Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Autonomní řízení skupiny robotů a jejich reakce na neočekávané podněty

Bakalářská práce

Lukáš Vencálek

Vedoucí práce: PhDr. Milan Novák, Ph.D.

České Budějovice 2015

Bibliografické údaje

Vencálek L., 2015: Autonomní řízení skupiny robotů a jejich reakce na neočekávané podněty. [Autonomous control of group of robots and their reaction to unexpected stimuli. Bc.. Thesis, in Czech.] – 49 p. , Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Bakalářská práce se zabývá sestrojením robotické modelové skupiny, se schopností se autonomně pohybovat prostorem za účelem dosažení cíle nebo pohybu v definovaném směru. Skupina musí být schopna reagovat na neočekávanou překážku v trase tak, aby nedošlo k narušení její integrity a členové skupiny se navzájem neomezili. Poté musí být skupina schopna nalézt původní trajektorii trasy a pokračovat v pohybu.

Annotation

The bachelor thesis deals with creating of mobile robotics group, with an abilities of autonomous movement to reach a target of a route or movement in defined direction. Group must be able to react to unexpected obstacle in the route to not disrupt its integrity and avoid robots to restrict each other. The group must be able to find the original trajectory of the route and continue movement.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne ..... Podpis autora .....

Poděkování

Děkuji PhDr. Milanu Novákovi, Ph.D. za cenné rady, odborný dohled a čas věnovaný vedení této práce. Dále děkuji přítelkyni a rodině, za podporu při studiu.

Obsah

Úvod

V důsledku technologického pokroku v oboru autonomních robotických jednotek se paralelně zvyšují nároky na kooperaci robotických skupin a zpřísňují se požadavky, týkající se reakcí robotů na podněty, které nebyly konkrétně definovány v řídící logice daných robotů, tzv. autonomita robotů.

Základní myšlenkou této práce je za pomocí vhodné kombinace hardwarových a softwarových komponent vytvořit skupinu robotů, která se bude autonomně pohybovat prostorem, přičemž bude tento prostor analyzovat, za účelem dosažení uživatelem definovaného cíle nebo směru a algoritmizace reakcí dané skupiny na podněty, které narušili trasu jejich cesty tak významně, že by na ně nebylo možné bez aplikace logiky definující řešení problému neočekávaných překážek zareagovat, aniž by byla narušena integrita skupiny nebo by docházelo k nedefinovaným stavům mezi jednotlivými roboty. Po vykonání algoritmu řešení překážek se od robotů očekává, že budou pokračovat v původně naplánované trajektorii trasy a dosáhnou cíle.

Cíle práce

Cílem práce je na základě analýzy dostupných řešení řízení pohyblivých skupin a řešení reakcí pohyblivých robotických skupin nebo jednotek, na neočekávané podněty v trase, vytvořit algoritmizaci pohybového, reakčního, kooperačního a komunikačního systému skupiny. Případně již hotová řešení vylepšit a vytvořit funkční celek - autonomně se pohybující robotickou skupinu, se schopností reagovat na překážky, které by bez aplikace reakční logiky zabránily robotům dosáhnout cíle.

Řešení zajistí v každém momentu pohybu integritu skupiny, bez nutnosti kooperace s nadřazeným navigačním systémem. Roboti nebudou kvůli univerzálnosti používat navigační systém GPS, protože by se tím značně omezilo pole jejich užití.

Algoritmizace reakce na neočekávaný podnět v trase skupiny zajistí deterministickou reakci. Skupina se bude schopna objektu, jenž nebyl při plánování trasy její součástí, vyhnout, vrátit se do původní dráhy a pokračovat v procesu dosažení cíle. Reakci na neočekávaný podnět bude vykonávat pouze jeden robot, respektive ten, který překážku identifikoval jako první. Ostatní budou čekat na reakční trasu, vytvořenou robotem, který vykonával reakci.

Z důvodu ověření funkčnosti algoritmizace bude za využití komerčně dostupných logických a mechanických periferií zkonstruována robotická skupina, na které se bude korektnost daného řešení testovat.

Pro možnost parametrizace robotických jednotek a definici trasy bude naprogramována počítačová aplikace. Tato aplikace bude graficky reprezentovat stav vstupů a výstupů skupiny a jedinců.

1. Metodika práce, dostupná řešení

K dosažení cílů práce bylo využito převážně principiálně jednoduchých metodik, jakými jsou pozorování a modelování.

Výchozím bodem řešené problematiky je rozbor dostupných řešení v kombinaci s analýzou obecně známých pohybových a reakčních principů jedinců a skupin, vycházejících z reálného života.

1.1 Dostupná řešení

V současné době není řešení problematiky řízení skupiny robotů a jejich reakce na neočekávané podněty jako celku definováno. Lze ale vycházet z dílčích řešení tvořící robotický celek. Struktura této kapitoly vychází z literatury [1]. Robotický celek lze rozdělit na:

- Řídící subsystém
- Pohonný subsystém
- Navigační subsystém
- Komunikační subsystém
- Senzorický subsystém
- Reakční algoritmy

1.1.1 Řídící subsystém

Implementace řídícího subsystému je základním předpokladem funkční robotické jednotky/skupiny. Nejčastěji použitou platformou bylo v analyzovaných projektech Arduino. Pokud byla v projektu implementována složitější logika, tzn. např. lokální nebo webový server nebo řízení o vyšším nároku na výpočetní výkon, používalo se Arduino v kombinaci s Rapsberry Pi. Jednalo-li se o řešení, které bylo vysoce specifické, často se používaly AVR, ARM nebo PIC procesory s proprietárními logickými obvody.

1.1.2 Pohonný subsystém

Pohonný subsystém je další klíčovou periferií pohyblivé skupiny robotů, je mu tedy třeba při analýze, návrhu a implementaci přikládat dostatečnou důležitost.

V současných realizacích robotizovaných jednotek nebo skupin se nejčastěji používají čtyřkolové nebo dvoukolové podvozky, kdy každé kolo pohání samostatný motor. Výhodné to je zejména z pohledu koordinace a pozicování robota, kdy se kola můžou pohybovat kontra. Nevýhoda spočívá zejména v nutnosti regulace motorů a celkově větších nároků na jejich údržbu. Speciální případy mohou mít jednu hnací hřídel na více kol, případně diferenciál.

Podvozek a motorová část jsou od řídící jednotky zpravidla fyzicky odděleny a moduly citlivé na interference jsou odizolovány zvlášť.

Výkonným prostředkem pohybového subsystému jsou v robotickém odvětví většinou elektrické motory - elektromotory, které pracují na principu elektromagnetické indukce a jsou složeny ze dvou hlavních částí - statoru a rotoru. Tyto části slouží k přeměně elektrické energie na mechanickou práci. Elektromotory můžeme dále dělit na stejnosměrné a střídavé, podle směru toku elektrického proudu.

Regulace

Nedílnou součástí pohybového subsystému je regulace, protože je zpravidla nutné motorovou základnu robota pro přímočarý pohyb regulovat. I typově stejné motory mají z hlediska fyzického sestavení drobné nuance a z toho důvodu se při stejných proudových hodnotách chovají jinak.

Mezi základní druhy regulace patří [7]:

- Proporcionální regulace
- Integrační regulace
- Derivační regualce
- PID regulace

Tyto druhy regulací se mezi sebou v praxi kombinují, pro sofistikovanou, rychlou a stabilní regulaci se používá PID regulace, která je kombinací proporcionálního integračního a derivačního regulátoru. Používá se jak v robotice, tak v průmyslu. Výhodou tohoto regulátoru je, při nalezení správných hodnot P konstanty, I konstanty a D konstanty, což není triviální, zejména rychlost. Není-li při regulaci kladen důraz na rychlost, respektive nedochází-li k rychlím změnám regulační odchylky, používá se PI regulace.

1.1.3 Navigační subsystém

Pro schopnost robota orientovat se v prostoru, je nutné do jednotky implementovat navigační subsystém, v opačném případně se používá manuální řízení.

Nejčastěji se v praxi používá kombinace 4 základních přístupů [2]:

- GPS navigace
- Relativní měření polohy
- Navigace založená na význačných bodech
- Vizuální navigace

Pokud není robot omezen na použití uvnitř budov nebo signálově stíněných míst, používá se ve většině případů robotických projektů navigace GPS a to zejména pro svoji HW modularitu, dostupnost, mezi-platformní kompatibilitu a z toho vyplývající SW podporu.

Algoritmizace, reakce na neočekávané podněty v trase robota, potom není tak složitá, jako v případě ostatních navigačních subsystémů, protože je možné na základě aktuálního směru a souřadnic dílčího cíle trasy vypočítat z každého bodu pohybu robota (v reakci) trajektorii, pro návrat do původního směru trasy.

V případě že je robot omezen na použití bez možnosti příjmu signálu GPS, používá se kombinace výše zmíněných přístupů. Nejpoužívanějším řešení je pro svoji jednoduchost odometrie v kombinaci s vizuální navigací, případně inerciální navigace. Jako korekční prvek směru se často používá magnetometr.

Odometrie je typem relativní lokalizace a je založena na odhadu změny pozice a orientace robota. Zpracovává data o otáčení jeho kol naměřených pomocí rotačních enkodérů. Nevýhodou je kumulativní chyba, vznikající při špatné parametrizaci robota, nedokonalosti adheze kol nebo neočekávaného vyosení robota. Vizuální navigace potom zprostředkovává informace o okolí.

V některých případech se jako nadřazený navigační prvek používá další robot s rozšířenou oblastí působnosti, tedy například kvadrokoptéra nebo roboti přímo uzpůsobeni navigační činnosti.

1.1.4 Komunikační subsystém

Pro možnost komunikace s okolím a kooperaci ve skupině je nutné robotickou jednotku nebo skupinu opatřit patřičným komunikačním zařízením. Komunikace je nejčastěji realizována radiovým spojením, tedy například pomocí Wifi, Bluetooth, GSM, zigBee. Ojediněle se používá neexplicitní navigace. Tou může být modifikace společného prostředí, například umísťování významových těles do prostoru pohybu skupiny, umísťování značek atd., vychází z literatury [8]. Komunikací můžeme ve skupině zajistit kooperaci mezi jednotlivými členy. Rozlišujeme dva základní komunikační přístupy:

- Centralizovaný přístup
- Distribuovaný přístup

Centralizovaný přístup se používá zejména v modelových skupinách, protože je jednodušší na implementaci a nadřazeným prvkem se kontroluje stav jednotlivých členů skupiny a parametrizují se ať už společné, nebo dílčí vlastnosti.

Decentralizovaný přístup se používá ve specializovaných aplikacích. Je implementačně složitější ale nehrozí riziko výpadku nebo ztráty komunikace s nadřazeným prvkem, což vede, v centralizovaném přístupu bez redundantní řídící jednotky, ke kolapsu systému.

1.1.5 Senzorický subsystém

Aby robot dokázal identifikovat překážku, která se stala nečekanou součástí trajektorie jeho trasy, potřebuje získávat informace o svém okolí. K tomu slouží senzory, které lze rozdělit do dvou základních skupin, vychází z literatury [1]:

- interní
- externí

Interní senzory slouží pro diagnostiku periferií robota. Externí potom zprostředkovávají robotu informace o jeho okolí. Z hlediska rekognoskace předmětu je tedy pro robota důležitá senzorika externí. Externí senzoriku můžeme dělit na:

- taktilní
- reflexní

Mezi senzory které slouží k získávání údajů o bezprostředním okolí patří taktilní čidla, která jsou většinou realizovány kontaktním spínačem. Sepnutím nebo rozepnutím kontaktu dojde ke změně logické úrovně která je robotem dále zpracována. Výhodou taktilních senzorů je téměř absolutní korektnost předané informace, bez jakéhokoliv zatížení chybou měření (kontakt je buď seplý nebo rozeplý, nebere se v potaz opotřebení materiálu, případně vnější vlivy), nevýhodou je nutnost bezprostředního kontaktu s okolím. Taktilní čidla se tedy většinou používají jako koncové bezpečnostní prvky, selže-li měření senzorů reflexních.

Reflexní čidla zprostředkovávají informace o okolí bez nutnosti fyzického kontaktu s měřeným tělesem a jsou zpravidla využívány k identifikaci překážky v pohybu. Výhodou je možnost zareagovat na přítomnost překážky před kolizní vzdáleností. Prováděný reakční úkon tedy může být oproti taktilnímu snímání spojitější, např. zpomalení, vybočení, atd. Pracují na principu odrazu specifického vlnění.

V praxi jsou k identifikaci překážky pro svoji dostupnost a dostatečnou spolehlivost často používána čidla ultrazvuková. Ta dokáží v ideálních podmínkách měřit do vzdálenosti 4 metrů. Jeli vyžadováno přesnější, případně prostorově obsáhlejší měření, používají se čidla optoelektronická, která už ovšem nejsou tak dostupná, zejména finančně, jako akustické senzory a jejich implementace je složitější. Nároky na výkon se zvyšují s úrovní sofistikovanosti optoelektronických senzorů. Měření z těchto senzorů je ale zpravidla zatíženo daleko menší chybou a je rychlejší, někdy až mnohonásobně.

1.1.6 Reakční algoritmy

Následující rozbor dostupných řešení předpokládá, že se robotická skupina nebo jedinec pohybují autonomně.

Dostupná jsou pouze řešení, ve kterých nemá robot definovanou trasu nebo má trasu definovanou v souřadném navigačním systému.

Jestliže není možné v logice robota definovat trasu pohybu, bývá často použito řešení, ve kterém robot změní směr pohybu na opačný směr vůči tomu, ve kterém

se překážka nachází. Jeli překážka přímo před robotem, vybere se levý, nebo pravý směr. Tímto způsobem prochází prostor a jeli hardwarově dobře navržený a čidla jsou schopna identifikovat všechny překážky v prostoru, nenarazí a překážkám se tímto triviálním způsobem vyhne.

Jestliže se robot pohybuje prostorem po definované trase a je navigován prostřednictvím souřadného systému, používá se řešení ve kterém je v reakci zvolen opačný směr, než ten, kde je překážka. Jeli překážka přímo před robotem, vybere se levý nebo pravý směr, do kterého se robot natočí a pokračuje v pohybu po libovolnou vzdálenost. Šikmo vůči původnímu směru se vrací k dílčímu cíli trasy. Narazí-li v reakci na překážku, opakuje se celý proces znovu.

1.1.7 Shrnutí

Analyzované projekty byly volně dostupné na internetu, vystavené převážně na diskuzních fórech, webových stránkách univerzit, jako bakalářské či diplomové práce studentů nebo se jednalo o komerční projekty. V publikacích jsou pak k nalezení principy fungování dílčích subsystémů robota a obecně známé algoritmy, které se v každé konkrétní implementaci liší. Za vynikající publikace lze v dané problematice považovat Mobilní roboty od pana prof. Dr. Ing. Petra Nováka [1] a Where am I od J. Borenstein [2].

Majoritním rozdílem mezi touto prací a dostupnými projekty bude použití nestandardního polohovacího subsystému. Nejčastěji je v podobných projektech implementován navigační systém GPS a tomu odpovídá algoritmizace reakcí na neočekávané podněty v trase robotů. Povaha práce tedy vyžaduje zcela nový přístup jak k navigační, tak k reakční části autonomního robotického systému.

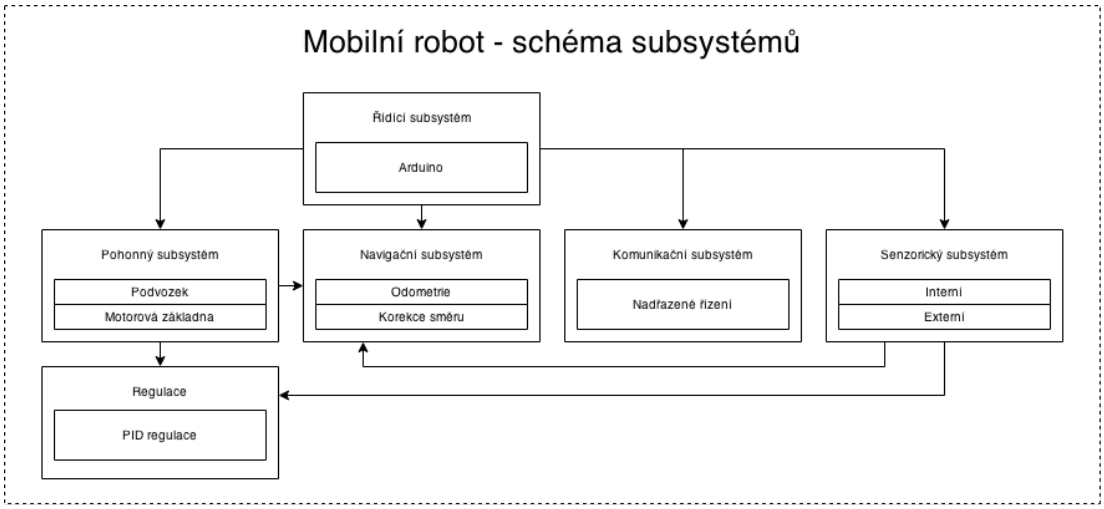
2. Možné způsoby využití robotické skupiny

Vzhledem k neustále se rozvíjejícímu robotickému poli a všeobecnosti výsledného řešení je prostor k využití finální verze práce veliký. V reálném světě se může jednat o využití v následující aplikacích:

- Skupina žacích strojů, jenž je nakonfigurována pro optimální pohyb po žací ploše, bez nutnosti zásahu lidského faktoru, při zachování, případně vylepšení vlastností, kterých bylo dosaženo s lidskou posádkou, či celkové zkvalitnění a urychlení zemědělských služeb.
- Robotická jednotka či skupina nasazená v oblastech prozkoumávání země, oceánu a vesmíru, kde by lidskou posádku vzhledem k její fyziologii nebylo možné použít bez omezení, které dané robotické jednotky, či skupiny, eliminují.
- Skupina využitá pro rekognoskování terénu při rizikových operacích, pátrání po určitém objektu či eliminaci škod po enviromentálních katastrofách. Vzhledem k předpokladu vykonání reakce pouze jedním členem skupiny a následné distribuci ostatním členům, se může průzkum, případně záchranné akce, či rekognoskace prostoru, v době řešení paralelizovat.
- Robotická skupina využitá v logistice, ať už globální, tedy sdílení komunikací s dopravními prostředky – mezinárodní doprava, mezipodniková doprava, osobní doprava nebo lokální, průmyslová logistika a zásobování výrobních linek, pásů, skladové hospodářství atd.

3. Teoretická východiska práce

Na základě předchozí kapitoly - dostupných informací týkajících se subsystémů autonomního robota a jeho algoritmů byla stanovena následující teoretická východiska.



Obrázek 3.1: Mobilní robot - schéma subsystémů

3.1 Řízení robota, platforma

Základním logickým prvkem robota je procesor, mikrokontroler nebo jedno či více čipový mikropočítač. V současné době jsou k dispozici logické jednotky, jednočipové mikropočítače které integrují všechny standardně potřebné vstupně výstupní periferie.

Populární je zejména Arduino, Raspberry Pi, případně Beaglebone. Ačkoli je konstrukce, periferní vybavení i úroveň na které uživatel se zařízeními pracuje rozdílná, dají se použít ke stejnému řešení.

Pro účely této práce je vzhledem k optimálnímu poměru cena/výkon/podpora vyhovující model z rodiny Arduino, Arduino UNO s 16MHz procesorem AtMega328 o 32KB paměti. Uno má 14 digitálních vstupů/výstupů (6 použitelných jako PWM výstup) a 6 analogových vstupů.

3.1.1 Arduino

Arduino je open-source platforma založená na mikrokontrolérech AtMega. Využívá se zejména pro svoji jednoduchost a bezprostřednost. Je lehce rozšiřitelná pomocí tzv. shieldů, které zajišťují další funkcionalitu, například správu motorů, ethernetovou komunikaci, 802.11 komunikaci atd.

K programování desek Arduino se používá programovací jazyk Wiring, který je ovšem velice podobný, ne-li stejný, jako programovací jazyk C++, rozdíl je pouze

v integraci specifických knihoven oproti standardu C++. Projekt pro Arduino vždy obsahuje dvě základní metody a to Setup() a Loop(). Medota Setup se spouští jednou po setu/resetu procesoru, metoda Loop se volá cyklicky.

Standardní IDE Arduina je jednoduché a přehledné, je vhodné pro jednoduché aplikace a začínající programátory. Pro složitější aplikace je možné integrovat do Visual Studia nebo Atmel Studia plugin, obsahující vývojové nástroje Arduina - Visual Micro. Obrovskou výhodou tohoto pluginu je podpora funkce Intellisense, která výrazně zjednoduší práci s rozsáhlým kódem.

Kompilátor umožní překlad z jazyku Wiring, respektive C++ a C.

3.2 Pohonný subsystém

Pohyb je definován jako vzájemná změna stavů hmotných objektů, je základním projevem existence hmoty. Aby bylo možné uvést těleso do pohybu je třeba nejprve vytvořit a následně převést na těleso dostatečný mechanický výkon - to zajišťuje pohonný subsystém, který je v oblasti robotiky nejčastěji reprezentován rotačními elektromotory.

Za výkonný prostředek pohonného subsystému robotů byly po porovnání v daném řešení použitelných motorů, viz níže, zejména kvůli dostupnosti, jednoduchosti užití a druhu napájení, vybrány stejnosměrné elektromotory typu GM37-3530 s převodovkou o převodovém poměru 1/90. Elektromotory pracují na principu elektromagnetické indukce, jsou složeny ze dvou hlavních částí - statoru a rotoru a slouží k přeměně elektrické energie na mechanickou práci [6].

3.2.1 Stejnosměrné motory

Stejnosměrný motor s permanentním magnetem je často používaným typem motoru v oblasti pohonů mobilních robotů. Mezi jeho hlavní výhody patří hmotnost/výkon, relativně snadné řízení otáček, dostupnost a cena. Mezi nevýhody patří složitější a tedy dražší rychlostní a polohové řízení v porovnání s například korovým motorem a fakt, že je motor zdrojem nezanedbatelného elektromagnetického rušení. Stejnosměrný motor obvykle pracuje v relativně vysokých otáčkách a nízkém momentu, což není pro potřebu pohonů mobilních robotů výhodné. Řešením je použití převodovky.

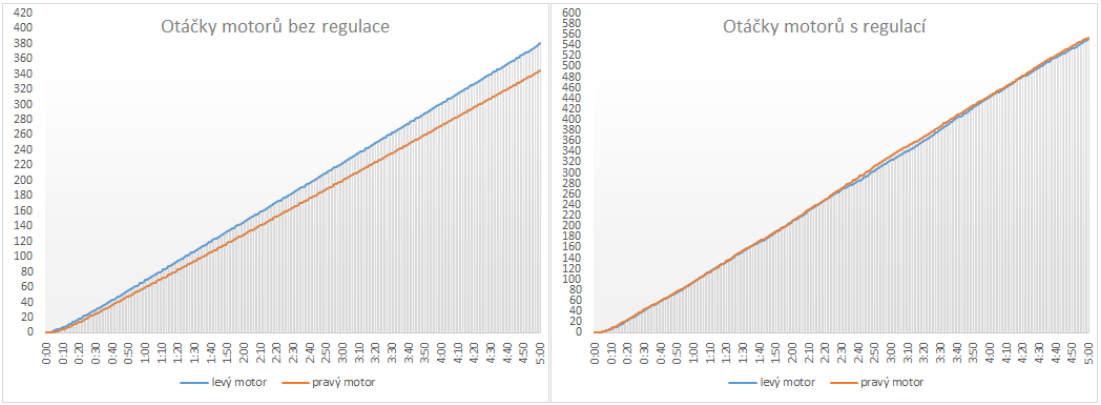
3.2.2 Střídavé motory

Střídavé motory jsou, jak již z názvu vyplývá, určeny do obvodů se střídavým proudem. Pro zajištění kontinuálního běhu motoru je nutné motor napájet třemi fázemi vzájemně pootočenými o 120, zvláštním případem je jednofázový asynchronní stroj, který má dvě cívky vzájemně pootočené o 180. Výhodou těchto motorů je relativně přesné řízení. Zpravidla se v jejich konstrukci totiž používá větší množství magnetických trojic, mezi kterými je možné plynule přecházet a motor lze i magneticky zabrzdit - nevýhodou tohoto typu brzdění je, že motorem v brzděném stavu prochází poměrně vysoký proud, na který je třeba dimenzovat řídící elektroniku.

Pro možnost regualce motorů a využití odometrických principů jsou do pohonného subsystému robotů implementovány dva optické enkodéry - TCST1103 s bipolárními kódovými kotouči, umístěnými na hřídele motorů.

3.3 Regulace

Stejné motory vykonávají při stejném napěťovém zatížení nepatrně rozdílnou mechanickou práci a to se projevuje v rovině trasy robota, respektive tedy zakřivení. Tuto nuanci je nutné regulovat. V praxi, jmenovitě například v průmyslu, se pro zregulování rozdílu dvou výstupních veličin používá tzv. zpětnovazebná regulace PID.



Obrázek 3.2: Porovnání pulzů robota bez a s regulací

Kde je na ose y čas v sekundách, na ose x jsou pulzy z enkodérů (součet pulzů z levého i pravého motoru).

3.3.1 PID regulátor

PID regulátor je paralelním spojením proporcionálního, integračního a derivačního regulátoru. Patří mezi spojité regulátory. Význam jednotlivých složek v použití PID regulace [7]:

- P - tvoří základní zpětnovazební složku
- I - zajišťuje nenulovou akční veličinu pro nulovou regulační odchylku
- D - omezuje překmity

PID regulace se hodí pro regulování rychlých změn regulační odchylky, zpravidla se používá v průmyslu, pro modelářské účely většinou postačí PI regulace.

Výhodami této regulační metody jsou vysoká jakost a rychlost regulace. Nevýhodami jsou oproti jednodušší regulaci, např. proporcionální, vyšší nároky na výpočetní výkon a složité manuální nastavování regulačních parametrů.

Výpočet akční veličiny u(t):

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{3.1}$$

Kde:
K_p : hodnota proporcionální složky
K_i : hodnota integrační složky
K_d : hodnota derivační složky
e : Chyba = požadovaný stav výstupu - současný stav výstupu
t : aktuální čas
τ : proměnná integrační složky, nabývá hodnot z t_0 do t

3.4 Navigační subsystém

Variace navigačních subsystémů, které je možné v robotickém odvětví použít jsou diferencovány použitými senzory při sběru dat, jejich zpracování a následné kombinaci. Zpravidla se v praxi navigační subsystémy kombinují a využívají dohromady. Mezi základní navigační přístupy patří navigace na základě význačných bodů, vizuální navigace, navigace na základě mapy, navigace pomocí GPS souřadnic a relativní měření polohy [2].

Jako polohovací subsystém robota, robotické skupiny byla zvolena metoda Realativní navigace - Odometrie v kombinaci s magnetometrem pro korekci směru.

Odometrie byla zvolena zejména pro nezávislost na explicitních polohovacích prvcích, na rozdíl od GPS navigace, která je závislá na komunikaci se satelity, z toho vyplývá nezávislost na prostředí, ve kterém se bude skupina pohybovat. Relativně jednoduše se také implementuje. Prostředky k využití odometrie jsou inkrementální optické enkodéry, které budou dále využity k regulaci motorů. I přes zatížení odometrického měření nezanedbatelnou chybou v průběhu navigace stále převažují pozitiva výběru, ty se stupňují za předpokladu, že se bude modelová skupina pohybovat po ploše, na které nehrozí prokluz kol a konstrukce robota bude taková, že nebude docházet k nečekanému vyosení podvozku.

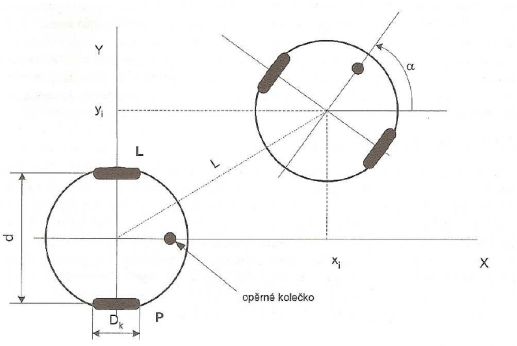
Relativní měření polohy je výpočetně nenáročný a za použití vhodného senzorického subsystému také implementačně jednoduchý způsob navigace. Vychází z pozice, která byla v čase $t-1$ známá a naměřených dat o pohybu robota - rychlost pohybu * čas, počet otáček hřídele motoru, kola, atd. Na základě výše zmíněných informací a jejich vhodné kombinaci lze určit přibližnou polohu v čase t. Aplikace relativního měření polohy se v reálném nasazení jako samostatný navigační subsystém používá zřídka, zejména kvůli velkému zatížení kumulativní chybou a žádné zpětné vazbě. Relativní měření polohy by bylo za ideálních podmínek velice přesné, neboť je každý krok robota matematicky popsatelný. V reálném světe je však toto měření, jak již bylo zmíněno výše, zatíženo napříč celým procesem navigace chybou.

3.4.1 Odometrie

Odometrie je typem relativní lokalizace, je založená na odhadu změny pozice a orientace kolového robota a zpracovává data o otáčení jeho kol naměřených pomocí rotačních enkodérů. Aby bylo možné odhadnout změnu pozice robota na základě ujeté vzdálenosti naměřené jednotlivými enkodéry, je nutné znát kinematický a geometrický model robota. Popis problematiky vychází z literatury [1][p. 169]. Většinu mobilních robotů je možné zařadit do jedné z následujících skupin, dle typu řízení:

- Ackermanovo řízení - robot s tímto typem řízení se nedokáže otáčet na místě. Otočení dosahuje natočením kola či nápravy a následným pohybem vpřed či vzad. Tento typ řízení využívá většina současných automobilů.
- Diferenciální řízení - do této kategorie spadají roboti jenž jsou schopni se otáčet kolem své osy, ale pohybují se pouze vpřed či vzad. Změna orientace závisí na rozdílu rychlosti levého a pravého kola/pásu.
- Všesměrové řízení - roboti řízení tímto typem se dokáží pohybovat všemi směry bez ohledu na aktuální orientaci. Dokážou se otáčet na místě.

Jako druh řízení robotů bylo vybráno řízení diferenciální. Výhodou tohoto řešení je zejména mobilita a dostupnost podvozků v dané kategorii. Diferenciálně řízené podvozky je nutné, ve chvíli kdy nejsou opatřeny gyroskopem a regulační jednotkou, stabilizovat třetím podpůrným kolem nebo kuličkou, viz obrázek č. 3. 2.



Obrázek 3.3: Dvoukolový podvozek s diferenčně řízenými koly a jedním opěrným kolem, převzato z literatury [1][172].

V případě, že má robot pohon s převodovkou s převodovým poměrem n, průměrem hnacího kola D_k a enkodérem s C_0 pulzy na otáčku. Obě hnaná kola od sebe vzdálena d. Odvodíme vztahy, které budou vyjadřovat nové souřadnice robota x_i a y_i a úhel natočení α způsobem popsaným níže.

Protože máme k dispozici údaj o počtu pulzů enkodéru, vyjádříme konverzní součinitel c_k, který odpovídá ujeté vzdálenosti robota na jeden pulz enkodéru:

$$c_k = \frac{\pi D_k}{nC_0} \tag{3.2}$$

Přírustek ujeté vzdálenosti levého (Δu_L) respektive pravého (Δu_P) kola s ohledem na počet pulzů příslušných enkodérů N, potom platí:

$$\Delta u_{L/P,i} = c_k N_{L/P,i} \tag{3.3}$$

Inkrementální posun středu robota vypočítáme:

$$\Delta u_i = \frac{\Delta u_{L,i} + \Delta u_{P,i}}{2} \tag{3.4}$$

Inkrementální natočení středu robota vyjádřené v radiánech:

$$\Delta \alpha_i = \frac{\Delta u_{L,i} - \Delta u_{P,i}}{d} \tag{3.5}$$

Nové relativní natočení robota vypočítáme:

$$\alpha_i = \alpha_{i-1} + \Delta \alpha_i \tag{3.6}$$

Relativní pozici středu vypočítáme: $\mathrm{x}_i = x_{i-1} + \Delta u_i \cos(\alpha_i)(3.6)$

$$y_i = y_{i-1} + \Delta u_i \sin(\alpha_i) \tag{3.7}$$

Výsledný směr pohybu, který je kombinací činnosti výkonných prostředků pohonného subsystému a odometrických výpočtů, není možné bez explicitní korekce udržet. Jako korekční prvek je použit elektronický kompas (magnetometr).

3.5 Komunikační subsystém

Jako komunikační přístup byl vzhledem k jeho výhodám, které byly popsány v kapitole dostupné informace zvolen přístup centrální, kde bude nadřazeným komunikačním prvkem PC aplikace. Prostřednictvím aplikace budou roboti parametrizováni. Naopak budou aplikaci předávat stavové informace.

Za komunikační technologii byla zvolena bezdrátová technologie Bluetooth, pracující v pásmu 2,4GHz. Tato technologie je použita zejména kvůli dostupnosti komunikačních modulů a jednoduchosti komunikace jako takové, která vychází ze sériového drátového rozhraní RS-232.

3.6 Senzorický subsystém

Senzory můžeme z pohledu robota rozdělit na interní a externí. Interní senzory měří aktuální parametry robota, případně parametry jeho subsystémů a externí senzory poskytují informace o okolí, vychází z literatury [2]. Nejdůležitější, klíčové jsou pro robota takové senzory, na základě jejichž změření a následného výpočtu je definován stav robota. Jedná se o navigační a diagnostickou činnost.

3.6.1 Interní senzory

Interní senzory poskytují robotu informace o jeho subsystémech. Pro diagnostické účely je to například stav baterie, monitorování komunikace nebo kontrola teploty robota. Pro účely navigace jsou to informace o akčním subsystému, což je obvykle poloha, rychlost jednotlivých pohonů nebo výstupních členů. Příklady interních senzorů:

- Senzory natočení

  Inkrementální senzory

  Absolutní senzory

- Otáčkoměry

3.6.2 Externí senzory

Externí senzory slouží k získávání informací o okolí robota. Podle způsobu měření můžeme externí senzory rozdělit na pasivní, vyhodnocující pouze přijaté záření a aktivní, vyhodnocující vlastní odražené záření. Příklady externích senzorů jsou:

Reflexní snímání

Reflexní senzory patří do skupiny aktivních senzorů a pracují na principu zachycení odrazu vlnění od objektu. Vlnění je v čase $t-1$ odesláno čidlem, směrem k měřenému objektu. Senzor v čase t odražené vlnění přijímá. Reflexní senzory můžeme dělit na:

Akustické

Akustické reflexní senzory fungují na principu měření generovaného nebo již existujícího ultrazvuku na specifickém spektru frekvencí, který se dodatečně, pro eliminaci rušení, moduluje. Tento zvuk je generován pomocí piezoelektrického měniče.

Optoelektrické

Na rozdíl od akustického měření je optoelektrické měření daleko přesnější a rychlejší. Principem měření je jednoduchá triangualce, kdy máme ohniska zářiče a snímače v přesné vzdálenosti a promítnutím světelného bodu na matici optického snímače jej přepočítáváme na snadno měřitelné hodnoty v obrazových bodech.

3.7 Skupina

Skupina obecně označuje seskupení jedné, dvou či více různých entit v jeden logický nebo funkční celek [3]. Ve skupině je na základě vztahů mezi jednotlivými členy definováno hierarchické zařazení a formace. Na základě chování jedinců je definován stav skupiny.

3.7.1 Stavy skupiny

Stejně jako každý jedinec mění svoje chování během životního cyklu natolik výrazně, že nabývá různých stavů, mění svoje chování - tedy nabývá různých stavů i skupina logicky nebo funkčně stejně zaměřených entit. V praktické pracovní skupině jsou definované tři základní stavy:

- Pohyb - ve stavu pohyb se skupina nachází tehdy, když se všechny robotické jednotky pohybují k cíli trasy, nebo v předem definovaném směru.
- Reakce - ve stavu reakce se skupina nachází tehdy, když je alespoň jeden robot nucen reagovat na neočekávanou překážku na trase.
- Ostatní - ve stavu ostatní se skupina nachází, nenachází-li se ani v jednom z výše definovaných stavů.

3.7.2 Hierarchie

Mluvíme-li o hierarchii, mluvíme o uspořádání členů skupiny tak, že jsou jasně definovány vztahy nadřízených a podřízených jedinců. Hierarchii skupiny je tedy možné reprezentovat stromem, jehož kořen, uzly a listy jsou entity ze skupiny a hloubka těchto entit definuje jejich postavení. Problematika vychází z literatury [4].

Protože je vzhledem k povaze práce nutné aplikovat řešení algoritmu vyhýbání se překážkám alespoň na dvou jedincích, byla jako pracovní skupina zvolena skupina o velikosti právě dvou jedinců. Tito roboti jsou za stavu pohyb na stejné hierarchické úrovni. Narazí-li alespoň jeden robot ze skupiny na překážku, tedy, dojde-li ke změně stavu na reakce, změní se hierarchické postavení robota, který narazil na překážku jako první, na vůdce. Hierarchické postavení druhého robota se změní na následovníkm ne však vůči prvnímu robotu, ale vůči PC aplikaci, která na základě výsledků reakčního procesu prvního robota vytvoří sled pohybových instrukcí, kterými druhý robot překážku objede bez nutnosti opakování reakčního algoritmu. Po úspěšném vyhnutí se překážce se stav robotů opět změní na stejnou hierarchickou úroveň, tedy vůdce - vůdce.

3.7.3 Formace

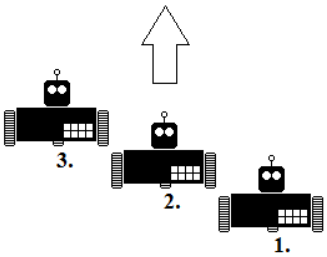
Jako nevhodnější formace skupiny byla pro danou problematiku zvolena formace Rojnice s modifikací, vzchází z literatury [5].

Ve formaci rojnice se všichni roboti, kromě robota s postavením vůdce, řídí omezením, jenž je definováno sousedem/sousedy. Roboti udržují konstantní vzdálenost vůči robotům na bocích. Viz obrázek 3.2.

Modifikací je vůči standardnímu návrhu formace lokační posun vzhledem k prvnímu robotu. Lokační posun je definován šířkou a výškou prvního robota. Druhý robot je o vzdálenost šířky prvního robota + offset posunut na levou nebo pravou stranu a o vzdálenost výšky prvního robota + offset posunut dozadu.

Implementace lokačního posunu má následující výhody:

- Jednoznačné určení robota, který jako první identifikoval překážku - eliminace komunikačního zdržení při oznámení překážky oběma roboty v jeden časový interval
- Eliminace nutnosti zahrnutí do reakčního algoritmu pozici druhého robota
- Snížení časových nároků na oznámení překážky - roboti mají dostatek času na reakci aniž by při řešení reakčního algoritmu došlo ke vzájemnému omezení



Obrázek 3.4: Formace typu rojnice s modifikací

Offset je vzdálenostní hodnota která zajistí dostatečnou časovou rezervu robotů při reakci na překážku. Eliminuje nutnost pozorovat reakční doby jednotlivých HW periferií a jejich následnou propagaci do systému za účelem zachování výhod lokačního posunu v každé změně stavu.

4. HW robota

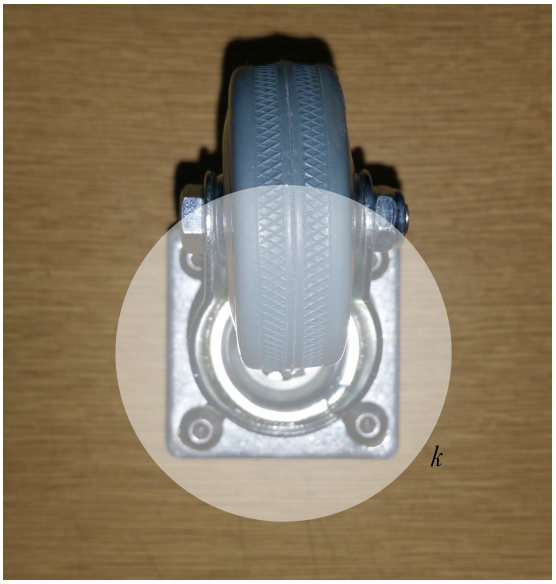
Praktická část této práce je rozdělena do několika kapitol podle chronologie implementace. Ačkoliv se vývoj jednotlivých částí během implementace prolínal, bude zachováno rozdělení do specifických kapitol.

4.1 Řízení

Arduino UNO, které plní řídící funkci robotické jednotky má 16MHz procesor AtMega328 a 32KB paměti. Disponuje 14 digitálními vstupy/výstupy a 6 analogovými vstupy. 6 z digitálních vstupů/výstupů může být použito pro PWM modulaci, které se využívá například při řízení rychlosti motorů - respektive modulaci digitálního signálu na signál elektrický. Na logické desce je posazený motorshield, který fyzicky propojuje jednotlivé piny z UNA na svoji desku, respektive svoje piny. Po třech pinech je obsazeno na řízení jednoho motoru, dohromady je tedy na řízení motorů rezervováno 6 pinů.

4.2 Podvozek

Majoritním stavebním prvkem pohyblivého robota je podvozek. Konstrukce podvozku velkou měrou ovlivňuje výběr a následnou implementaci pohybového systému. Za podvozek robota byl zvolený dvoukolový podvozek s diferenčně řízenými koly a jedním opěrným kolem, jedná se o zakoupené, komerčně dostupné řešení, které je vzhledem k povaze práce naprosto dostačující.



Obrázek 4.1: Pohyb stabilizačního kola po obvodu k

Původní stabilizační jednotka podvozku (značí celek, tzn. stabilizační kolo i s konstrukcí pro připevnění k podvozku) způsobovala při změně úhlu natočení,

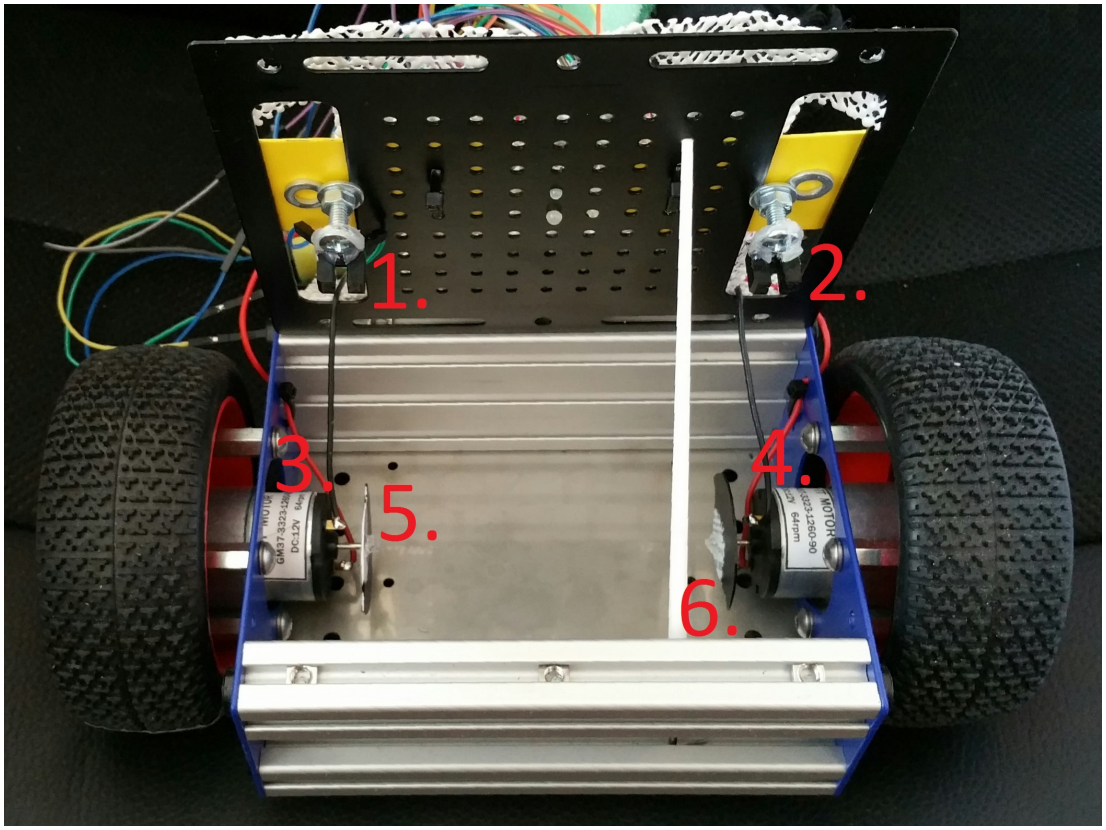
z úhlu $\alpha = 90°$ do úhlu $\gamma = 0°$, což je rozdíl úhlů vyžadovaný pro definované zatočení, přílišné vyosení robota, z důvodu vysunutí stabilizačního kola vůči jeho konstrukci. Z toho vyplývá nutnost pohybu kola po obvodu kružnice kopírující dráhu jeho otočení, viz obrázek 4.1.

Toto kolo bylo nahrazeno stabilizační kuličkou. Kulička je oproti stabilizačnímu kolu náchylnější na vy-osení robota při přechodu mezi strukturou povrchu (spáry kachlové podlahy), nicméně se při zachování heterogennosti povrchu tato chyba eliminuje, při praktických testech výsledného řešení je tedy vyžadován povrch bez výraznějších přechodů.

4.3 Pohonný subsystém

Výkonnou jednotku pohonného subsystém zastupují dva stejnosměrné elektromotory GM37-3530 s převodovkou o převodovém poměru 1/90, obrázek č. 4.2 - položky 3, 4. Jsou napájeny 11.1V baterií. O řízení se stará motorshield Arduina od firmy Deek-Robot. Trojice pinů je použita na řízení jednoho motoru, jeden digitální výstup je použit na povolení/zakázaní práce motoru, druhý digitální výstup na řízení brzdy a třetí PWM výstup je použit na řízení rychlosti motoru.

Motory je nutné regulovat, proto je na hřídel každého z nich umístěno bistavové regulační kolečko, obrázek č. 4.2 - položky 5, 6, jehož stav odečítá optozávora - TCST 1103, obrázek č. 4.2 - položky 1, 2.



Obrázek 4.2: Podvozková část robota s motory a regulačními prvky

Ujetá vzdálenost je odečítána pomocí odometrických principů v řídící logice robotů, je založena na následujících výpočtech. Vzdálenost připadající na jeden pulz enkodéru vypočítáme:

$$c_k = \frac{\pi D_k}{nC_0} \tag{4.1}$$

Kde n značí převodový poměr, průměr hnacího kola je D_k a C_0 značí počet pulzů na jednu otočku hřídele.

Převodovka robotů má převodový poměr 90:1, průměr hnacího kola je 11,68cm a počet pulzů na jednu otočku hřídele je vzhledem k rozlišení bistavového regulačního kolečka jedna. Výsledná vzdálenost připadající na jeden pulz z enkodéru tedy odpovídá 0,4078cm.

V logice robota se sčítají pulzy, které jsou v případě potřeby převedeny na vzdálenost pomocí výše uvedené konstanty. Pro zpřesnění výpočtu se sčítají pulzy z obou enkodérů, které se následně průměrují - poníží se tak případná chyba prokluzu jednoho z kol.

4.4 Komunikační subsystém

Komunikaci robotů s nadřazenou aplikací zajišťuje Bluetooth modul HC-06. Komunikace probíhá sériově a je založena na standartu RS-232, ke komunikaci se tedy dle standartu používají datové vodiče RxD a TxD. Modulační rychlost modulů je nastavena na 9600Bd, stejná rychlost musí být použita na PC, kde běží nadřazená aplikace. Bluetooth modul se dá zakoupit ve třech variantách, v jedné je modul dostupný za roli slave, v druhé za roli master a v třetí se dají role přepínat. Vzhledem k tomu, že se roboti v komunikační hierarchii chovají jako podřízení - slaves, je role použitých modulů slaves. Parametrizace modulů probíhá prostřednictvím tzv. AT příkazů, např.:

AT+NAME : Nastavení jména modulu
AT+VERSION : Dotaz na verzi modulu
AT+UART : Nastavení modulační rychlosti
AT+ROLE: Podporuje-li modul možnost přepínání mezi rolemi, parametrizace rolí (1=master, 0=slave)
AT+ORGL : Obnovení továrního nastavení
AT+PSWD: Nastavení hesla

Parametrizace modulu probíhá prostřednictvím sériové linky, zadáním příkazu AT. Hodnota daného parametru se píše rovnou za příkaz. Příklad nastavení jména: AT+NAMEMojeJmeno.

4.5 Senzorický subsystém

Robot je osazen třemi různými druhy senzorů, optickými inkrementálními senzory, magnetometrem - respektive modulem který v sobě zahrnuje jak magnetometr,

tak gyroskop a akcelerometr (poslední dva submoduly nejsou v řešení použity) a ultrazvukovými čidly.

4.5.1 Magnetometr

Magnetometr spadá do skupiny externích senzorů. Typ implementovaného modulu je GY-80. Tento modul v sobě mimo magnetometr integruje také akcelerometr a gyroskop - tyto periferie nejsou v práci použity. Typové označení magnetometru je - Honeywell MC5883L, na jeho kalibraci je využita explicitní kalibrační knihovna která na základě minutového měření vypočítá offsety jednotlivých os - x, y, z. Tyto offsety se následně manuálně přičítají k výstupnímu měření magnetometru. Kalibraci čidla je třeba provádět při jakékoliv změně hardwarových periferií robota.

Magnetometr se využívá pro korelaci směru. Specifický směr je výsledkem kombinace odometrických výpočtů a softwaru robota. Směr pohybu je definován uživatelem prostřednictvím nadřazené aplikace. Také se na základě měření z magnetometru odečítá úhel otočení robota, robot se pohybuje v síti, viz. kapitola - Algoritmizace robota. V současné době je v softwaru robota implementováno otáčení jak na základě odometrických principů, tak na základě měření magnetometru. Možnost definovaného otočení na základě odometrických principů byla implementována ve chvíli, kdy se vzhledem k nekompatibilní knihovně nedal kompas zkalibrovat a uvést do stabilního stavu. I přes to, že se podařilo na magnetometr Honeywell MC5883L nalézt a upravit knihovnu, byla pro případ, že by měření z magnetometru selhalo nebo se magnetometr porouchal, zachována možnost otáčení na základě odometrie.

Vzhledem k citlivosti magnetometru na elektromagnetické rušení musí být modul vynesen cca 25-30cm nad úroveň usazení HW periferií, zejména potom nad stejnosměrné motory, které jsou největším zdrojem elektromagnetického rušení.

V prvotní návrhu práce byl magnetometr zatížený velkou chybou měření, kterou se nakonec podařilo odstranit jinou kalibrační metodou, než původně použitou. Na trhu je několik typů modulů, většinou bez konkretizovaných kalibračních a řídících knihoven, používají se univerzální. Pro správnou funkci některých modulů je třeba tyto knihovny explicitně upravit.

4.5.2 Optické závory

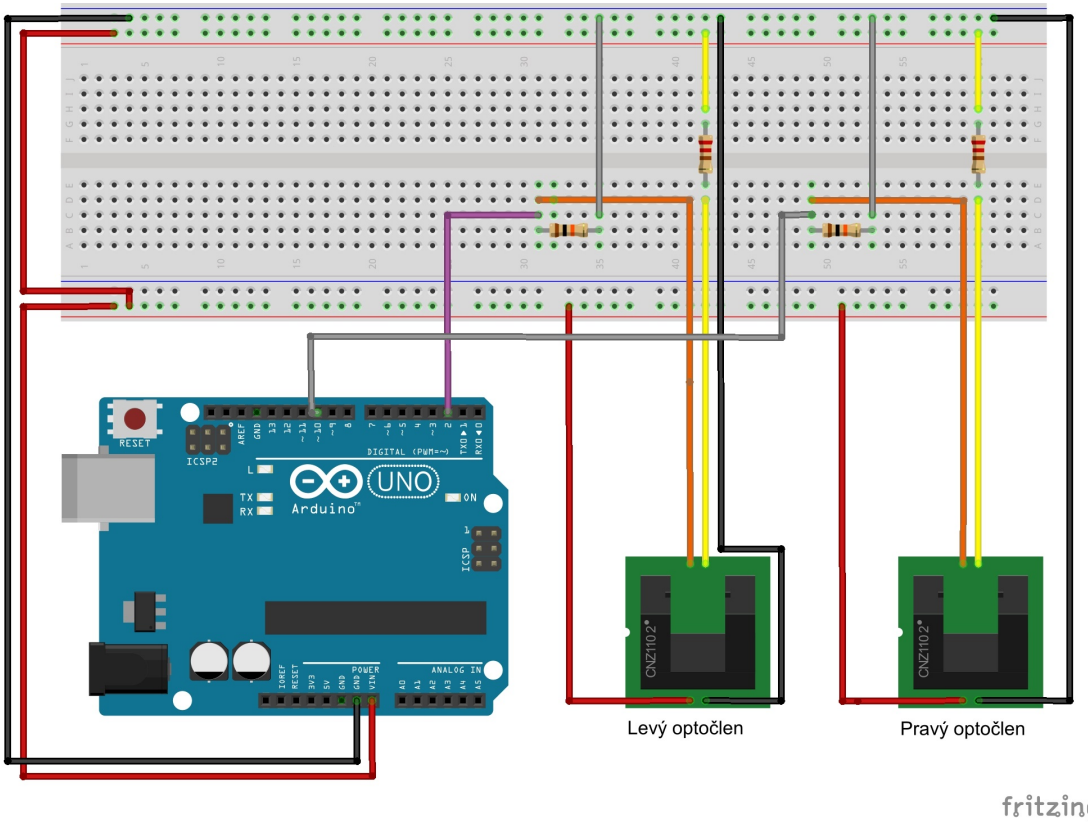
Optické závory jsou použité pro odčítání pulzů z regulačního kolečka, které je umístěno na hřídele motorů. Vybavovací frekvence těchto závor je 5kHz. Optozávora má emitor infračerveného záření a fototranzistor, který na toto záření reaguje. Při přerušení záření ze změní napěťová úroveň výstupu optočlenu, kterou je možné odečíst.

Kolečko je vzhledem k velkému převodu motorů - 1/90 bistavové, tzn. že má jeden stav, přes který infračervené záření optické závory neprojde a druhý, přes který záření projde. Při použití většího rozlišení regulačního kolečka, což by bylo výhodné jak pro regulaci, tak pro odometrické výpočty (regulace by byla více spojitá a odometrické výpočty by byly přesnější), nebylo možné zachytit všechny

změny stavu tohoto kolečka. Testovací rozlišení bylo 16bitů a 4bity. Zachycení nebylo možné kvůli rychlosti volaného přerušení na procesoru - ten nebyl schopen přerušení v daném rozlišení korektně zpracovat.

Přerušení je vzhledem k nemožnosti použití triggeru na vzestupnou hranu nebo logickou jedničku voláno každou změnu stavu, kde je následně tento stav odečten.

Přerušení vzestupnou hranou je možné použít pouze na určitých pinech vývojové desky. Tyto piny jsou dva a jeden z nich je rezervován pro správu motorů a tak ho není možné použít. Druhé přerušení je tedy nutné volat přímo na procesoru ATMega 328, který není kompatibilní s možností přerušení vzestupnou hranou nebo logickou jedničkou či sestupnou hranou. Přerušení tedy probíhá v reakci na změnu stavu napětí na pinu, což zbytečně vytěžuje procesor, ale jiné řešení za použití daného HW není možné použít.



Obrázek 4.3: Schéma zapojení optických závor

4.5.3 Ultrazvukové senzory

Ultrazvukové senzory patří do skupiny externí senzoriky. Zajišťují robotu informace o okolí, respektive o objektech, se kterými by mohlo dojít během pohybu ke kolizi. Pracují na principu zachycení ultrazvuku, který byl senzorem v čase $t-1$ vygenerován. Následně je odraz ultrazvuku - je li odraz dostupný, zachycen a odečten. Pokud není zvuk odražen, znamená to, že není v měřitelné vzdálenosti

čidla žádný objekt, od kterého se může ultrazvuk odrazit. Na základě rozdílu emise ultrazvuku a doby zachycení jeho odrazu je vypočítána vzdálenost.

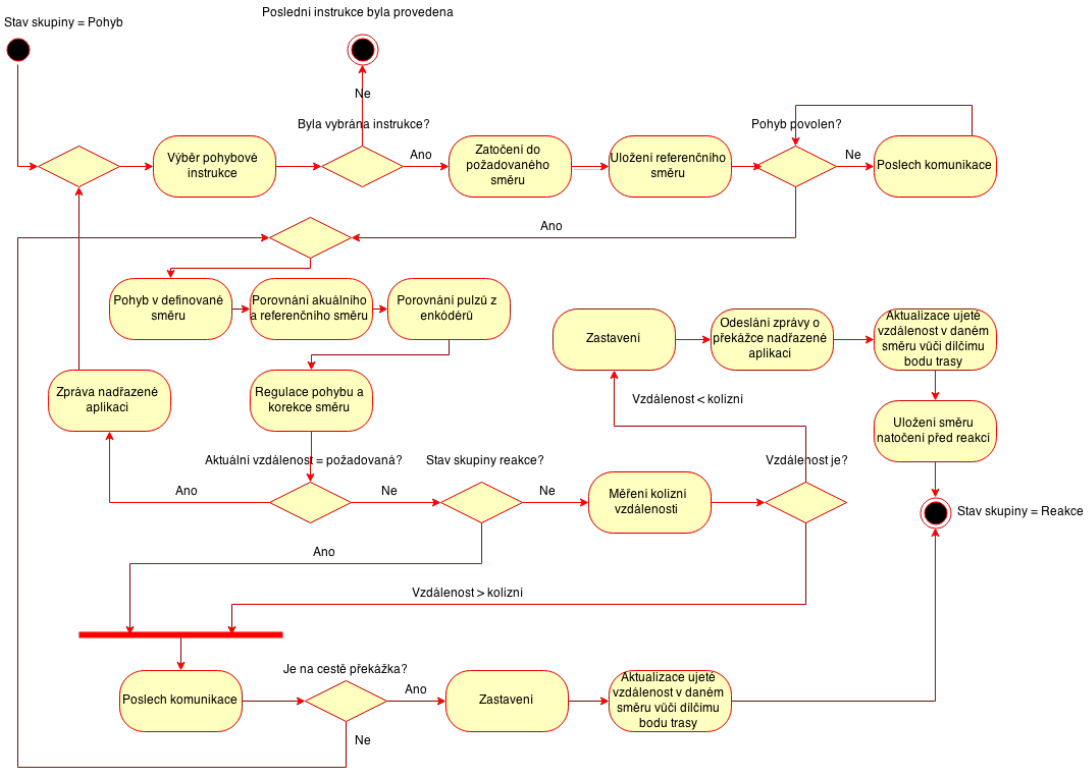
Umístění ultrazvukových čidel na robota je v optimální výšce vzhledem k pravděpodobnosti výskytu překážky - tedy cca 3cm nad zemí. Na robota je umístěna trojice ultrazvukových čidel, a to z důvodu pokrytí co největšího množství kolizních bodů prostoru. Čidlo má vyřazovací úhel 15°, z tohoto důvodu jsou boční čidla umístěna tak, aby byl zajištěn co nejstabilnější odečet, tzn. vůči rovině objektu pod 15°.

5. Algoritmizace

Algoritmizace je jedním z hlavním pilířů této práce. Z algoritmického řešení se odvíjí jak program robota, tak software nadřazené aplikace. Sekce bude rozdělena na Pohybové a Reakční řešení.

5.1 Pohyb

Aby byl robot schopen kontrolovaného pohybu, musí mu být definovány pohybové instrukce, které jsou jednoznačně proveditelné. Po jejich vykonání musí být robot schopen jednoznačně určit svoji polohu v prostoru. K výpočtu uražené vzdálenosti slouží, jak již bylo řečeno v části Teoretická východiska práce, odometrie. K definici směru slouží algoritmické řešení popsané v následující sekci Dopředný pohyb a regulace pohybu.



Obrázek 5.1: Diagram stavu robota - pohyb

Dopředný pohyb a regulace pohybu

Dopředný pohyb robota je realizován na základě točivého momentu motorů. Ujetá vzdálenost je výsledkem konverze hodnot proměnných programu, které se inkrementují při dopředném pohybu robota, respektive při otáčení hřídele motorů. Každý enkodér inkrementuje jednu proměnou. Není-li třeba počítat vzdálenost

nebo vykonává-li robot jiný pohyb, než pohyb dopředný, jsou hodnoty resetovány a inkrementace je pozastavena.

Počet pulzů proměnných je v případě potřeby převeden na cm, v těchto jednotkách je potom vzdálenost uchována v paměti robota. Při jízdě dopředu se cyklicky regulují motory a poslouchá se komunikace pro kontrolu změny stavu skupiny.

Při brzdění robota docházelo bez aplikace explicitního brzdícího mechanizmu k prodlevě v zastavení, což způsobovalo lokační vychýlení robota, které nebylo zahrnuto do odometrických výpočtů. Toto vychýlení bylo eliminováno tzv. kontra brzděním. V globální proměnné se uchovává aktuální stav otáčení kol robota, na základě tohoto stavu jsou kola robota na malý okamžik uvedena do kontra pohybu tohoto otáčení, čímž je jeho pohyb okamžitě zastaven.

V prvotním návrhu regulačního řešení byla implementována statická regulace. Tzn. že se nezávisle na velikosti regulační odchylky zvyšovala, případně snižovala hodnota offsetu, který se následně přičítal, nebo odčítal ke statické rychlosti motorů. Tato regulace byla jednoduchá na implementaci a stabilní v ideálních podmínkách. Nebyla ovšem schopna, vzhledem k žádné trvalé zpětné vazbě, reagovat na rychlé změny regulační odchylky. Proto bylo použití statické regulace zváženo a následně byla implementována PID regulace, která eliminuje nedostatky statické regulace.

Regulace pohybu probíhá na základě odečtu hodnot proměnných, které se inkrementují dle logického stavu implementovaných enkodérů. Tato operace se opakuje každých 100ms a na základě výsledného rozdílu pulzů (regulačí odchylka) se pomocí PID regulace vypočítá akční veličina, která je následně přičtena nebo odečtena od statické rychlosti motorů. Přičítá se na motor A ve chvíli, kdy je motor A pomalejší než motor B, od rychlosti motoru B se v situaci je li pomalejší motor A offset odečítá - ten samý princip je aplikovaný na zpomalení motoru B. Operace se invertují ve chvíli kdy je motor A rychlejší než motor B.

Pulzy se přičítají do globální proměnné programu ve chvíli, kdy je na výstupu optické závory daného motoru logická jednička. Po výpočtu offsetu se tyto hodnoty nulují a inkrementují se znovu. Každý motor má svoji proměnnou.

Jádro PID regulace:

```
get_ticks_since_last(&dlticks, &drticks, &dms);
lticks += dlticks;
rticks += drticks;
ms += dms;

if (ms >= 100)
   {
      diff = lticks - rticks;
      diff = (diff * 100L) / ms;
      pid_sumErrs += diff;
      delta = diff - pid_lastErr;
      int16_t P = Kp*diff + Ki*pid_sumErrs + Kd*delta;
      pid_lastErr = diff;
      int16_t adjust = (P / 2);
      motorLeftOffset -= adjust;
```

```
      motorRightOffset += adjust;

      lticks = 0;
      rticks = 0;
      ms = 0;

      updateMotors();
   }
```

Metoda spravující hodnoty proměnných přerušení a čas přístupu k jádru PID regulace. Předáním hodnot z proměnných přerušení do proměnných regulace eliminuje ztrátu pulzů, která by nastala v případě přímé práce s proměnnými přerušení.
Parametry lft a rht jsou proměnné přerušení, v parametru ms je předáván počet milisekund od posledního volání:

```
void get_ticks_since_last(int16_t *lft, int16_t *rht, uint16_t *ms)
{
   cli();
   *lft = ticksLeft;
   *rht = ticksRight;
   long now = millis();
   *ms = (uint16_t)(now - lastCall);
   lastCall = now;
   ticksLeft = ticksRight = 0;
   sei();
}
```

Explicitní přerušení procesoru, inkrementace proměnných přerušení. Jako parametr je předán vektor přerušení procesoru:

```
ISR(PCINT0_vect)
{
   bool tick = (digitalRead(rOG) == HIGH) ? 1 : 0;
   if (gF90ON)
   {
      gF90TicksR += tick;
   }
   ticksRight += tick;
}
```

Korekce směru

Protože není pohybový ani navigační subsystém schopen samostatně zajistit absolutní dodržení směru pohybu, je nutné implementovat explicitní korekci. Výkonným prostředkem korekce je magnetometr.

Na začátku dopředného pohybu je uložen tzv. referenční směr, který je mediánem 1s měření magnetometru (měření po 200ms tedy 5 měření). Referenční směr je následně porovnán se směrem aktuálním. Je-li tento odečet větší než 0, tzn.

že se robot při dopředném pohybu vychyluje ze směru, je tato chyba přičtena k regulační odchylce a zanesena PID regulace. Pokud je robot v pohybu jiném než dopředném, referenční bod se resetuje.

Pokud je robot v reakci, je referenční bod uložen na začátku řešení reakce, porovnání a následné zanesení chyby do regulace (regulátoru) potom probíhá stejně jako v předchozím řešení. Reference se uloží na začátku reakce z důvodu povahy reakčního algoritmů, respektive předpokladu, že se robot snaží dostat před překážku, tzn. že bude v reakčním pohybu převažovat směr, ve kterém se nacházel před reakcí.

5.1.1 Algoritmizace pohybu

Je nutné algoritmizovat pohyb robota tak, aby byla vždy jednoznačně určena jeho poloha. Řešení, které je v práci aplikováno, bylo nazváno Pohyb v síti. Jedná se o řešení, které má v současné chvíli definovány čtyři směry. Při zachování základních principů by jej ale bylo možné aplikovat na n směrů, pohyb robota by tedy mohl být zcela spojitý.

Pohyb v síti

Robot má definované čtyři stavy natočení (vlevo, vpravo, dopředu, dozadu) a vždy se při autonomním režimu nachází v jednom z nich. Toho lze dosáhnout přesnou definicí otočení robota.

Definovat otočení můžeme za pomocí odometrie, kdy hodnota, o kterou je nutné se otočit (v pulzech enkodéru), vychází ze vztahu:
360° otočení kola = 90 pulzů enkodéru (dopředný pohyb)
180° otočení kola = 45 pulzů enkodéru (dopředný pohyb)
180° otočení kola / 2 = 45 pulzů enkodéru (pohyb kontra - diferenciálně řízený podvozek)

Hodnotou 45 pulzů je zajištěno otočení o 90°.

Další variantou definice otočení je dosažení směru, který je rozdílem měření magnetometru. V rozsahu měření magnetometru, což je 359° je na základně požadovaného úhlu natočení, což je v našem případně 90°, aktuálního směru a směru požadovaného natočení vypočtena hodnota, které musí robot pro definované zatočení v požadovaném směru dosáhnou. Je-li aktuální hodnota výstupu magnetometru rovna vypočítané hodnotě, je jednotka otočena o 90°. Implementována jsou obě řešení a to zejména z důvodu prvotní nestability měření magnetometru. V současné implementaci se dá za stabilnější řešení považovat řešení s využitím magnetometru, a to kvůli zatížení odometrických výpočtů chybou při pohybu na neideálním povrchu, zejména pak při otáčení.

Definice zatočení na základě měření magnetometru:

```
headingNow = readCompass();

int16_t desiredHeading = headingToReach(headingNow, checkingAngle,
    true);

go_right();
```

```
while (headingNow != desiredHeading)
{
   regulate();
   headingNow = readCompass();
}
brake();
```

Definice zatočení na základě odometrie:

```
   go_right();

   while (!(gF90TicksL > angle && gF90TicksR > angle))
   {
      regulate();
   }
   brake();

   gF90ON = false;
```

Příklad:

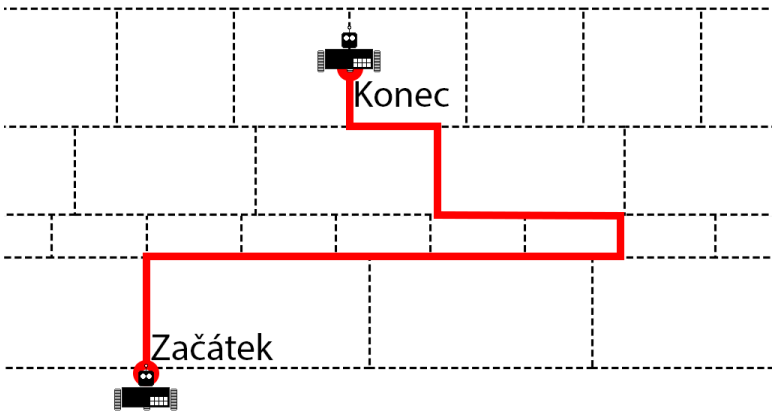
Stav robota je vlevo. Přijde požadavek na otočení doprava, robot se otočí doprava o 90°. Vypočítá se nový stav natočení - ze stavu levá a otočení doprava vznikne na základě obecně známých principů ve čtyř směrném prostoru rozděleného rovnoměrně ve 360° směr dopředu. Toto řešení je aplikovatelné z každého směru natočení do každého směru natočení.

Aby bylo možné robotu definovat trasu, je třeba tento algoritmus aplikovat reverzně.

Robot se v počátečním bodě trasy vychází ze stavu dopředu. Ze záznamu trasy (definuje uživatel) vybere směr ve kterém se musí pro dosáhnutí dílčího cíle trasy pohybovat. Toho dosáhne porovnáním aktuálního pohybového stavu, se stavem zadaným. Následně je v programu definován stav nový, do kterého se robot natočí, příklad:

Stav robota je dopředu. Ze záznamu trasy vybere instrukci, která mu definuje směr pohybu dozadu. Robot na základě programu - obecně známých principů, kdy je ve čtyř směrném prostoru rozděleného do 360° ze stavu dopředu - doleva a doleva = dozadu nebo doprava a doprava = dozadu, vybere sled instrukcí, kterých je třeba dosáhnout pro požadovaný směr, v tomto případě tedy doprava a doprava nebo doleva a doleva. Po vykonání těchto instrukcí se robot nachází ve směru dozadu, který si uloží jako aktuální.

Pohyb robota je tedy popsatelný jako pohyb v nepravidelné obdélníkové síti, kde je jasně definované zatočení, ale velikost stran se na základě odometrických výpočtů mění dynamicky - ujetá vzdálenost, viz obrázek č. 5.2.



Obrázek 5.2: Příklad pohybu robota v síti

5.1.2 Trasa

Trasa je v paměti robota uložena jako n krát iterovaná dvojrozměrná struktura, kde je n v současné době kvůli paměťovým možnostem robota omezeno na 10 průjezdních bodů. Trasu definuje v prvotní fázi uživatel prostřednictvím nadřazené aplikace, robot si ji uloží a následně vybírá jednotlivé instrukce. Stejná struktura je použita pro management reakční trasy.

Jedna instrukce se skládá ze směru pohybu a vzdálenosti, kterou je třeba urazit pro dosažení dílčího cíle. Nejprve se tedy robot natočí do směru trasy - viz sekce Pohyb v síti a následně jede dopředu - viz sekce Dopředný pohyb. Jakmile robot dosáhne dílčího cíle trasy, oznámí daný stav nadřazené aplikaci, označí si danou instrukci jako provedenou a čeká na pokyn k pohybu. Na svolení k pohybu čeká kvůli lokační synchronizaci s ostatními roboty, následně se celý proces opakuje.

Roboti ve skupině mají definovanou stejnou trasu. Pozičně jsou rozlišeni lokačním posunem.

Správa jednoho průjezdního bodu trasy, konrkétně bodu č. 10:

```
if (routeToMove.direction10 != none)
{
   if (routeToMove.direction10 == reached)
   {
      if (moveInReactAllowed)
      {
         Bluetooth.println("TARIR"); //Target all reached in react
         goToPosition(2);
         moveInReactAllowed = false;

         moveAllowed = true;
```

```
      }
      else
      {
         Bluetooth.println("TAR"); //Target all reached, movement
         moveAllowed = false;
      }
   }
   else
   {
      turnToDirectionToMove(routeToMove.direction10);
      if (go_forwardForDistance(routeToMove.distance10))
      {
         if (moveInReactAllowed)
         {
            routeToMove.direction10 = reached;
            Bluetooth.println("T10R"); //Target 10 reached
         }
         else
         {
            routeToMove.direction10 = reached;
            sendNTimes("-TR-"); //Target reached - group synchronization
            moveAllowed = false;
         }
      }
   }
}
else
{
   if (moveInReactAllowed)
   {
      Bluetooth.println("TARIR"); //Target all reached in react
      goToPosition(2);
      moveInReactAllowed = false;

      moveAllowed = true;
   }
   else
   {
      Bluetooth.println("TAR"); //Target all reached, movement
      moveAllowed = false;
   }
}
```

5.2 Reakce

Reakce robota je vzhledem k povaze práce jeho nejdůležitějším stavem. Vychází ze stavu pohyb, kdy jeden z robotů oznámí překážku v trase.

Stav reakce lze rozdělit na základě toho, na co který jedinec reaguje. Prvním reakčním stavem je bezprostřední reakce, ta se týká v jeden moment pouze jednoho robota a to toho, který narazil na neočekávaný podnět v trase. Druhým stavem je reakce skupiny, to je reakční stav který se týká robotů, jenž dostali oznámení o překážce v trajektorii trasy.

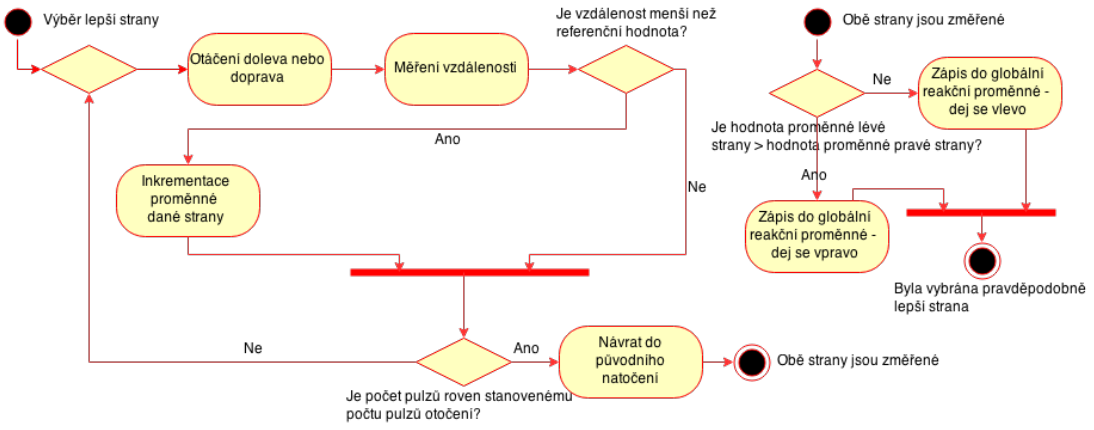
5.2.1 Bezprostřední reakce

Po přechodu robota ze stavu pohyb do stavu bezprostřední reakce, je jeho prvním cílem zjistit, na kterou stranu bude pravděpodobně výhodnější se vydat. Tento proces bude dále v práci používán pod názvem výběr lepší strany.

Výběr lepší strany patří do prvního podstavu bezprostřední reakce a to tzv. prvotní reakce. Druhý podstav se nazývá druhotná reakce.

Prvotní reakce

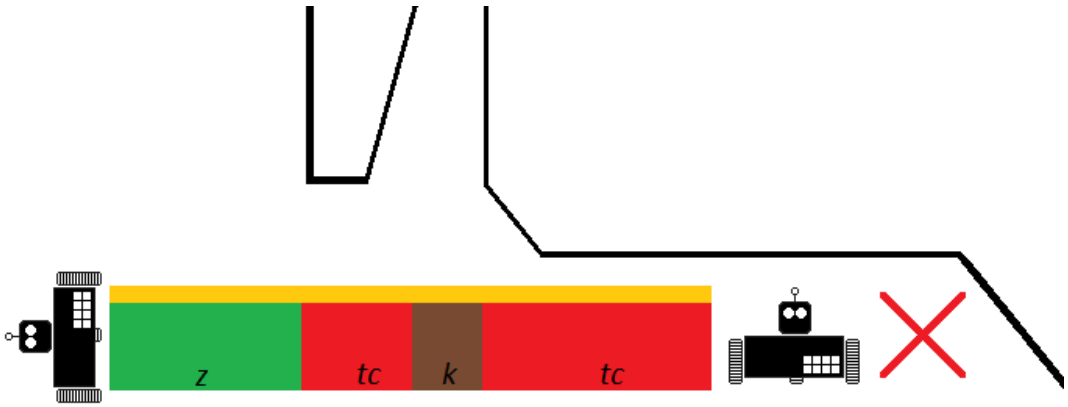
Pravděpodobně výhodnější stranu pro pohyb zjistí robot tak, že si při zatočení doleva o 90°, dále už jen zatočení doleva, inkrementuje proměnou a to pouze ve chvíli, kdy je vzdálenost menší, než vzdálenost referenční. Vzdálenost referenční má v současné době vzhledem k prostoru, kde je skupina testována, hodnotu 125cm. Poté se vrátí do výchozí pozice a opakuje celý postup znovu při natočení doprava o 90°, dále už jen doprava a inkrementuje jinou proměnnou. Následně jsou obě proměnné porovnány a ta strana proměnné, která je nižší, obsahuje méně bodů bližších, než strana druhá. Je tedy pravděpodobně lepší se danou stranou vydat. Následně proběhne uložení výhodnější strany.



Obrázek 5.3: Diagram výběru pravděpodobně výhodnější strany

Po výběru lepší strany se robot natočí do výsledné strany operace. Jede kupředu dokud není vzdálenost, která je měřená ultrazvukovým čidlem umístěným na opačné straně vůči straně výhodnější, větší než tzv. vzdálenost uvolnění, po dobu

větší než ujetí tzv. nutné vzdálenosti, která odpovídá šíři robota + rezerva, to je 40cm. Vzdálenost uvolnění je v současné době nastavena na 40cm.



Obrázek 5.4: Shcéma prvotní reakce

Kde:
Žlutý blok označuje stranu čidla, kterým se měří vzdálenost
tc: vzdálenostní blok kde vzdálenost v boku $<$ vzdálenost uvolnění
k: blok kde vzdálenost v boku $>$ vzdálenost uvolnění a vzdálenost bloku $<$ nutná vzdálenost ujetí
z: blok, po jehož přejetí robot přechází do stavu druhotné reakce, kde vzdálenost v boku $>$ vzdálenost uvolnění a vzdálenost bloku $>$ nutná vzdálenost.

Robot si v průběhu reakčního stavu vytváří reakční trasu, používá se stejná struktura která je použita pro definici trasy uživatelem. Tato trasa bude po ukončení stavu bezprostřední reakce předána zbytku skupiny. Ta nebude muset procházet stavem bezprostřední reakce ale vyřeší objetí překážky stavem reakce skupiny. Tím se ušetří výkon a čas zbytku skupiny, která by mohla v modifikovaném řešení práce vykonávat v době řešení stavu bezprostřední reakce, úplně jinou činnost, např. analýzu okolí, sběr vzorků, atd.

Po nalezení prvního bloku *z* z obrázku 5. 4, robot uloží první hodnotu do globálně deklarované instance struktury trasa. Následně do struktury, v průběhu stavu bezprostřední reakce, ukládá průjezdní body, kterými překážku objíždí. Ukládá se pár: směr a vzdálenost - dále už jen uložení reakčního bodu.

Narazí-li robot v prvotní reakci na překážku, uloží reakční bod trasy a resetuje stav prvotní reakce.

Část reakčního kódu, pro výsledek operace výběr lepší strany - levá:

```
case left:
  while (!canStop)
  {
    regulate();

    moveToAdvance(sonarRight, ticksAfterRFD, pulsesToAM, canStop);

    if (obstacleInWayForReact(gLF) == true)
```

```
    {
        gF90ON = false;
        Bluetooth.println("ObsInReactLOC");
        brake();
        distance = 0.4 * ((gF90TicksL + gF90TicksR) / 2);

        routeReact.storePair(actualDirection, distance);
        return;
    }
}
break;
```

Druhotná reakce

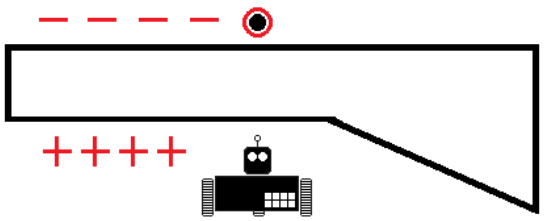
Do druhotné reakce přechází robot ve stavu reakce, z podstavu prvotní reakce, po přejetí bloku *z* z obrázku 5.4. Klíčovou hodnotou tohoto stavu je výsledek výběru lepší strany. Byla-li vybrána jako pravděpodobně lepší strana levá - bude hodnotou primárního zatočení vpravo a naopak. Primární natočení je směr natočení, kterého bude robot v tomto podstavu využívat.

Robot se otočí do primárního natočení a jede dopředu, dokud není vzdálenost na ultrazvukovém čidle strany primárního zatočení větší než vzdálenost uvolnění, a to po dobu větší, než je ujetí nutné vzdálenosti.

Narazí-li robot během druhotné reakce na překážku, uloží reakční bod trasy a přejde do stavu prvotní reakce.

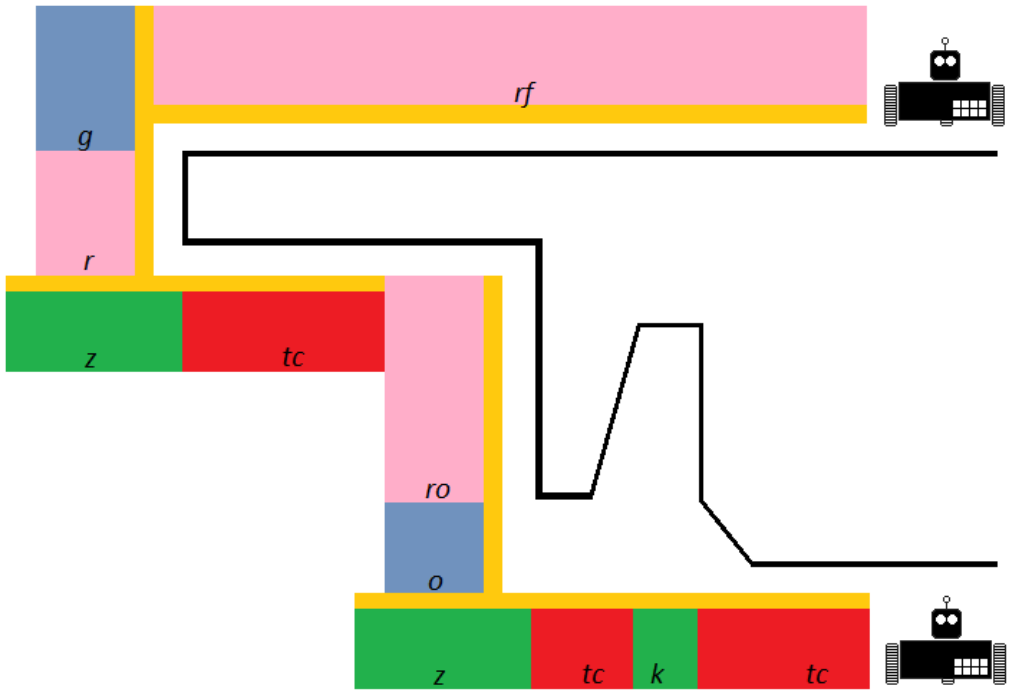
Během dopředného pohybu robot kontroluje odečet vodorovných stran vůči uloženému směru před přechodem do stavu reakce. Například, je-li výstup z výběru lepší strany levá a robot se nacházel před stavem reakce ve směru dopředu, bude odečítat součet vzdálenosti ujeté v levém směru od součtu vzdálenosti ujeté ve směru pravém.

Jeli tento odečet nulový, robot úspěšně objel překážku.



Obrázek 5.5: Shcéma výpočtu ujetých vzdáleností ve vodorovných stranách (ukončení reakce) - druhotná reakce

Tento algoritmus je aplikovaný na všechny výchozí směry v kombinaci se všemi výstupy výběru lepší strany.



Obrázek 5.6: Schéma druhotné reakce

Kde:
Žlutý blok označuje stranu čidla, kterým se měří vzdálenost
tc, k, z: viz obrázek č. 5.4
o: blok kde vzdálenost v boku $>$ vzdálenost uvolnění a vzdálenost bloku $<$ nutná vzdálenost ujetí
ro: blok, ve kterém robot přechází, vzhledem k překážce v cestě do prvotní reakce, vzdálenost v boku $<$ vzdálenost uvolnění
r: blok, kde vzdálenost v boku $<$ vzdálenost uvolnění
g: blok, po jehož přejetí robot zatočí do primárního natočení, zůstává ve stavu druhotné reakce, vzdálenost v boku $>$ vzdálenost uvolnění a vzdálenost bloku $>$ nutná vzdálenost.
rf: blok, ve kterém robot, na základě odečtu vodorovných směrů, deklaroval objetí překážky, změna stavu reakce na ostatní, vzdálenost v boku $<$ vzdálenost uvolnění

Po úspěšném objetí překážky robot přechází ze stavu reakce do stavu ostatní, následně sečte z reakční trasy všechny ujeté vzdálenosti ve směru, ve kterém se nacházel před stavem reakce. Tento součet odečte od vzdálenosti směru před stavem reakce ve standardní trase a uloží ho do reakční trasy. Tento odečet je proveden z důvodu případné akumulace vzdálenosti ve směru daném před stavem reakce. Poté předá nadřazené aplikaci vytvořenou trasu a přejede do bezpečné vzdálenosti od místa ukončení reakce, tím zamezí srážce s ostatními roboty, která

by bez tohoto posunu, vzhledem k tomu, že reakce skupiny končí ve stejném bodě prostoru jako bezprostřední reakce, nastala.

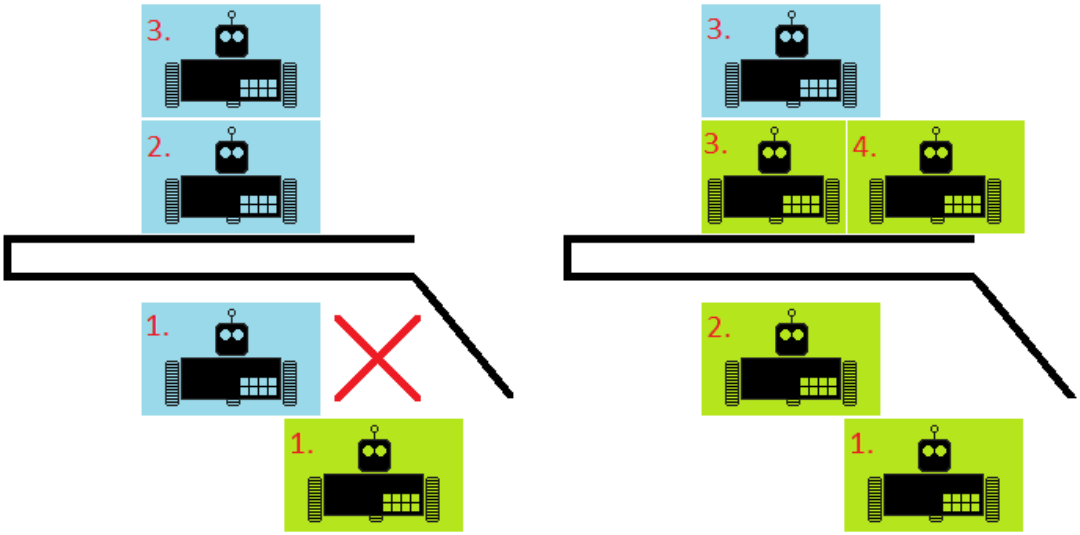
Po lokační synchronizaci s ostatními jedinci robot přejde do stavu pohyb.

5.2.2 Reakce skupiny

Během dopředného pohybu robota je cyklicky kontrolována komunikace s nadřazenou aplikací. Na základě došlé zprávy, konkrétně zprávy 'h' dojde ke změně stavu skupiny na stav reakce skupiny, načež roboti reagují zastavením pohybu - zabrzděním.

Zpráva je přeposlána nadřazenou aplikací v reakci, na oznámení překážky v trase prvním robotem - tento robot se nachází ve stavu bezprostřední reakce, nemění svůj stav na reakce skupiny. Ostatní roboti zastaví všechny probíhající procesy, aktualizují ujetou vzdálenost v daném směru a inicializují proměnné reakce.

Kvůli riziku srážky s prvním robotem čeká skupina po definovaný čas na místě, kde se zastavila. Čas čekání je v současné době nastaven na 15s. Poté přejede další v pořadí skupiny na pozici prvního robota, respektive na poslední pozici, ve které se nacházel před stavem reakce. Tam robot čeká na obdržení reakční trasy. Vzhledem k počtu robotů modelové skupiny není v logice robota implementováno přejíždění při pořadí robota ve skupině > 2.



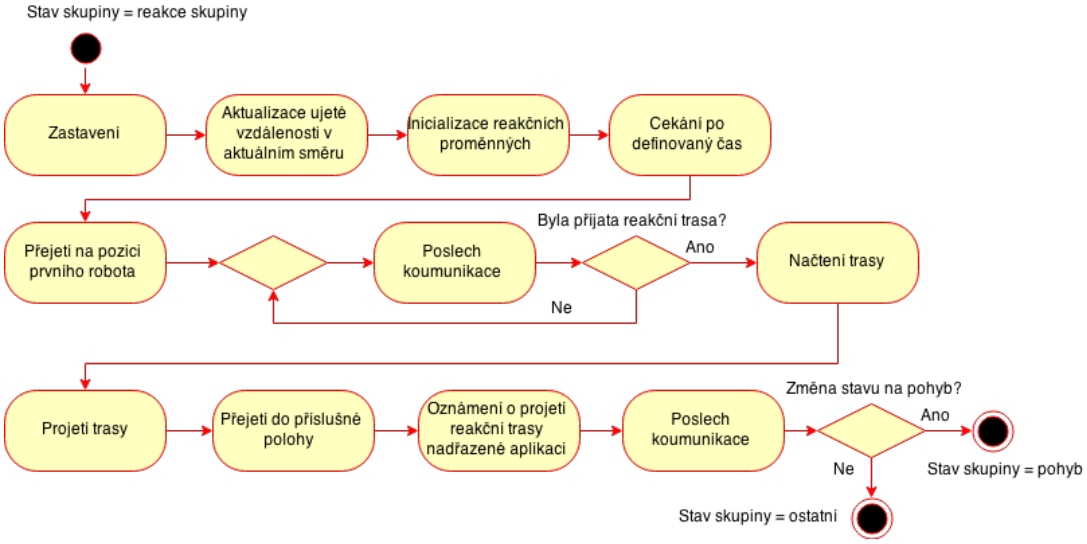
Obrázek 5.7: Schéma pozicování robotů

Pro načtení reakční trasy je využito stejných principů jako při definici trasy uživatelem. Definice trasy se oproti standardní trase liší v poslední dvojici, směru a vzdálenosti. Poslední dvojice reakční trasy definuje odečet vzdálenosti směru v posledním směru před reakcí ve standardní trase. Robot provede odečet a při zpracování trasy nebere poslední instrukci v potaz.

Po načtení trasy robot přejde do stavu pohyb v reakci. Stav pohyb v reakci není od standardního stavu pohyb implementačně oddělený, jedná se ale o podstav stavu reakce. V tomto podstavu není brán ohled na stav ultrazvukových senzorů

identifikujících překážku. Předpokládá se, že první robot našel řešení takové, že nebude třeba během přejetí trasy přejít do stavu reakce - bezprostřední reakce. Algoritmus průchodu trasou je stejný, jako v trase standardní. Po úspěšném průchodu trasou robot zastaví a přejede do pozice takové, aby byl zachován původní lokační posun robotů. Následně předá jedinec informaci o úspěšném projetí reakční trasy nadřazené aplikaci, která, jsou-li všichni roboti lokačně synchronizováni, odešle povolení k pohybu. Nedostane-li robot/skupina okamžitou odpověď o změně stavu na pohyb, přechází do stavu ostatní.

Jakmile skupina obrží povolení k pohybu, je stav skupiny změněn na pohyb, pokračuje se tedy v pokusu o dosažení cíle trasy.



Obrázek 5.8: Diagram stavu robota - reakce - reakce skupiny

6. Nadřazená aplikace

Nadřazená aplikace je napsána v programovacím jazyce C# a využívá grafických knihoven Windows Forms. Je nadřazeným komunikačním prvkem skupiny robotů.

K tomuto řešení bylo přistoupeno zejména kvůli možnosti bezdrátové správy robotů, tj. parametrizaci HW periferií, definici trasy, kalibraci senzorů, atd. Ze strany robota jsou prostřednictvím aplikace zprostředkovávány informace o stavech jeho vstupů a výstupů. Tyto informace by v řešení bez nadřazené aplikace bylo nutné prezentovat na každém jednotlivém robotu, což by bylo náročné na výpočetní výkon a bylo by nutné použít další HW periferie, jako např. reproduktory, displeje, atd. Výkonnostně nezanedbatelný je management komunikace, který by nebylo možné v současné HW konfiguraci robotů implementovat.

Aplikace s roboty komunikuje prostřednictvím sériové linky, zajištěné bezdrátovou technologií Bluetooth - dále už jen BT. Pro vytvoření spojení je třeba adresovat port, nastavit modulační rychlost a rychlost výpisu, respektive rychlost výběru komunikačního bufferu. Defaultní hodnota modulační rychlosti je 9600bd a rychlosti vyčítání linky je 1000ms.

Ihned po navázání komunikace musí uživatel definovat robota, který je připojený k lince, to kvůli rozličnému nastavení periferií. Např. konstant PID regulace, kalibračních hodnot magnetometru, atd.

Předpokladem pro optimální běh aplikace je počítač s BT modulem a jeho funkčním řízením, nutná je také podpora multiklientského přístupu ze strany BT modulu a dostatečný výkon PC na obsloužení této komunikace bez znatelné latence.

Základní funkcí aplikace je parametrizace robotů a trasy.

Kvůli jednotě kódu v sobě roboti uchovávají nastavení všech členů skupiny, v současné době tedy dvě různá nastavení. Vzhledem k tomu, že se jedná o paměťově zanedbatelné položky, není třeba tyto hodnoty posílat robotům explicitně, ale je možné je mít uložené v kódu. Prostřednictvím SW aplikace se tedy vybere nastavení, které se na periferie robota aplikuje.

Po navázání komunikace se robot dotáže na nastavení - robot číslo 1 má nastavení 1 a robot číslo 2 má nastavení 2.

Příklad konfiguračních hodnot pro robota č. 2:

```
// PID constants
Kp = 12.10;
Ki = 1.10;
Kd = 1.0;

// Magnetic field measurement offset
compass_x_offset = -90.34;
compass_y_offset = -3.71;
compass_z_offset = -129.83;

// Magnetic field gain error
compass_x_gainError = 0.88;
compass_y_gainError = 0.39;
```

```
compass_z_gainError = 0.88;

// Motor static speed offset
motorStaticLeftOffset = 10;
motorStaticRightOffset = 0;
```

K parametrizaci trasy jsou využívány komponenty viz obrázek č. 6.2 (1). Uživatel vybere směr a vzdálenost, kterou má robot ujet. Vzdálenost je udána v centimetrech. Není-li vzdálenost definována, je nastavena na implicitní hodnotu 999. Trasa je před odesláním přeformátována na řetězec, který je robot schopný přijmout. Formát tohoto řetězce je - směr:vzdálenost&směr:vzdálenost, atd., dle poču průjezdních bodů. První dvojice trasy neudává směr a vzdálenost ale definuje pozici robota - první robot 1:0, druhý 2:0, robot v manuálním režimu 0:0.

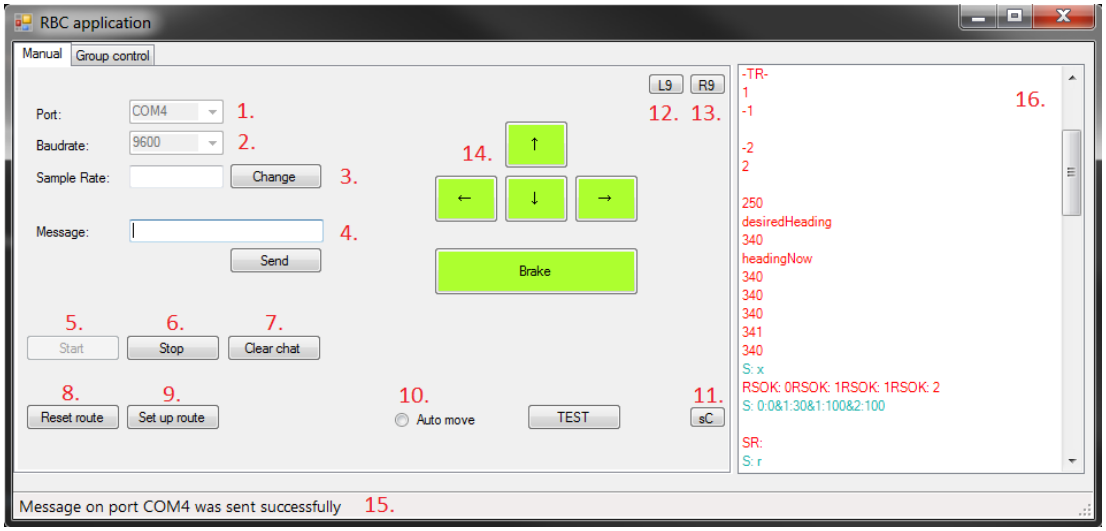
Trasa může vypadat následovně - 0:0&3:100&1:50&3:50&2:50&4:150. Kdy první dvojice definuje robota v manuálním režimu, následuje pohyb dopředu 100cm, pohyb doleva 50cm, pohyb dopředu 50cm, pohyb doprava 50cm a pohyb dozadu 150cm.

Aplikace je rozdělena do dvou majoritních záložek a těmi jsou manuální řízení a řízení skupiny.

6.1 Manuální režim

Manuální režim slouží zejména pro debugování funkcionality robotů, je možné v něm navázat pouze jedno spojení. Poskytuje zpětnou vazbu při testování dílčích částí reakčních a pohybových algoritmů.

V manuálním režimu je možné posílat robotu libovolné příkazy. Pro často používané příkazy (zprávy), jsou implementovány ovládací prvky. Je možné parametrizovat trasu a následně robota přepnout do autonomního režimu. Veškerá komunikace je vypisována do stavového okna.



Obrázek 6.1: Obrazovka manuálního řízení

Kde:
1: Výběr sériového portu
2: Výběr baudrate
3: Změna frekvence vyčítání BT bufferu
4: Box pro zadání zprávy, tlačítko odeslat pod boxem
5: Zahájení komunikace
6: Zastavení komunikace
7: Odstranění všech záznamů z informačního okna (16.)
8: Reset pohybové trasy
9: Nastavení pohybové trasy
10: Aktivace autonomního režimu nebo potvrzení synchronizace
11: Uložení obsahu informačního okna (16.) do souboru
12: Zatočení o 90°vlevo
13: Zatočení o 90°vpravo
14: Navigační tlačítka pro manuální pohyb - robota je možné řídit šipkami a mezerníkem, tyto stisky jsou graficky znázorněny na navigačních tlačítkách
15: Stavový řádek procesů aplikace
16: Informační okno

K parametrizaci trasy se využívá skupina komponent z obrázku č. 6.2 (1.)

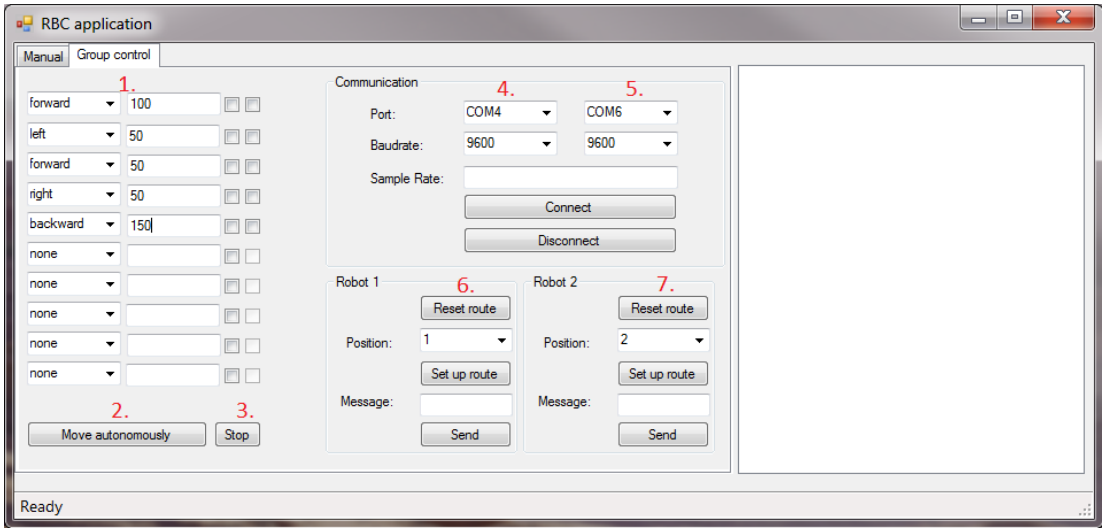
6.2 Řízení skupiny

Při komunikaci se členy skupiny jsou využívány stejné metody jako v kartě manuálního řízení, s tím rozdílem, že není možné skupinu řídit prostřednictvím kláves. V aplikaci je v současné době implementována možnost navázat dvě paralelní spojení. Vzhledem k povaze současného standartu BT by jich bylo možné navázat sedm. Roboti posílají, kvůli případnému dočasnému rušení spojení, vždy tři zprávy. V potaz se bere jedna z této trojice, ostatní jsou filtrovány. Nejvýraznější změnou v implementaci řízení skupiny je komunikační přepínač, který na základě došlé zprávy přepošle zbytku skupiny zprávu odpovídající.

Jedná se o došlou zprávu týkající se lokační synchronizace nebo překážky v trase.

Ve chvíli, kdy roboti dorazí do dílčího cíle trasy, odešlou zprávu $-TR-$, jestliže aplikace obdrží zprávu tohoto typu od všech jedinců skupiny, tzn. že jsou roboti lokačně synchronizováni, pošle jim zprávu x, čímž jim povolí pohyb a roboti pokračují v dosažení cíle trasy.

Jestliže jedinec skupiny narazí na překážku v trase, oznámí tuto skutečnost nadřazené aplikaci zprávou $-O-$, ta ostatním robotům odešle příkaz k zastavení a přechod do reakce - h. V současné implementaci je každá zpráva odeslána aplikací třikrát.



Obrázek 6.2: Obrazovka řízení skupiny

Kde:
1: Sestavení trasy
2: Aktivace autonomního režimu
3: Deaktivace autonomního režimu
4: Parametry spojení s 1. robotem
5: Parametry spojení s 2. robotem
6: Operace nad trasou 1. robota, definice pozice
7: Operace nad trasou 2. robota, definice pozice

Roboti během během manuálního i autonomního režimu posílají aplikaci stavové informace o svých periferiích, případně dosažených částech řídícího kódu. Tyto informace jsou následně zobrazeny v informačním okně.

Přehled stavových hlášení robota:

Popis stavu	Stav
Route set OK	RSOK
Route is undefined	RIU
Target all reached	TAR
Target all reached in reaction	TARIR
Target x reached	TxR
In - set actual direction after turn	I_SADAT
In - react secondary	I_RO
Obstacle in the way	OIW
React second method call follow	rOF
Obstacle center	OC
Obstacle right	OR
Obstacle left	OL
Obstacle in react - restart	OIR
Checking sides start	CSS

Popis stavu	Stav
Set pozition	SP
Set route	SR
Waiting for position	WFP
Position has been set	PHBS
Nothing to update, update after obstacle in way	NTU
Waiting for react route	WFRR
Wrong format of road	WFOR
Obstacle in the way - autonomous mode	-O-
Target reached - autonomous mode	-TR-
Configurations of robot 1 set	CS1
Configurations of robot 2 set	CS2
Reference direction set in react	Rh
Reference direction set	Sh

Obrázek 6.3: Obrazovka řízení skupiny

7. Testování a shrnutí

Testování robotů probíhalo paralelně s hardwarovou konstrukcí a implementací pohybových a reakčních algoritmů. Aby bylo možné testovat jak pohybovou, tak reakční složku řešení, měl prostor, ve kterém byla skupina testována zpravidla 2x3 metry a více. Překážky skupině tvořily tělesa různých tvarů a materiálů.

Při testování byly sbírány výstupy z periferií robotů, které byly následně analyzovány. Na základě analýzy a vizuálního vjemu z funkčnosti řešení byly definovány případné úpravy. Nejdůležitějším nástrojem procesu testování byla nadřazená aplikace, díky které bylo možné pracovat s periferiemi všech robotů najednou.

Výstupem této práce je autonomní skupina robotů se schopností definovaně reagovat na překážky. V každé iteraci testování skupiny bylo na základě pozorování a informací předaných nadřazené upraveno dílčí řešení tak, aby bylo cíle dosaženo.

Vzhledem k tomu, že jsou roboti schopni pohybové a komunikační kooperace a jsou schopni bez narušení integrity skupiny zareagovat na překážku, lze konstatovat, že byl cíl práce naplněn.

7.1 Shrnutí

Během procesu testování bylo shromážděno množství dat, na základě kterých se následně modifikovaly softwarové nebo hardwarové části práce.

7.1.1 Řízení

Vytížení programovatelné desky Arduino Uno je v současné době na hraně stability a to jak z pohledu výkonu, paměťové kapacity tak dostupnosti vstupů/výstupů. Vzhledem k jednovláknové architektuře procesoru není možné procesy, u kterých by to bylo vhodné, obsloužit paralelně a tím se zvyšuje čas vykonání určitého kódu, respektive čas dosažení tohoto kódu. Obsazenost vstupů/výstupů desky je téměř 100%.

V případě rozšíření funkcionality řešení by bylo nutné použít výkonnější desku s větším počtem vstupů/výstupů. Na současné modelové řešení je ale deska stále dostačující.

7.1.2 Konstrukce robota

V původním návrhu práce nebyla věnována dostatečná pozornost regulačním a korekčním subsystémům autonomního robota, proto byl pohyb značně nestabilní, robot nedokázal udržet směr. Na základě tohoto problému byla implementována statická regulace, která se ukázala být nedostačující. Byla nahrazena PID regulací, která je schopna systém zregulovat dostatečně rychle.

Po úspěšném usměrnění robota byl shledán problém ve stabilizačním kole, které bylo při otáčení příčinou, pro pohybové subsystémy nezaznamenatelného,

vyosení. Podpůrné kolo bylo nahrazeno stabilizační kuličkou, která tuto chybu eliminovala.

Změn oproti původnímu návrhu doznalo také brzdění, které bylo v původním řešení založeno na principu snížení rychlosti motorů. Dojezd robota který následoval ovšem způsobil chybu v odometrickém měření. Tato chyba byla vyřešena implementací brzdění kontra, kdy se při snížení rychlosti zároveň invertuje směr otáček motorů.

Rozmístění ultrazvukových čidel bylo z původních 45°oproti středovému čidlu upraveno na 85°a to z důvodu nekompletního pokrytí prostoru před robotem.

7.1.3 Software

Změny v programu robotické jednotky a nadřazené aplikace reflektují změny v hardwarovém řešení. Jak pohybové, tak reakční algoritmy byly navrženy a beze změn vůči tomuto návrhu implementovány.

Závěr

Cílem práce bylo vytvořit skupinu mobilních autonomních robotů a algoritmizovat jejich reakce na neočekávané podněty. Tento problém byl vzhledem ke své rozsáhlosti rozdělen do několika dílčích problémů, které byly postupně řešeny a následně provazovány.

Prvním krokem k úspěšnému naplnění cíle byla analýza dostupných řešení práce, technologií a algoritmů. Na základě tohoto šetření a porovnání výstupních informací byla stanovena teoretická východiska práce.

Poté byla vytvořena robotická skupina a byly implementovány základní logické a pohybové algoritmy.

Výsledné řešení pohybové a navigační problematiky práce není v oblasti robotiky ničím novým, byla zvolena kombinace dostupných a osvědčených principů. Kvůli udržení nezávislosti skupiny na prostoru působení byl navržen a následně implementován navigační algoritmus, který umožňuje skupině se pohybovat prostorem, bez závislosti na explicitních navigačních periferiích, tzv. pohyb v síti. Úspěšnému řešení tohoto problému předcházela nutnost analýzy a porozumění většině dostupných navigačních a pohybových principů.

Majoritní problematikou práce byla reakce skupiny na neočekávané podněty. Vzhledem k nedostupnosti řešení této problematiky bylo nutné vytvořit řešení zcela nové. Reakční algoritmus byl rozdělen do dvou složek - prvotní a sekundární reakce, roboti si v něm vytváří a předávají reakční trasu. Algoritmus je dostatečně dynamický na to, aby zajistil reakční stabilitu i v početných robotických skupinách.

Při rozboru dostupných řešení bylo doznáno vhodnosti využití nadřazené komunikační aplikace, která vede k tzv. centralizaci komunikace skupiny. Byla vytvořena nadřazená aplikace, která zajišťuje a spravuje komunikaci mezi roboty. Dále se využívá k parametrizaci periferií robota, případně jeho vnitřních proměnných. Aplikace zprostředkovává stavy vstupů a výstupů robota a slouží jako diagnostický nástroj.

Za použití komerčně dostupných logických obvodů a minimálního množství základních senzorů se podařilo vytvořit skupinu schopnou autonomního pohybu. Narušení autonomního pohybu neočekávaným podnětem je skupina schopna vyřešit vytvořením nové, nekolizní trasy, která je mezi roboty následně rozdistribuována. Po úspěšném projetí reakční trasy jsou roboti schopni nalézt původní trajektorii a v ní pokračovat.

Seznam použité literatury

[1] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení.* Vyd. 1. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-730-0141-1.

[2] BORENSTEIN, J. - EVERETT, H.R. - FENG, L. - WEHE, D. *Where am I? Sensors and Methods for Mobile Robot Positioning.* University of Michigan, new edition edition, 1996.

[3] *Skupina* [online]. 13. 10. 2014 [cit. 2014-11-17]. Dostupné z: http://cs.wikipedia.org/wiki/Skupina

[4] KLVAŇA, Marek. *Tvorba formací skupiny mobilních robotů: Formation creating of mobile robots groups* [online]. Brno: Vysoké učení technické, Fakulta strojního inženýrství, 2007 [cit. 2014-11-13]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: http://autnt.fme.vutbr.cz/szz/2007/BP_Klvana.pdf. Bakalářská práce. Vysoké učení technické v Brně.

[5] NAFFIN, David James. *Multi-robot formations: Rule-based synthesi and stability analysis* [online]. Faculty of the graduate school university of southern California 2006 [cit.2.3.2007]. Dostupné z: http://cres.usc.edu/Research/files/naffin_thesis_2006.pdf

[6] *Profi Elektrika* [online]. 2008 [cit. 2015-02-04]. Dostupné z: http://elektrika.cz/

[7] *Introduction: PID Controller Design* [online]. 2012 [cit. 2015-02-07]. Dostupné z: http://ctms.engin.umich.edu/CTMS/

[8] KOŠNAR, Karel. *Mobilní robotik a. In: Mobilní robotika* [online]. 2007 [cit. 2015-04-09]. Dostupné z: http://www.roznovskastredni.cz/dwnl/pel2007/06/Kosnar.pdf

Seznam obrázků

Přílohy

Přiložené CD obsahuje elektronickou verzi práce ve formátu PDF a zdrojový kód logiky robotů a nadřazené aplikace. Software robotů je přiložen ve formě projektu vývojového prostředí Visual Studio 2013, plugin Visual Micro. Software nadřazené aplikace je přiložen jako projekt vývojového prostředí Visual Studio 2013.

V kořenovém adresáři CD je umístěna elektronická verze bakalářské práce.
Ve složce SW_Robot je projekt obsahující řídící logiku robotů.
Ve složce SW_Aplikace je projekt obsahující program nadřazené aplikace.