1). Tato převodovka převádí točivý moment dvou DC motorů pomocí dvou navzájem oddělených soustav ozubených kol a hřídele do pásů. Převodovka umožňuje převod na tři rychlosti označenými písmeny A-C od nejrychlejšího převodu po nejpomalejší. Pro pohon pásového vozidla není ani tak potřeba rychlosť, jako síla. Proto využívám třetí převod C.

Nad převodovkou je za pomoci dílů stavebnice Merkur vytvořena plošina pro Arduino (viz obr. 4.2). Stavebnice Merkur a Tamiya jsou velice podobné. Hlavním rozdílem je hmotnost jednotlivých dílů stavebnice. Toto je kvůli tomu, že Merkur je kovový oproti Tamiji. Důležité je ovšem to, že obě stavebnice mají stejnou rozteč otvorů pro šrouby i když jinou hustotou dírek. Nahoru na plošinu je za pomoci oboustranné lepící pásky připevněna pěnová podložka, která má dvě funkce. První, izolovat Arduino od kovové konstrukce aby nedošlo ke zkratování vývodů na spodní části Arduina. Druhá, zafixovat Arduino, aby se při jízdě a nárazech nepohybovalo.

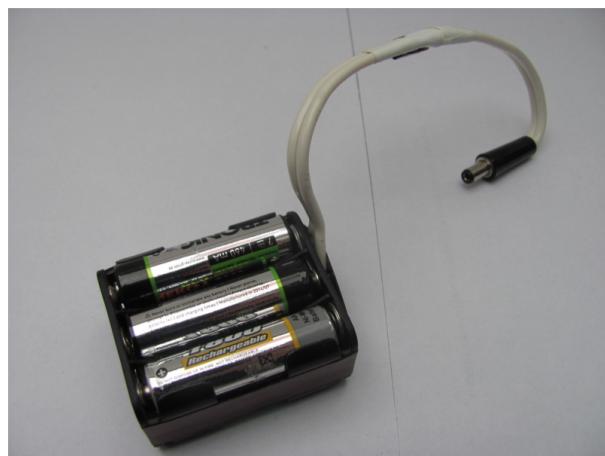


Obr. 4.1: Převodovka Twin-motor gearbox



Obr. 4.2: Skelet APV

V přední části plastového skeletu je upevnění napájení pásového vozidla opět za pomocí stavebnice Merkur. Na povrchu je opět pěnová podložka jako izolace a tlumič nárazů. Napájení je tvořeno plastovou klecí 3x2 na články velikosti AA (viz obr. 4.3). Výhodnější je využít nabíjecí akumulátory z důvodů časté výměny. Nabíjecí akumulátory mají napětí 1,2 V. Celé autonomní vozidlo je tedy napájeno 7,2 V DC¹. Napájení je přivedeno do Arduina pomocí bílé dvoulinky zakončené souosým konektorem. Vodič s kladnou polaritou jsem přerušil a propojil pomocí klasického dvoupolohového vypínače. Je to kvůli častému odpojování a připojování napájení aby se neničil konektor Arduina a z důvodu snazšího vypnutí a zapnutí.



Obr. 4.3: Napájení APV

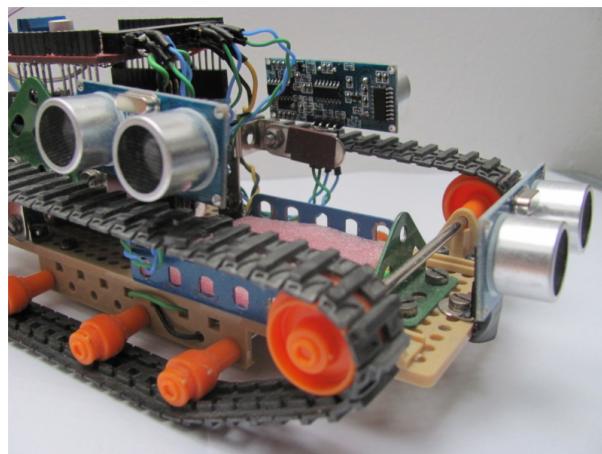
Další důležitá věc je, kde a jak upevnit ultrazvukové senzory HC-SR04. Senzory musí být umístěny tak, aby se ultrazvuková vlna neodrazila od konstrukce APV. Na orientaci v prostoru jsou zapotřebí minimálně tři senzory, z nichž jeden vepředu a dva na stranách. Přední senzor je umístěn doprostřed přední části skeletu na délku opět ze stavebnice Merkur. Na tento délku je připevněna dutinková lišta, do které se dá jednoduše zasunout senzor. Zbylé dva jsou připevněny na kovové konstrukci držící Arduino, každý jedné straně. Stejně jako přední senzor tak i postranní senzory se zasouvají do dutinkové lišty. Postranní senzory jsou umístěny tak, že vysílač a přijímač ultrazvuku jsou zarovnány rovnoběžně s pásy. Díky tomu se vlny neodráží od pásu a zároveň při pohybu blízko překážky nehrozí poškození ani ulomení senzoru.

Jak jsem již zmiňoval, na kovové konstrukci nad převodovkou je umístěno Arduino Leonardo do Arduina je zasunuta deska Ardumoto a do ní ještě LCD display. Problém je, jak vyvést signály a napájení z senzorů. Výstupy Arduina nejsou vyvedeny z LCD desky. Dalo by se říci, že je to koncový stupeň. K vývodu výstupů jsem musel využít část universálního pájivého pole, které je umístěno na desce Ardumoto. Na okraj tohoto pole jsem zapájal úhlovou dutinkovou lištu 16x2. Pomocí vodičů jsem na tuto lištu vyvedl několik napájení a zemí a potřebné výstupy.

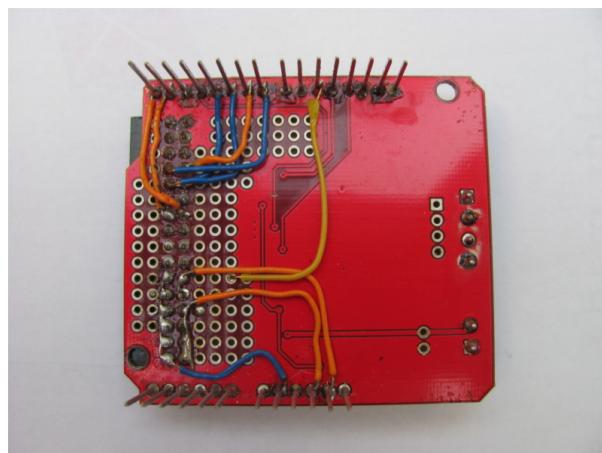
Legenda k úhlové liště s umístěním a typem výstupu je zobrazena na obr. 4.6. V legendě jsou uvedeny všechny výstupy Arduina Leonarda, ale zapojeny jsou jen potřebné. Zapojeno je napájení a z digitálních 0 a 1 a potom 4-7. Zbylé piny nevyužívám, a proto nejsou zapojeny.

Každý ze tří senzorů je připojen pomocí modrých a zelených vodičů zapojených do úhlové

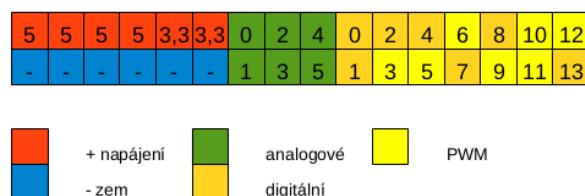
¹ Stejnosměrný proud



Obr. 4.4: Upevnění senzorů na APV

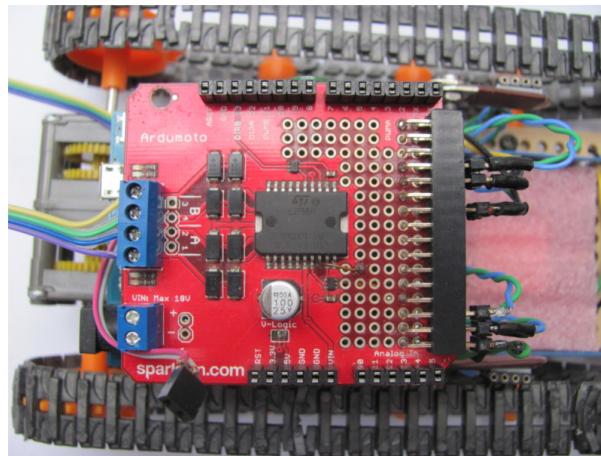


Obr. 4.5: Propojení úhlové patice



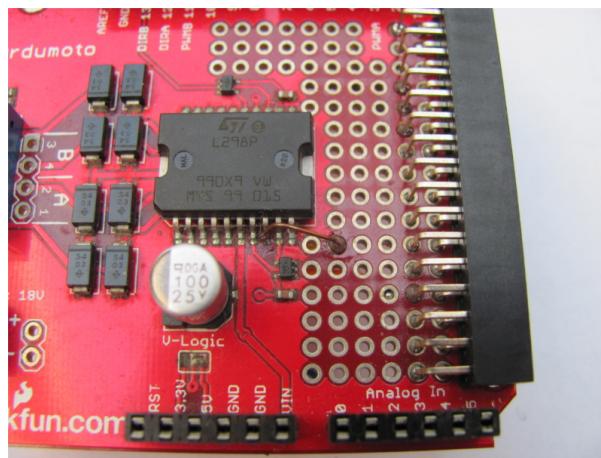
Obr. 4.6: Legenda k zapojení úhlové patice

patice (viz obr. 4.7). Všechny dráty jsou pečlivě zapleteny do plastovokovové konstrukce. Zapletení drátů má více důvodů. Prvně z důvodu bezpečnosti, protože kdyby pásové vozidlo jelo přes drsnější terén, mohly by se volně visící kabely o tento terén zachytit. Za druhé z důvodu estetiky a za třetí z důvodu úspory místa. V poslední řadě je zapotřebí vyřešit napájení motorů. Toto je vyřešeno pomocí plochého čtyřvodičového kabelu připojeného do svorek na Ardumoto desce.



Obr. 4.7: Připojení kabeláže

Nakonec je potřeba vyřešit na hardwarové stránce komunikaci Arduina, Ardumoto desky a LCD desky. Obvody na LCD desce komunikují s Arduinem přes digitální výstupy 2 a 3. Ardumoto deska taky používá výstup 3 a to na řízení rychlosti motoru A pomocí PWM. Přes jeden výstup se nemůže provádět dvojí komunikace. Proto jsem přerušil zalamovacím nožem na Ardumoto desce vyleptanou cestu vedoucí k výstupu 3 a přepojil jsem ji na výstup 10, který ho dokáže plně nahradit (viz obr. 4.8).



Obr. 4.8: Přepojení PWMA na PWM 10

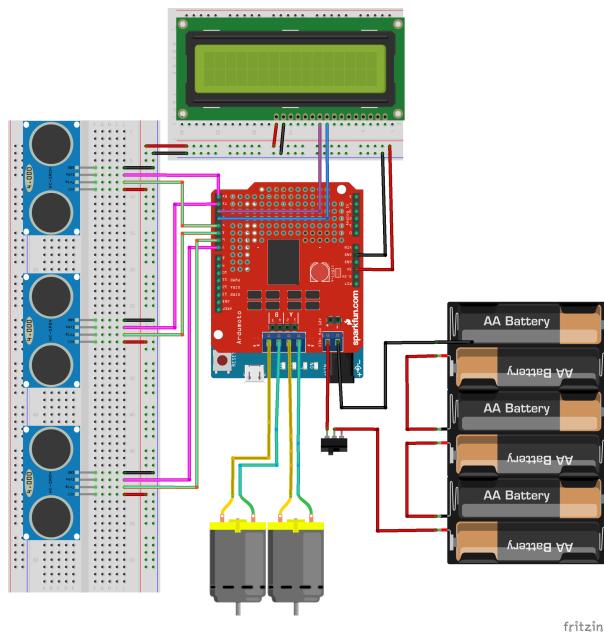
Tím je uvolněna komunikace displaye i Ardumoto desky. Display komunikuje pomocí výstupů 2 a 3, ale pouze s Arduinem UNO, ale s Arduinem Leonardo komunikuje za pomocí krajních výstupů SDA a SCL. Ardumoto je primárně děláno na Arduino Uno, kvůli tomu nemá na sobě tyto výstupy vyvedeny. Proto jsem musel vyrobit propojku z ploché dvoulinky propojující

na přímo výstupy SDA a SCL na Arduinu s LCD deskou. Díky tomu nyní dokáží obě desky nezávisle na sobě komunikovat s Arduinem. Celá sestava je zobrazena na obr. 4.9. Sestava je umístěna na plošině nad převodovkou.



Obr. 4.9: Kompletní sestava řídící jednotky

Máme kompletní a funkční řídící jednotku. Zapojení senzorů motorů a napájení je znázorněno na blokovém schématu na obrázku 4.10. Senzory HC-SR04 jsou popsány viz kap. [Senzor](#). Pravý senzor má Trigger² připojený na digitální výstup 4 a Echo³ na 0. Přední senzor má Trigger připojený na 5 a Echo na výstup 1. Levý senzor má Trigger připojen na 6 a Echo na 7. Výstup Echo každého senzoru musí být připojeno na digitální výstupy s přerušením. Kompletní schéma vytvořené v programu Fritzing⁴ je zobrazeno viz obr. 4.10.



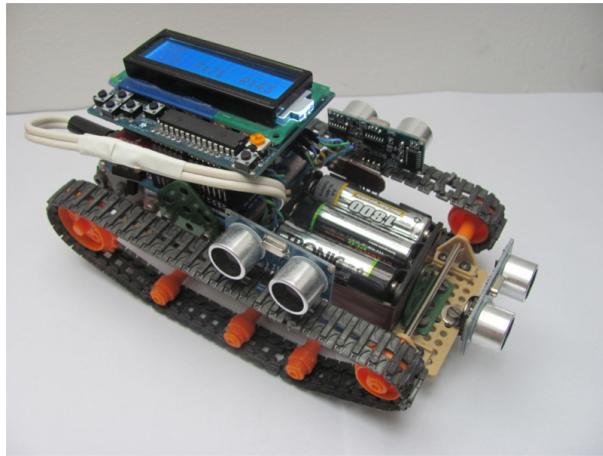
Obr. 4.10: Blokové schéma zapojení

² Vstup

³ Výstup

⁴ Open source program pro vytváření návrhů obvodů a DPS

Celé pásové vozidlo jehož konstrukci a zapojení jsem nyní popsal je zobrazeno viz obr. 4.11. Pásové vozidlo má rozměry šxdxv 10,5x20x10 [cm] a hmotnost přesně 600 g i s šesti akumulátory.



Obr. 4.11: Kompletní konstrukce Autonomního vozidla

V poslední části této podkapitoly se budu věnovat bludišti. Bludiště by mělo být pevné, ale zároveň pružné, aby při náhlém nárazu se vozidlo nepoškodilo. Mělo by být skladné a přenosné. Z těchto důvodu jsem si jako materiál pro výrobu bludiště vybral fitness puzzle koberec, který je pružný a snadno sestavitelný. Fitness koberec se skládá ze středových čtvercových dílů o rozměrech 30x30 [cm] a z okrajových dílů ve tvaru obdélníků o rozměrech 8x30 [cm]. Tím je stanovena i má šířka bludiště (30 cm) a také délka bludiště, která činí cca 270 cm. Bludiště se skládá za dvou levotočivých a ze dvou pravotočivých zatáček a tří rovinek (viz obr. 4.12).



Obr. 4.12: Bludiště

Program

Úkolem programu je získat hodnoty ze senzorů na základě těchto hodnot vyhodnotit situaci v jaké se vozidlo nachází a podle toho jednat. Celý kód je napsaný jazykem Wiring v Arduino

IDE. Arduino IDE i Wiring jsou popsány viz kap. SW. V této kapitole jsou popsány jednotlivé části kódu a jejich význam a funkce.

Na ovládání LCD desky jsou potřeba dvě knihovny. Jedna je na ovládání samotného LCD displaye a druhá na odvládání obvodu, skrze který můžeme ovládat LCD display pouze pomocí I2C sběrnice. Také je potřeba knihovna na řízení časovače 3 využívaného na měření vzdálenosti skrze ultrazvukové senzory.

```
#include <Wire.h>
#include <Adafruit_MCP23017.h>          // Knihovna na ovladani obvodu
→na LCD
#include <Adafruit_RGBLCDShield.h>        // Knihovna na ovladani LCD
→displaye
#include <TimerThree.h>                   // Knihovna na ovladani
→casovace 3
```

Ve svém kódu hodně využívám definice. Zpřehledňují kód a činí ho čitelnějším a zlepšují orientaci v něm. Využívám je ve *switch()*¹ na pojmenování stavů.

```
#define STATE_BEG 0                      // Stav pocatek
#define STATE_FORW 1                     // Stav jed dopredu
#define STATE_REG 2                      // Stav reguluj
#define STATE_TURN_RIGHT 3                // Stav zatoc doprava
#define STATE_TURN_LEFT 4                 // Stav zatoc doleva
#define STATE_STOP 5                     // Stav stuj
#define STATE_OTOC 6                     // Stav otoc
#define STATE_TURN_AROUND 7               // Stav otoc se na druhou
→stranu
#define STATE_ROV 8                      // Stav rovne
#define STATE_WAIT 9                     // Stav cekej
```

Ve funkci *setup()* nastavuji vstupy a výstupy senzorů, nastavuji přerušení, časovač, display a nastavuji jednotlivé úlohy plánovače.

```
void setup()
{
    pinMode(trigPin_L, OUTPUT);
    pinMode(echoPin_L, INPUT);
    // Nastaveni vstupu a vystupu u leveho ultrazvukoveho senzoru

    Timer3.attachInterrupt( timerIsr );
    // Nastaveni preruseni od casovace 3

    attachInterrupt(echo_int_L, echo_interrupt_L, CHANGE);
    // Nastaveni preruseni od leveho senzoru, ktore bude reagovat na
→zmenu

    lcd.begin(16, 2);                  // Nastaveni displaye

    pinMode(pwm_a, OUTPUT);           // Rychlosť motoru A
```

¹ Přepínač

```

pinMode(pwm_b, OUTPUT);           // Rychlosť motoru B
pinMode(dir_a, OUTPUT);          // Smer motoru A
pinMode(dir_b, OUTPUT);          // Smer motoru B

digitalWrite(dir_a, HIGH);        // Primarni smer dopredu
digitalWrite(dir_b, HIGH);        // Primarni smer dopredu

//-----
// Ukázka struktury úlohy měření pro plánovač //
//-----


stTasks[TASK_MER].iTaskID = TASK_MER;
stTasks[TASK_MER].lPeriod = 300;    // Perioda opakovani
stTasks[TASK_MER].lNextRun = 50;    // Dalsi spusteni
stTasks[TASK_MER].lDuration = 0;    // Doba trvani
stTasks[TASK_MER].bType = TYPE_CYCLIC; // Typ
stTasks[TASK_MER].bState = STATE_RUNNING; // Stav
stTasks[TASK_MER].fnCallBack = &fnTask_MER;
}

```

Ve funkci *loop()* se volá jediná funkce a to funkce plánovač, která je nadřazená ostatním funkcím.

Funkce plánovač *fnScheduler()* je funkcí kooperativního multitaskingu (viz *SW*). Při řízení APV je vyžadováno využití některého z multitaskingů, kvůli potřebě vykonávání několika úloh (procesů) naráz. Nepřetržitě musíme měřit vzdálenosti, tyto vzdálenosti musíme vyhodnotit a zobrazit a dát impulz motorům. To vše je nutno provézt v co nejkratším čase a nezávisle na sobě. Preemptivní multitasking nepřichází v úvahu kvůli nevyhovujícímu hw. Plánovač prochází seznam úloh a na základě plánovaného času spouštění vybere úlohu, která se má spustit a provede její spuštění.

```

void fnScheduler()
{
    unsigned long lNow = millis();
    // Zjisteni aktualniho casu
    unsigned long lTmp;
    for(int i=0; i<TASKS_NUM; i++) {
        // Postupne zkонтroluje vsechny ulohy
        if(stTasks[i].bState == STATE_RUNNING) {
            // Kdyz je uloha spustena, pokracuje
            if(stTasks[i].lNextRun <= lNow) {
                // Kdyz cas dalsiho spusteni se shoduje, pokracuje
                stTasks[i].lLastRun = lNow;
                // Do posledniho spusteni priradi aktualni cas
                lTmp = micros();
                // Do promenne priradi aktualni cas
                stTasks[i].fnCallBack(stTasks[i].iTaskID);
                stTasks[i].lDuration = micros() - lTmp;
                if(stTasks[i].bType == TYPE_CYCLIC) {
                    // Kdyz je uloha cyklicka, pokracuje

```

```
    stTasks[i].lNextRun += stTasks[i].lPeriod;
    // K dobe dalsiho spusteni pricte periodu
}
else {
    stTasks[i].bState = STATE_SLEEPING;
    // Jinak ulohu uspi
}
}
}
}
}
```

U úloh, které se spouštějí pouze jednou, provede plánovač jejich uspání. U APV je vždy nejdůležitější, aby mělo vždy dostatek dat podle kterých se může rozhodovat. Tyto data, neboli hodnoty získává ze tří ultrazvukových senzorů (viz. *Konstrukce*). Každý výstup Echo je připojen na přerušení, které reaguje na změnu.

```
void echo_interrupt_L()                                // Funkce obsluhujici int L
→senzoru
{
    switch (digitalRead(echoPin_L))                  // Kontrola log1 nebo log0
    {
        case HIGH:                                  // Pro log1
            echo_end_L = 0;                         // nastavi na 0
            echo_start_L = micros();                // predá čas od spuštění
            break;

        case LOW:                                   // Pro log0
            echo_end_L = micros();                 // nastavi čas od spuštění
            echo_duration_L = echo_end_L - echo_start_L;
            // celková doba získaná odečtením začátku od konce
            break;
    }
}
```

Arduino po spuštění, nebo restartu automaticky počítá jak dlouho je spuštěné. Díky tomu se za pomoci funkce *micros()*² dá jednoduše zjistit, jak dlouho trvala cesta ultrazvuku od vysílače k příjmači. A následně vypočítat podle vzorce výslednou vzdálenost (viz *Senzor*).

```
void GetSensorValues()                                // Funkce pro výpočet
→vzdálenosti
{
    noInterrupts();                               // Zakázání přerušení
    lL = echo_duration_L;                       // Doba cesty ultrazvuku
    interrupts();                                // Povolení přerušení
    lL = lL/58;                                  // Výpočet vzdálenosti pro L
    →sen.
}
```

² Funkce z knihovny Wiring

Naměřené vzdálenosti se zobrazují na LCD display. Zobrazování se provádí ve funkci *zobraz()*. Díky knihovnám je ovládání display velmi jednoduché. Funkcí *lcd.setCursor()* nastavíte pozici kurzoru a funkcí *lcd.print()* zobrazíte.

```
void zobraz()
{
    lcd.setCursor(0, 0);                                // Nastavení kurzoru
    lcd.print("S:");
    lcd.print(state_dis);                            // Zobrazení aktuálního stavu
    ↳dis.
    lcd.setCursor(0, 1);                                // Nastavení kurzoru
    lcd.print("L:");
    getCharNum(lL, sL);                             // Převedení na řetězec
    lcd.print(sL);                                    // Zobrazení vzdálenosti L sen
```

}

Duležitou část tvoří úloha regulátoru rovné jízdy. Popis a funkce regulátoru je popsána v kap. [Řízení](#). Následuje krátká ukázka regulátoru.

```
iErr1 = L_SET - lL;                                // Rozdíl vzdálenosti vlevo
iErr2 = fnUhel(pole_l);                            // Úhel natočení
if(iErr1 < 0) {                                  // Když je moc daleko
    speed_motor_l = NORMAL_SPEED_L + iErr1 * K;
    // Zpomal levý → zatoc doleva
    speed_motor_r = NORMAL_SPEED_R;
    // Pravý normální rychlostí
}
else if (iErr1 > 0) {                            // Když je moc blízko
    speed_motor_l = NORMAL_SPEED_L;                // Levý normální rychlostí
    speed_motor_r = NORMAL_SPEED_R - iErr1 * K;
    // Zpomal pravý → zatoc doprava
}
```

K rovné jízdě je potřeba znát odchylku od správné vzdálenosti a úhel natočení vůči stěně.

Nejdůležitější částí celého programu je *Dispečer*, který je realizovaný pomocí příkazu *switch()*. V hlavním stavu dispečera (*STATE_BEG*) se rozhoduje na základě hodnot získaných ze senzorů o dalším pohybu APV.

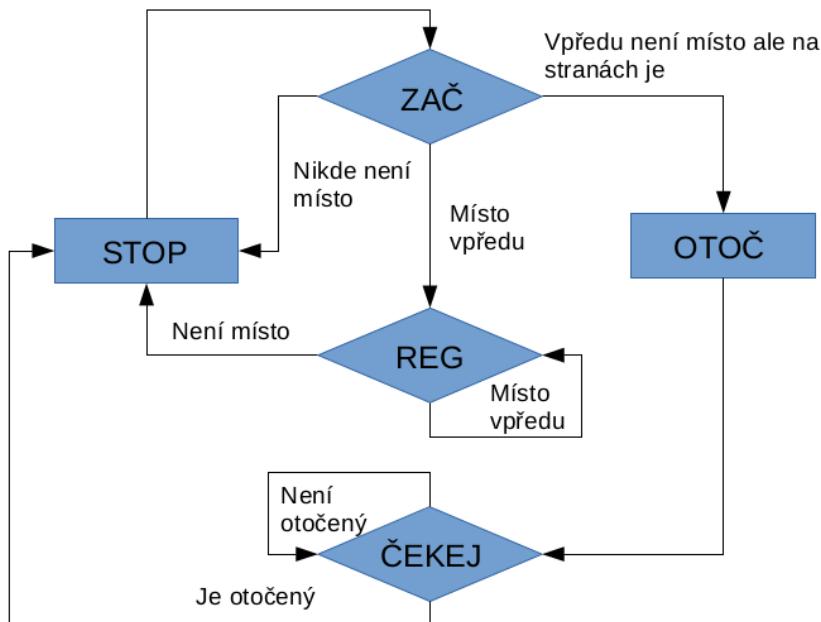
```
switch(state_dis)
{
case STATE_BEG:
    if(lF>MIN_DELKA_F) {
        state_dis = STATE_FORW;
    }
    else{
        if(lR>MIN_DELKA_S)
            state_dis = STATE_TURN_RIGHT;
        if(lL>MIN_DELKA_S)
            state_dis = STATE_TURN_LEFT;
        if(lR<=MIN_DELKA_S && lL<=MIN_DELKA_S)
```

```

        state_dis = STATE_STOP;
    }
break;

```

Kompletní *Dispečer* se všemi stavami je vidět na obr. 4.13. V hlavním bloku (ZAČ) se rozhoduje, jaká operace se bude provádět. Ukázka je k vidění výše. Po vyhodnocení situace pojede APV bud' dopředu, nebo začne zatáčet, nebo bude stát na místě.



Obr. 4.13: Vývojový diagram dispečeř

Ladění

Ladění je jedna z nejdůležitějších částí každého projektu i toho sebemenšího. Máme kompletní konstrukci i program, ale je jen velmi malá pravděpodobnost, že vozidlo bude dělat to co potřebujeme. Je mnoho konstant a proměnných, které jsou navzájem provázané a mnoho situací a povrchů. Při ladění Autonomního pásového vozidla jsem provedl bezpočet testů aby jsem docílil co nejlepšího výsledku.

Rovná jízda

Aby vozidlo jelo rovně musíme zahrnout několik aspektů. K těm patří např.: tření a prokluzování pásů na povrchu, konstrukční nedokonalosti, různé výkony motorů a další. Při testovacích jízdách jsem zjistil, že APV se chová jinak na hladkém povrchu (parkety) a jinak na hrubším (koberec). Na hladkém povrchu pásy prokluzují a není možné docílit zcela rovné jízdy. Při docílení relativně rovné jízdy na hladkém povrchu se při přechodu na jiný povrch se APV chová

jinak. Naopak na hrubším povrchu mají pásy APV lepší záběr a rozdíl tvoří už jen nevyrovnané výkony motorů. Kvůli tomu bludiště nemá pouze stěny, ale i dno, aby bylo docíleno co nejstejnорodějšího povrchu o stejné hrubosti.

Otáčení

Dalším důležitým bodem je dobré otočení v bludišti. Při testování vyplynulo, že do zatáčky APV takřka nikdy nenajede dokonale rovně. Vozidlo se vždy otáčí cca. o 90° , což znamená, že tuto odchylku získá i při výjezdu ze zatáčky.

Zastavování

Pro dokonalé projekční je důležité mít nastavenou vhodnou vzdálenost pro zastavení. Vzdálenost musí být taková, aby po otočení se o 90° bylo vozidlo opět blízko středu a zároveň, aby vozidlo při otáčení nevrazilo do stěny bludiště. K této vzdálenosti jsem se po sérii testů dostal, ale z důvodu rychlosti jakou dispečer¹ kontroluje vzdálenosti.

¹ Část programu vyhodnocující aktuální situaci

KAPITOLA 5

Závěr

Cílem této práce bylo navrhnut, sestrojit a naprogramovat Autonomní pásové vozidlo orientující se samo ve volném nebo ohraničeném prostoru. Toto vozidlo je schopno se řídit podle svislých stěn a tak se dostat do potenciálního cíle. Řídící jednotkou vozidla je Arduino Leonardo které se za pomocí hodnot ze 3 ultrazvukových senzorů řídí pásové vozidlo.

V průběhu práce jsem ověřil základní principy programování mikrokontrolérů, funkci ultrazvukového snímače vzdálenosti a řízení stejnosměrného motoru pomocí pulzně šířkové modulace. Funkce vozidla byla průběžně ověřována na testovací dráze.

Při testování jsem zjistil, že pro reálné využití vozidla by bylo potřeba využít senzory s vyšší přesností a na každé straně vozidla umístit senzory dva. To by umožnilo snadnější určení polohy a orientace vozidla v prostoru. Zvýšení počtu senzorů by si rovněž vyžádalo použití výkonnějšího mikrokontroléru.

KAPITOLA 6

Zkratky a veličiny

APV Autonomní pásové vozidlo

DC Stejnosměrný proud

V Volt

AA Tužková baterie

LCD Display z tekutých krystaků

IDE Vývojové prostření

I/O Vstupy a výstupy

LED Elektroluminiscenční diody

PWM Pulzně šířková modulace

USB Univerzální sériová sběrnice

A Ampér

mA Miliampér

RGB Barevný model

I2C Sériová sběrnice

cm Centimetr

Hz Hertz

° Stupeň

g Gram

Seznam obrázků

1.1	Úhel natočení	2
2.1	Arduino Leonardo	4
2.2	Zapojení vstupů a výstupů desky Arduino Leonardo [<i>Pig13</i>]	4
2.3	ArduMoto	5
2.4	LCD shield od Adafruit	6
2.5	HC-SR04	7
3.1	Arduino IDE 1.0.6 v Ubuntu 14.04	9
3.2	Dopředu	11
3.3	Doprava	11
3.4	Doleva	12
3.5	Stop	12
4.1	Převodovka Twin-motor gearbox	14
4.2	Skelet APV	14
4.3	Napájení APV	15
4.4	Upevnění senzorů na APV	16
4.5	Propojení úhlové patice	16
4.6	Legenda k zapojení úhlové patice	16
4.7	Připojení kabeláže	17
4.8	Přepojení PWMA na PWM 10	17
4.9	Kompletní sestava řídící jednotky	18
4.10	Blokové schéma zapojení	18
4.11	Kompletní konstrukce Autonomního vozidla	19
4.12	Bludiště	19
4.13	Vývojový diagram dispečer	24

Literatura

- [BWK06] Dennis M. Ritchie, Brian W. Kernighan. *Programovací jazyk C*. Computer press, 2006.
- [Cre85] A. P. Creedy. *Skid Steering of wheeled and tracked vehicles - analysis with coulomb friction assumptions*. Maribyrnong Victoria, 1985.
- [FR14] Jin shi Chen, Feng Ren, Xin hui Liu. Analysis of Skid Steer Loader Steering Characteristic. *Advances in Mechanical Engineering*, 2014.
- [JB89] Luděk Skočovský, Jan Brodský. *Operační systém Unix a jazyk C*. SNTL, Praha, 1989.
- [Ryb99] Jiří Rybička. *LATEX pro začátečníky*. Konvoj Brno, 1999.
- [Sta88] Jaroslav Starý. *Mikropočítáč a jeho programování*. SNTL, Praha, 1988.
- [Vod14] Zbyšek Voda. *Průvodce světem Arduina*. HW Kitchen, 2014.
- [Ele14] Elecfreaks. Ultrasonic Ranging Module HC - SR04. [online], Naposledy aktualizováno 17. 2. 2014. URL: <http://www.micropik.com/PDF/HCSR04.pdf>.
- [Fri15] Limor Fried. RGB LCD Shield. [online], Naposledy aktualizováno 16. 4. 2015. URL: <https://learn.adafruit.com/rgb-lcd-shield/overview>.
- [Gom11] Davide Gomba. HC-SR04 Ultrasonic Arduino library. [online], 20. 10. 2011. URL: <http://blog.arduino.cc/2011/10/20/hc-sr04-ultrasonic-arduino-library/>.
- [Pig13] Pighixxx. Arduino Leonardo Pinout v2. [online], 8. 3. 2013. URL: <http://pighixxx.tumblr.com/post/44858834224/arduino-leonardo-pinout-v2>.
- [pseait15] Střední průmyslová škola elektrotechnická a informačních technologií. Logo. [online], 2015. URL: <https://moodle.sspbrno.cz/course/view.php?id=387>.
- [Wik14] Wikipedie. Multitasking. [online], Naposledy aktualizováno 17. 2. 2014. URL: <http://cs.wikipedia.org/wiki/Multitasking>.

Rejstřík

Symbols

°, [27](#)

A

A, [27](#)

AA, [27](#)

APV, [27](#)

Arduino, 3

Arduino Leonardo, 3

C

cm, [27](#)

D

DC, [27](#)

G

g, [27](#)

H

HC-SR04, 6

Hz, [27](#)

I

I/O, [27](#)

I2C, [27](#)

IDE, [27](#)

L

LCD, [27](#)

LED, [27](#)

M

mA, [27](#)

Merkur, 13

Multitasking, 9

P

PWM, [27](#)

R

RGB, [27](#)

U

USB, [27](#)

V

V, [27](#)

W

Wiring, 8