Mendelova univerzita v Brně
Provozně ekonomická fakulta

Návrh a implementace řídicího software mobilního robotu

Diplomová práce

Vedoucí práce:
ing. Vít Ondroušek Ph. D.

bc. František Ostřížek

Brno 2015

zadani

Děkuji svému vedoucímu diplomové práce Ing. Vítovi Ondrouškovi Ph. D. a všem z týmu Aistorm za cenné a konstruktivní rady, čas a trpělivost, které mi v průběhu tvorby závěrečné práce věnovali. Rád bych také poděkoval Robertu Čížkoví a Martinu Pánkovi za práci, kterou odvedli při konstrukci hardwarové části řešení.

Abstract

Ostřížek, F. Desiging and implementation software of mobile robot. Master thesis. Brno, 2015

The main aim of this diploma thesis is the design and implementation of the control unit of the autonomous wheeled robot. The designed control software consists of several layers and various modules which deals with the problem of localization in the known map of the environment and path planning. Problem of localization is solved using mathematical model of movements of the robot and the GPS sensor. The problem of the roads detection is solved using image analysis. The designed solution is able to ensure autonomous movements of the robot along the roads of selected city parks.

Keywords: localization, robot, design, control, control unit, BuggyMan

Abstrakt

Ostřížek, F. Návrh a implementace řídicího software mobilního robotu. Diplomová práce. Brno, 2015

Hlavním cílem této diplomové práce je návrh a realizace řídicí jednotky autonomního kolového robotu. Navržený ovládací software se skládá z několika vrstev a různých modulů, které se zabývají problémem lokalizace ve známém mapě prostředí a plánování cesty. Problém lokalizace je řešen pomocí matematického modelu pohybu robotu a snímače GPS. Problém detekce cesty je řešen za použití obrazové analýzy. Navržené řešení je schopno zajistit autonomní pohyb robotu podle cesty vybraných městských parků.

Klíčová slova: lokalizace, robot, návrh, řízení, řídicí jednotka, BuggyMan

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Návrh a implementace řídicího software mobilního robotu**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 20. května 2015 .....

Obsah

1 Úvod a cíl

1.1 Úvod

V rámci týmu Aistorm Mendelovy univerzity v Brně vznikl projekt BuggyMan, tento projekt řeší autonomního robotu, který je určen k pohybu po zpevněných cestách parků. Základ platformy tvoří modelářský podvozek auta s řízeným elektrickým pohonem a servo pohonem pro zatáčení. Vlastnosti podvozku lze modelovat pomocí Ackermanova řízení.

Pokud po mobilním robotu požadujeme samostatné řešení úlohy přemístění se ze startovní pozice do cílové, musí řídicí jednotka vyřešit mnoho dílčích úkonů, např. čte data ze senzorické soustavy a rozhoduje o nastavení akčních veličin. Hlavní řešené problémy jsou plánování mapování a lokalizace. Na základě použitých algoritmů pro zpracování těchto informací vyhodnocuje svoji pozici v globálním nebo lokálním souřadném systému a snaží se každým krokem udržet na nalezené cestě a přiblížit se svému cíli. Dosažení posledního bodu v naplánované cestě rozděluje úlohu lokalizace podle použitého prostředí na vnitřní a vnější. Robot BuggyMan je hardwarově i softwarově vybaven pro venkovní použití, zejmena pro zpevněné cesty městských parků. V tomto prostředí jednotka vyhledává cestu za použití kamerových snímků a plánuje své úlohy v lokálním souřadném systému. Robot rovněž provádí lokalizaci v globálním souřadném systému známé mapy. Veškeré výpočty pro potřeby provádění lokalizace a plánování jsou realizovány ve výpočetní jednotce, která je umístěna přímo na robotu. Všechny události a přechody stavů jsou zaznamenávány do souborů pro pozdější analýzu.

K vývoji algoritmu globální lokalizace a lokálního plánování probíhá sběr dat v arboretu Mendelovy univerzity. Část práce je zaměřená na zpracování a použití zaznamenaných dat k testování hlavního programu. Uložená data mají jednotnou strukturu. Vložená časová značka umožňuje sestavit přesnou posloupnost volání rutin offline.

1.2 Motivace

Projekt BuggyMan vznikl za účelem účasti na soutěžích Robotem rovně, Robotour, Robofield. Jako testovací prostředí je zvoleno Arboretum Mendelovy univerzity, které se nejvíce podobá soutěžním podmínkám. Toto prostředí vyzkouší hardwarovou i softwarovou vybavenost robotu a zároveň klade nemalé požadavky na realizované řešení. Tím se rozumí různé typy povrchů, cesty s významným sklonem, velká variabilita prostředí. Z pravidel soutěží a provedení veřejně přístupných parků lze očekávat že se robot bude pohybovat po zpevněných cestách. Robot tedy autonomně projíždí vytyčenou trasu parkem než dosáhne stanoveného cíle. Hlavní motivací pro vytvoření této práce je navržení řídicí jednotky autonomního robotu tak, aby byl robot schopen obstát ve velké konkurenci na soutěžích konaných v uvedených venkovních podmínkách.

1.3 Cíl

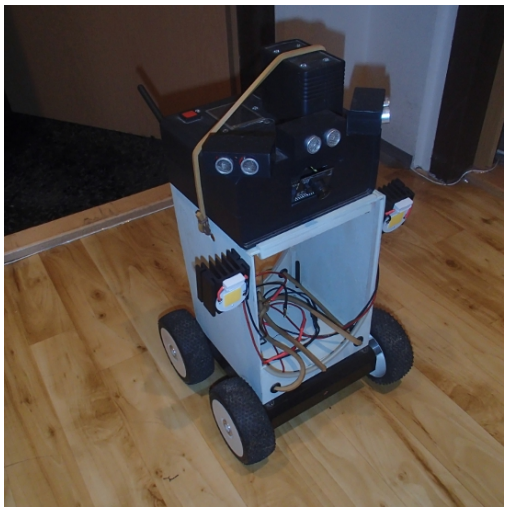
Cílem práce je navrhnout řídicí software kolového mobilního venkovního robotu, který bude schopen autonomního pohybu po zpevněných cestách městských parků. Na základě zpracovaných oborných textů zabývajících se řešením lokalizace ve známé mapě je navrhnuta logika a struktura softwaru řídicí jednotky ovládající připojené komponenty robotu. Navržený koncept autor implementuje pomocí jazyku Java do funkčního celku a testuje úspěšné projetí vytyčené trasy v areálu arboreta Mendelovy univerzity. Vhodnost svého návrhu testuje pomocí jednotkových, integračních a reálných testů. Výstupy z těchto testů jsou analyzovány a zhodnoceny na závěr práce.

2 Současný stav problematiky

2.1 Popis podobných projektů

Mezi již existujícími funkčními jednotkami, které se pravidelně účastní soutěží, jsou prezentováni roboti univerzitních týmů a týmů, které se věnují robotice jako zájmové činnosti. V univerzitním přehledu najdeme jména jako AutoLut2 (Lodz University of Technology team), Cogito(je odnož členů skládající se ze tří týmů Short Circuits, Mart a Eduro), JECC (Robert Koch High School of Deggendorf), Smelý Zajko (Univerzita Komenského v Bratislavě), URPi (Slovenská technická univerzita v Bratislavě), Logio (Univerzita Karlova v Praze), FoG (České vysoké učení technické v Praze), Bender a Red (Vysoké učení technické v Brně) a mezi nadšenci ze střední evropy se naleznou jména Husky (Česko), AmBot(Česko), ArBot(Česko), Blade XXII(Slovensko), Istrobotics(Slovensko), NDTeam (Česko), Plecharts (Česko), Radioklub Písek (Česko), Tapas (Polsko) a mnoho dalších. Všechny výše zmíněné týmy se snaží zapsat do soutěží a řešit společné problémy mobilní robotiky každý po svém. V následujícím textu autor představuje vybrané mobilní roboty, které dosáhly úspěchu.

2.1.1 ARBot



Obrázek 1: čtyřkolový diferenciální podvozek ARBot

Zdroj: http://www.arbot.cz/image.axd?picture=%2F2013%2F10%2FARBotV2.jpg

Na obrázku č. 1 je vidět, že ARBot je robotické vozidlo konstruované na komerčním podvozku A4WD1 od firmy Lynxmotion. Robot je osázen čtyřmi koly o průměru 110 milimetrů. Každé kolo je samostatně poháněné a opatřené enkoderem. Robot je vybaven kamerou, GPS (Global Positioning System), AHRS (Attitude And Heading Reference System) jednotkou a třemi ultrazvukovými senzory.

O výpočty se stará základní deska s označením DSP BF537. Veškerá komunikace uvnitř robotu je zprostředkována standardizovaným protokolem I2C a RS232. Řídicí program robotu je psán v jazyce C a výkonově kritické rutiny v asembleru. Pro detekci cesty je použito kamerových snímků, které jsou segmentovány histogramem do černého a bílého prostoru. Vychází se z představy, že hranice detekované cesty jsou reprezentovány řádkovými extrémy bílých bodů od středu obrazu. Řídicí algoritmus se snaží udržet robot uprostřed této nalezené cesty. Barevně odlišné předměty nejsou detekovány jako cesta a to vede k jejich automatickému objíždění. Obraz je postupně od spodního řádku segmentován pro výpočet středu cesty a tyto body se proloží přímkou. Od směrnice přímky se pomocí konstanty odvodí požadovaný úhel otočení robotu. Tento postup má několik nevýhod:

- algoritmus je citlivý na střídání světla a stínu na cestě
- nebere v úvahu projekci kamery
- algoritmus nefunguje v mezních situacích, kdy robot je postaven na okraji cesty v jejím směru.

Ultrazvukový senzor slouží pro detekci překážek, jejich objíždění a zastavení v případě hrozící kolize. Pokud klesne měřená vzdálenost na levém či pravém ultrazvukovém senzoru pod určenou mez, začne se robot otáčet na stranu kde je více prostoru. Údaje z encoderů, AHRS a GPS jsou použity k určení pozice a orientace robotu. Po dlouhé cestě vývoje se projektu ARBot podařilo vyhrát soutěž Robotour v roce 2013 a účastnit se soutěže Robotem rovně.(ARBot, 2008)

2.1.2 Robo@FIT

Na obrázku č. 2 je robot postaven na komerčním podvozku Pioneer 3AT. Tento podvozek má poháněná všechna 4 kola a je diferenciálního typu. Jeho výpočty zpracovává vezený notebook. Program je založen na robotickém operačním systému ROS a software je psán v programovacím jazyce C++, Java a Python. Pro detekci cesty využívá primárně dvojici firewire kamer, ale robot je vybaven také 3D laserovým dálkoměrem Velodyne, GPS a AHRS jednotkou.(Robo@FIT, 2014) Lokalizaci robotu a vzdálené ovládání zpracovává externí zařízení s operačním systémem android. Tato externí jednotka využívá mapového serveru google a navrženého GUI(Graphical User Interface) pro zobrazení informací přenesených z robotu ke klientovi. Spojení mezi robotem a tabletem je realizováno ad-hoc sítí a navržený komunikační protokol přenáší informace mezi nimi. (Pavlenko, 2014)

2.1.3 Tým Short Circuits Prague

Robot vystavěn na podvozku modelu monster trucku velikosti 1:5. Hlavní výpočetní jednotku tvoří standardní notebook s Windows. Hardware je monitorován deskou Arduino Duemilanove. Senzorické vybavení robotu tvoří magnetický kvadraturní enkodér, kompas, sonary, lidar, GPS přijímač, IMU a kamera. Software je psaný

Obrázek 2: čtyřkolový diferenciální podvozek Robo@Fit

Zdroj: http://robotika.cz/competitions/robotour/2013/teams/robo-fit.jpg

v jazyce Python. Informace o stavu HW získává jednočip ATMEGA328, který je v pravidelných intervalech posílá dále do hlavního programu. Hlavní program zajišťuje transformace stavu HW do jednotek a struktur nezávislých na HW platformě, PID kontrolér rychlosti, lokalizaci z odometrie, GPS a kompasu, detekce překážek a reakce na ně a konečně navigaci po naplánované trase.

2.1.4 RoboAuto

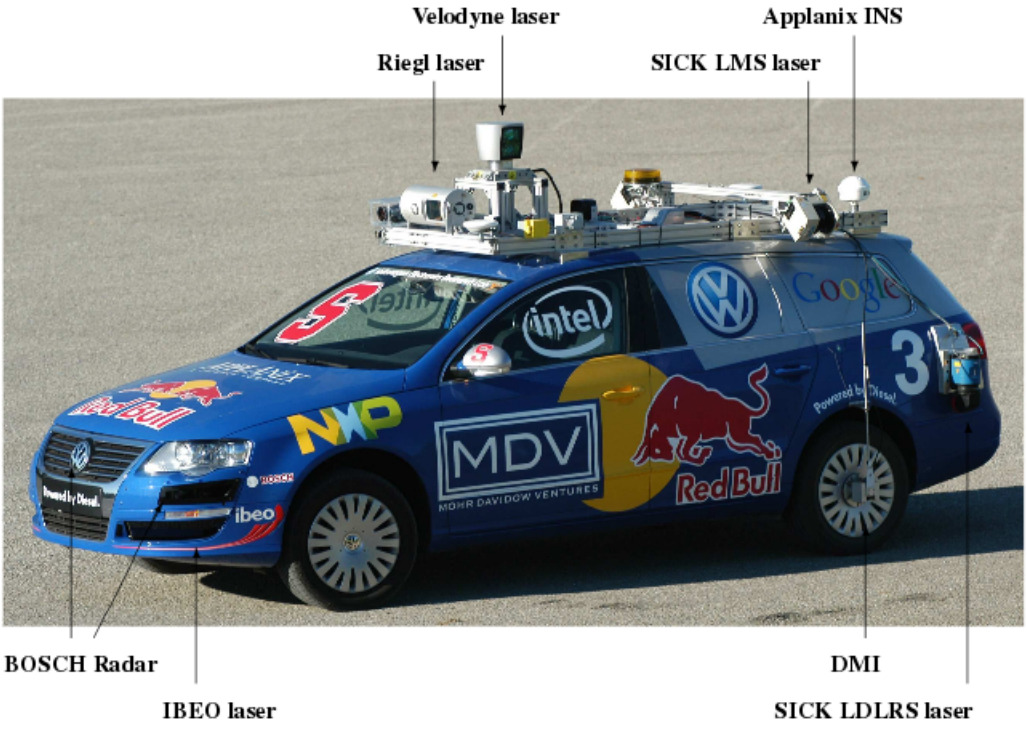
RoboAuto je projektem skupiny odborníků a akademických pracovníků v oblasti softwarového vývoje a umělé inteligence se zájmem o technologie automatického řízení automobilu. Cílem je vytvoření provozuschopné verze automobilu s plným řízením pomocí počítače. Jedná se o akademický projekt, na kterém lze testovat různé nové metody snímání okolí, řízení automobilu apod. Projekt je realizován pod hlavičkou Fakulty Informačních Technologií Vysokého Učení Technického v Brně.(Roboauto, 2012)

Jako podprojekty této skupiny vznikly roboty Karlík a Quido. Jedná se o platformy založené na Ackermanově podvozku. Jsou vybaveny laserovými senzory, inerciální jednotkou, GPS, ultrazvukovými snímači vpředu, kompasem a kamerou. Řízení zajišťují dva notebooky propojené přes 100 MBit switch. Software je pak psán pod Windows a jedná se o kombinaci aplikací C++, Delphi a Java. Vzájemná komunikace implementovaných modulů je přes TCP/IP.

Hlavním přínosem je navigační architektura, která spočívá v myšlence sestavování několika relativně jednoduchých přístupů k řešení částečných pravděpodobnostních problémů v autonomní navigaci a zvýšení robustnosti celého systému pomocí vysoké úrovně uvažování. Všechny moduly fungují zcela samostatně a decentralizo-

vaně. Uvnitř robotu běží přibližně 25 modulů v 6 vrstvách (HW, reaktivní vrstva, plánování cesty, odhad silnice, mapování, lokalizace). Mapové podklady jsou použity z datového souboru v Open Street Map formátu. Robotu Quido vyhrál robotour v ročníku 2011 a o rok dříve byl druhý. (Herman, 2011)

2.1.5 projekt Junior



Obrázek 3: Autonomní vozidlo Standrodské univerzity

Zdroj: (Standford, 2005, str. 2)

Projekt autonomního vozu Junior (obrázek 3) vznikl na Stanfordské univerzitě a snaží se zapojit do soutěží vypisovaných společností DARPA (Defense Advanced Research Projects Agency). Soutěže jsou známy pod názvy Urban Challenge nebo Grand Challenge a mají prověřit možnosti bezpilotních vozů v běžném provozu. Projekt Junior je postaven na upraveném podvozku Wolkswagen Passat Wagon. Je vybaven inerciální navigací, po stranách a ve předu laserovými senzory značky SICK a RIEGL. Na střeše je umístěn laserový senzor Velodyne. Po obvodu automobilu jsou umístěny vzdálenostní senzory BOSH. Struktura použitého programu nese šest vrstev, kde na nejnižších úrovních je senzorické vnímání okolí. a nad těmito vrstvami jsou navigace, lokalizace, mapování a uživatelské rozhraní. Napříč všemi vrstvami

použitého softwaru jsou služby, které umožňují komunikaci mezi sebou a zaznamenávají průběh jízdy do souborového systému vezených počítačů. Jedna z klíčových úloh je lokalizace vozidla oproti známé mapě ve formátu RNDF (The Route Network Definition File). Měření z vozu Junior pomocí laserových skenů je zatíženo průměrně 80 cm odchýlením oproti realitě. (Standford, 2005)

2.2 Lokalizační algoritmy

V této kapitole jsou popsány různé přístupy k problému lokalizace autonomních robotů. Má li řídicí jednotka k dispozici mapu prostředí, pak algoritmus lokalizace řeší nalezení polohy robotu v této mapě. Nejčastěji používané skupiny lokalizačních algoritmů jsou rozšířený Kalmanův filtry a jejich modifikace, partikulární filtry, aproximační metody, neuronové sítě a další. V případě že řídicí jednotka nemá k dispozici mapu prostředí musí řešit tzv. synchronní lokalizaci a mapování dále jen SLAM (Simultaneous localization and mapping). Jedná se o výpočetní problém, výstavbu nebo aktualizaci mapy neznámého prostředí a zároveň udržení přehledu o umístění agentů v něm, kde agent je reprezentován souřadnicemi a směrem. Lokalizovaného agenta si můžeme představit jako proměnou množinu signálů, které můžeme rozdělit na spojité nebo diskrétní v závislosti na použité funkci pro sběr dat. Pod zpracováním signálu rozumíme transformaci daného signálu na jiný signál s požadovanými vlastnostmi. Signál si zpravidla představujeme jako funkci času, i když se často setkáváme s vícerozměrnými signály, které jsou navíc funkcemi prostorových proměnných, jako je tomu při zpracování obrazů nebo lokalizaci.(Smekal, 2004)

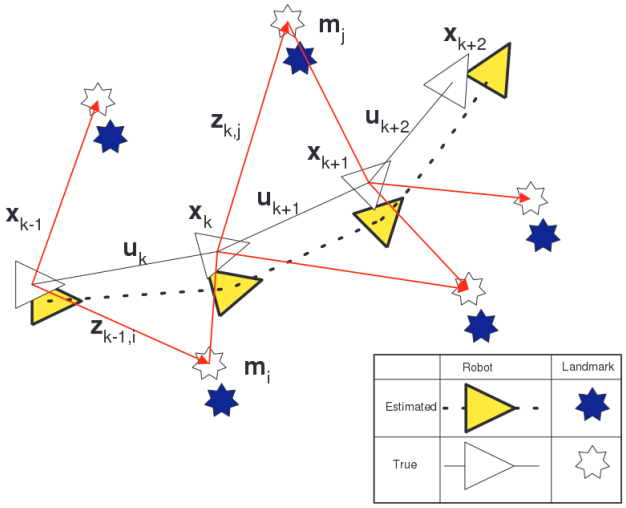
Lokalizaci můžeme rozdělit na lokální a globální. Lokální systém je nadřazen globálnímu systému a slouží především pro zajištění reaktivního chování (detekce překážek a vyhnutí se kolizím). Systém pracuje se souřadným systémem robotu, tedy například senzor na robotu má určitý délkový rozsah, ve kterém detekuje překážku a systém navigace tomu musí být přizpůsoben, aby mohl včas rozhodovat při možné kolizi s překážkou. Z toho vyplývá, že o výběru senzorů nerozhodují pouze jejich vlastnosti, ale i fyzické rozměry robotu. (Novák, 2007)

Úlohou globálního systému je určit polohu robotu na úrovni jeho pracovního prostoru. V případě projektu BuggyMan se jedná o prostředí parku. Globální lokalizaci lze určit pomocí relativní navigace. Ta je založena na principu měření přírůstků změny polohy a orientace robotu. Změna je vztažená k počáteční pozici robotu a nebo k pozici, kde byla naposledy určena absolutní poloha. Absolutní poloha využívá ke stanovení přesné polohy robotu vztahu k referenčním bodům. Principu referenčních bodů v kosmickém prostoru využívají systémy GPS (Global Point System), Gallileo, Glonas. Na zemi lze použít i signál z pozemních stanic pro mobilní pásmo a principu triangulace nebo trilaterace.

Následující obrázek č. 4 ilustruje problém lokalizace, kde:

- x_k je stavový vektor popisující lokaci a orientaci robotu

- u_k je kontrolní vektor aplikovaný v čase $k-1$ k řízení robotu ve stavu x_k v čase k
- m_i je vektor popisující lokaci i-tého landmarku, jehož skutečné umístění předpokládá neměnnou dobu
- z_{ik} je pozorování položené z pozice robotu i-teho landmarku v čase k, když je spatřeno více landmarku nebo když specifický landmark není relevantní k diskuzi je spatření zapsáno jednoduše jako z_k



Obrázek 4: Schéma principu techniky lokalizace

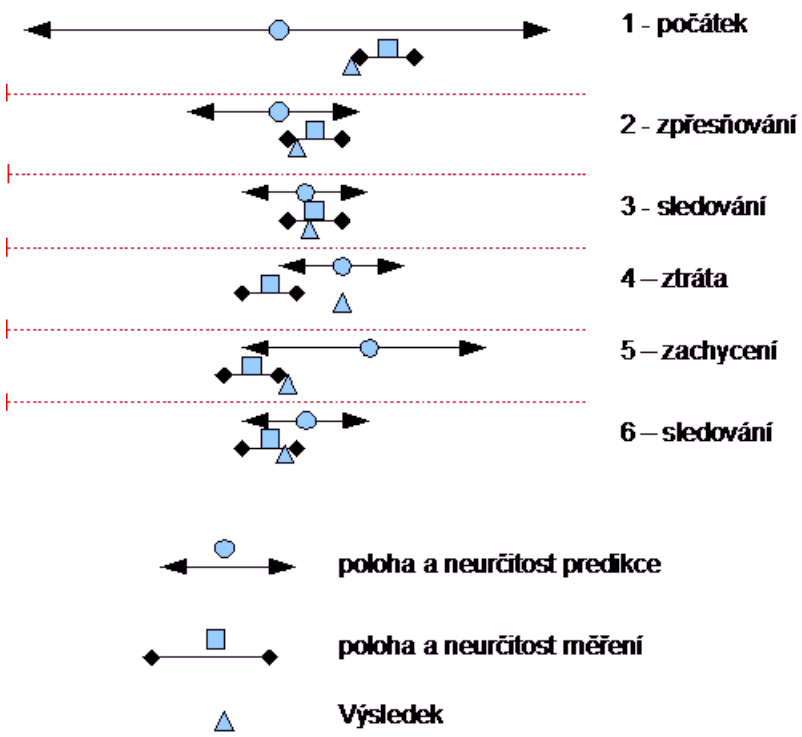
Zdroj: (Durrant-White, 2006, str. 2)

2.2.1 Kalmanovy filtry

Kalmanův filtr je vhodný například pro trasování objektů. Umožňuje predikovat polohu, upřesnit ji na základě současného měření, a to vše v návaznosti na kvalitu hodnot. Kvalita hodnot je určena pomocí rozptylu proměnných. Výsledkem je odhad trajektorie na základě nepřesných hodnot.

Důležitou součástí Kalmanova filtru je model systému (dynamický systém), který budeme sledovat. Na jeho základě vytváříme predikce. V okolí predikce můžeme očekávat výskyt hledaného objektu. Pokud objekt nenalezneme, pokračuje dále v predikci pro další krok, pouze prohledávané okolí se zvětší. V případě, že objekt nalezneme, provedeme korekci současné polohy (na základě predikce a měření) a pokračujeme v dalším kroku. Prohledávané okolí bude menší, protože předchozí hodnota je díky úspěšnému měření (detekci objektu) přesnější.

Obrázek č. 5 ukazuje stručný postup mechanizmu kalmanova filtru v několika krocích:



Obrázek 5: Schéma principu Kalmanova filtru

Zdroj: (Richter, 2008)

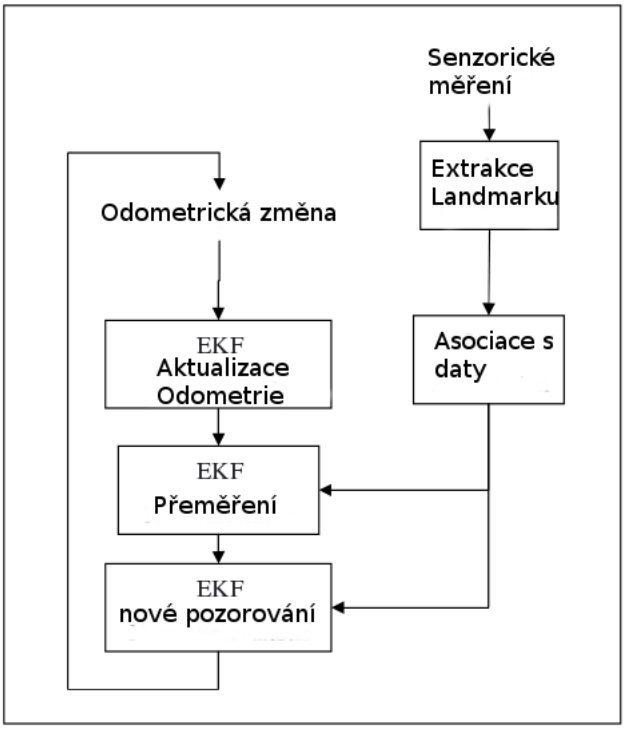
1. počátek – určíme nejlepší předpokládanou polohu (predikce polohy) a její neurčitost stanovíme podle toho jak jí věříme (zde jí tedy moc nevěříme protože je velká). Následně v rozmezí neurčitosti hledáme (měříme) aktuální polohu. Ta má neurčitost danou kvalitou měření. Výsledná poloha se určí jako kombinace predikce a měření (jako vážený průměr poloh a neurčitostí).
2. Zpřesňování - na základě minulých hodnot a stanoveného modelu se vypočte predikce. Neurčitost je dána vývojem minulé neurčitosti podle daného modelu (a tedy se měřením zpřesní, ale vlivem neznámého/nekontrolovaného „vývoje“ opět vzroste. Proběhne měření a výpočet nové výsledné polohy polohy.
3. Sledování - predikce se při delším sledování zpřesňuje (váhy měření a predikce se sbližují, filtrace („vyhlazení“) průběhu.
4. Ztráta – v případě, kdy nedojde k měření není možné korigovat polohu a proto nedojde ani k jejímu zpřesnění. Dochází tedy k nárůstu neurčitosti predikce výsledku, což se projeví i v dalším kroku. Ke ztrátě dochází při nepředvídané změně polohy (mimo popis modelem a jeho parametry), nebo při překrytí nebo splynutí objektů vedoucímu k tomu, že nedojde k měření.
5. Zachycení – jelikož se při ztrátě zvětší neurčitost predice, narůstá i okruh, ve

kterém hledáme změřenou hodnotu. Po jednom nebo více krocích dojde k zachycení, a na jeho základě i ke zpřesnění polohy.

6. Sledování

(Richter, 2008)

Cílem procesu je použití prostředí k aktualizaci polohy robotu. Protože používané senzory jsou zatíženy chybou a výpočty zaokrouhlováním je zapotřebí fúze dat. Pomocí extrakce a opakovaným měřením lze chyby opravovat při pohybu robotu. EKF (Extended Kalman Filter) je odpovědný za aktualizaci stavů, kde si robot myslí, že stojí na základě získaných vlastností o prostředí. Tato funkcionalita se běžně nazývá značkování. EKF sleduje odhad nejistoty v pozici a polohách značek prostředí. Lokalizace oproti předešlé pozici se pohybuje s nejistotou nové pozice v aktualizací EKF. Orientační body jsou extrahovány z prostředí na nové pozici a pak spojeny s předchozími značkami. Nové pozorování značek prostředí je potom použito k aktualizaci EKF. Celé schéma procesu je znázorněno na obrázku 6.(Riisgaard, 2005, str. 10-11)



Obrázek 6: Schéma principu Kalmanova filtru

Zdroj: (Riisgaard, 2005, str. 10)

$$\widehat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \widehat{X}_{k-1} \tag{1}$$

Rovnice č. 1 popisuje postup výpočtu. Pamatujte si tedy, že index k znazorňuje stavy. Zde je možné uvažovat v diskrétních časových intervalech, jako například k = 1 znamená, 1ms, k = 2 znamená, 2ms.

Naším cílem je najít odhad $\widehat{X}$ ve stavu k, signálu x. a chceme ji najít za každým následujícím k. I zde, je naměřená hodnota Z_k. Mějme na paměti, že si nejsme zcela jisti těmito hodnotami. V opačném případě nebudeme potřebovat dělat všechno a hodnota K_k se nazývá „Kalman Gain“ (což je klíčový bod), a $\widehat{X}_{k-1}$ je odhad signálu na předchozím stavu.

Jediná neznámá složka v této rovnici je zisk Kalmanova zesílení. Vzhledem k tomu, že máme naměřené hodnoty, a už máme předchozí odhad signálu. Měli byste počítat Kalmanovo zesílení pro každý následný stav. To není snadné samozřejmě, ale máme všechny nástroje, jak to udělat.(Bilgin, 2009)

2.2.2 Bayesovské filtry

Bayes filtry jsou algoritmy použité v informatice pro výpočet pravděpodobnosti k odvození pozice a orientace robotu. V podstatě umožňují, aby průběžně aktualizovali své nejpravděpodobnější pozice v souřadnicovém systému, na základě naposledy získaných dat senzorů. Je to rekurzivní algoritmus a skládá se ze dvou částí předpovědi a inovace. V případě, že proměnné jsou lineární a mají normální rozdělení, pak je výstup Bayesova filtru skoro shodný s výstupem Kalmanova filtru.

V jednoduchém příkladu, se robot pohybuje po celé mřížce a může mít několik různých senzorů, které mu umožňují snímat informace o jeho okolí. Robot začíná s jistotou ve výchozí pozici. Pokud se vzdaluje, zvětšuje se jeho nejistota ve srovnání s předchozí polohou.(Recursive Bayesian estimatin, 2015)

Mezi nevýhody tohoto algoritmu patří velká výpočetní náročnost, možnost zvolit špatné předpoklady do výpočtu byasovské formule, velká míra nízké pravděpodobnosti výskytu pozice na cestě. Pokud budeme vycházet z Byasovské rovnice č. 2 dostaneme vztah, kde na levé straně je pravděpodobnost výskytu jevu při konfiguraci x dává y. To lze vypočítat jako vztah pravděpodobnosti krát důležitost pozorování lomeno důkaz.

$$\begin{aligned} P(y|x) = P(y|x)P(y) = P(y|x)P(x) \Rightarrow \\ P(x|y) = \frac{P(y|x)P(x)}{P(y)} = \frac{\text{pravděpodobnost} * \text{důležitost}}{\text{důkaz}} \end{aligned} \tag{2}$$

Na základě obecného vztahu Byasovské formule lze tedy lokalizaci formulovat jako rovnici 3.

$$p(x_t|z_{0:t}, u_{0:t}) = \underbrace{p(z_t|x_t)}_{\text{pozorování}} \int \underbrace{p(x_t|z_{t-1}, u_t)}_{\text{pohybový model}} \underbrace{p(x_{t-1}|z_{0:t-1}, u_{0:t-1})}_{\text{důležitost}} dx_{t-1} \tag{3}$$

V rovnici označíme x všechny systémové stavy a z přečtená senzorická data. Odometrie je modelována jako systémový vstup u. Celý předpoklad je prováděn v čase t.

Ohodnocení se obvykle provádí ve dvou krocích: predikčním kroku, který počítá distribuci současného stavu po integraci měření odometrie u_t, a aktualizaci kroku, který integruje poslední pozorování v předpokládané distribuci. V následujícím textu popíšeme podrobně tyto dvě operace.

Predikční krok je znázorněn integrálem v rovnici 3. Ten je tvořen konvolucí předchozího rozdělení stavové pravděpodobnosti. Dále pak důležitosti se kterou je volán pohybový model robotu. Zatímco předchozí důležitosti pochází z předchozího kroku odhadu, je model pohybu funkcí, která odhaduje pravděpodobnost hlavních stavů x_t dávajících startovní pozici ve stavu x_{t-1} a vypočítaný pohyb u_t z odometrie. Distribuce vyplývající z predikčního kroku je obecně méně informativní, než podle dosavadního stavu distribuce. Toto chování zachycuje pokles informací vyplývající z provádění slepých tahů.

Aktualizační krok počítá předpovídanou distribuci po vyhodnocení měření z_t v rovnici 3 označen jako pozorovací model. Ten měří pravděpodobnost aktuálního měření z_t pokud je robot v pozici x_t.

Podle rovnice 3 je pro odhad pravděpodobnosti zapotřebí dvou funkcí, které by měly být známy. První je model pohybu a druhy je model pozorování. Tyto funkce jsou charakteristické na systému a silně ovlivňují chování Bayesova filtru.(Marchetti, 2012)

2.2.3 Bayesovské sítě

Bayesovské sítě jsou reprezentací pro pravděpodobnostní vztahy skrze množinu náhodných proměnných. Poskytují konečnou množinu $X = (X_1, \ldots, X_n)$ diskrétních náhodných proměnných, kde každá proměnná X_i může obsahovat hodnotu z konečné množiny popsanou $Val(X_i)$. Bayesovská síť je označována jako diskrétní acyklický graf (DAG), který kóduje rozdělení pravděpodobnosti přes X. Uzly grafu korespondují s náhodnou proměnnou $X_1, \ldots, X_n$. Hrany grafu odpovídají přímému vlivu přechodu z jedné proměnné do druhé. Každý uzel je označen s distribucí podmíněné pravděpodobnosti (CPD - Conditional Probability Distribution), kde je reprezentován $p(X_i|Pa(X_i))$, kde $Pa(X_i)$ popisuje rodiče v grafu G. Pár (G, CPD) kóduje distribuci pravděpodobnosti $p(X_1, \ldots, X_n)$.

Jeden ze způsobů, jak sestrojit bayesovské sítě je ze znalostí domény. K tomu existují tři kroky:

- rozhodování o počtu a významu proměnných v zúčastněné oblasti
- stanovení přímých vztahů vlivu mezi proměnnými
- určení podmíněné pravděpodobnosti vzhledem ke struktuře sítě

Od druhého kroku byly navrženy metody s cílem usnadnit proces budování bayesovské sítě s kauzálními obory znalostí. Nicméně v mnoha oblastech pouze znalost domény nestačí k vytvoření kompletní sítě. V tomto případě může být síť naučená z dat, pokud jsou k dispozici.

Obecně lze rozdělit tento problém na znalost struktury a neznalost parametrů a problém učení struktury. (Bayes Nets, 2007)

2.2.4 Partikulární filtry

Také známe jako Sekvenční Monte Carlo (SMC) metoda je množina algoritmů odhadující pozici ve stavovém prostoru implementující Bayesianské rekurzivní rovnice. Tyto metody nejsou vhodné pro použití v mnohodimenzionálních kritériích. Vzorky vypočítané partikulárními filtry vlastní hmotnost, která představuje pravděpodobnost, že částice je právě z dané hustoty pravděpodobnosti.

Metoda přebírá částici x_k a pozorování y_k kde model může vypadat následovně:

- $x_0, x_1, \ldots$ je Markovův řetězec prvního řádu, který se vyvíjí v závislosti na distribuci $P_{x_k|x_{k-1}} : x_k|x_{k-1} \sim P_{x_k|x_{k-1}}(x|x_{k-1})$, kde $P_{(x_0)}$ je počáteční inicializace distribuce.
- pozorování $y_0, y_1, \ldots$ jsou podmíněné nezávislé za předpokladu, že $x_0, x_1, \ldots$ jsou známy. Jinými slovy y_k je závislé na x_k a rozdělení pro y_k je psáno jako $y_k|x_k\ simP_{y|x}(y|x_k)$

Příklad systému s těmito vlastnostmi: $x_k = g(x_k - 1) + w_k$
$y_k = h(x_k) + v_k$, kde obě w_k a v_k jsou vzájemně nezávislé a identicky distribuované sekvence se známými funkcemi hustoty pravděpodobnosti $g(\cdot)$ a $h(\cdot)$. Tyto dvě rovnice lze považovat za stavové rovnice a vypadají podobně jako stavové rovnice pro Kalmanův filtr. Jsou li funkce lineární a v Gausově normálním rozdělení najdou rovnice Bayesovskou distribuci. Filtry jsou také průměrem s dostatkem částic, které mohou být přesnější.(Particle filter, 2014)

Hlavním cílem partikulárního filtru je sledovat zájmové proměnné, které se vyvíjejí v průběhu času. Typicky s non-Gaussovým a potenciálně multimodálním souborem dat. Základem metody je zkonstruovat reprezentaci vzorku na bázi celého souboru. Série provedených akcí mění stav každé proměnné, která nějak souvisí s modelem. Kromě toho může být filtr v určitých časech omezován stavem regulované veličiny.

Pro výpočet regulované veličiny se používá více kopií částic z regulované veličiny. Každá částice je spojena s váhou která označuje kvalitu konkrétní částice. Odhad se tedy získává váženým součtem všech použitých částic. Partikulární filtr je rekurzivní algoritmus, který pracuje ve dvou krocích predikce a aktualizace. Po každé akci jsou všechny částice modifikovány podle stávajícího modelu, včetně přidání náhodného šumu. Poté je váha každé částice přehodnocena na základě aktualizace informací které jsou k dispozici. Za konečný počet iterací se všechny částice překontrolují a ty se zanedbatelně malou váhou jsou odstraněny. To je proces převzorkování.

Formálně je tedy pro partikulární filtr definovan soubor pozic robotu podle rovnice 4 jako souřadnice x, y a otočení robotu.

$$\mathrm{x}^t = [x^t, y^t, \theta^t] \tag{4}$$

Takto definovaná pozice je reprezentována jako množina M částic dle rovnice 5. Index j zde označuje částici nikoliv pozici robotu. Každá částice existuje v kopiích, které jsou při převzorkování filtrovány nebo sjednoceny.

$$S_i^t = [\mathrm{x}_j^t, w_j^t] : j = 1 \ldots M \tag{5}$$

Pro odhad pozice robotu je využito tří rozdílných metod vyhodnocení odhadované pozice.

- vážený průměr
- nejlepší částice
- vážený průměr v okolí nejlepší částice

Pro část predikce je použito pouze numerického modelu robotu v globálním souřadném systému. V části aktualizace je použito senzorického měření a vyhodnocení vlastní pozice pouze na základě naměřených dat. Funkce ohodnocující váhy částic při každém kroku počítá rozdíl mezi vypočítanou a naměřenou pozicí viz rovnice 6. Alternativní metodou vážící funkce může být jakýkoli rozdíl mezi pozicí a orientací.

$$w_j^t = asin\left(\left(\frac{\text{otočení Robotu}}{\text{ujetá vzádlenost}}\right)_{\text{vyp}}\right) - asin\left(\left(\frac{\text{otočení Robotu}}{\text{ujetá vzádlenost}}\right)_{\text{mer}}\right) \tag{6}$$

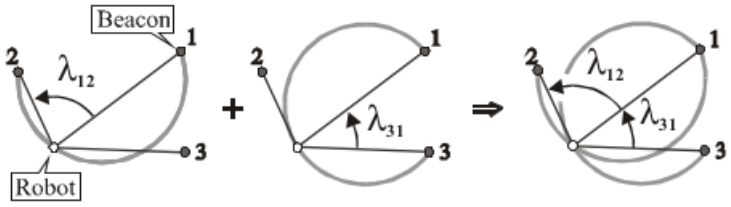
Výsledný odhad pravděpodobnosti pozice robotu se provede pomocí byasovské formule. (Rekleitis, 2008)

2.2.5 Triangulace

Triangulace s aktivními majáky je široce používaný přístup v absolutní lokalizaci mobilních robotů. Algoritmus triangulace umožňuje regulaci a lokalizaci robotu v rovině. Tento systém je založen na měření pozice robotu relativně vzhledem k umístění lokalizačních majáků na známých pozicích. K lokalizaci robotu lze využít i princip trilaterace, který je založen na měření vzdálenosti mezi robotem a majákem. Lokalizaci triangulací lze provádět za pomocí geometrické triangulace, tří nebo dvou kružnic, Newton-Raphson opakovaným vyhledáváním a další.

Pod pojmem maják se nemusí rozumět pouze aktivní prvek vysílající signál směrem k příjemci ale i skupina významných charakteristických bodů umístěných v prostředí, kde se robot pohybuje. Lokalizaci pomocí tří objektů a metody kružnic znázorňuje obrázek č. 7. Použitý přístup lokalizace pomocí kružnic naráží na problém, kdy se robot a majáky nacházejí na totožné kružnici. Geometrická triangulace rozeznává signály v kartézské rovině a rozeznává polohu a vzdálenost jednotlivých

majáků mezi sebou. Omezení triangulačních metod a jejich výpočtů je v označení bodů vzhledem k souřadnému systému robotu a současně musí být svíraný úhel krajních majáků menší než 180 °.(Sena Esteves, 2008)



Obrázek 7: Tří objektová triangulace pomocí kružnic

Zdroj: (Sena Esteves, 2008)

2.2.6 Markovovi řetězce

Markovův řetězec popisuje obvykle diskrétní náhodný (stochastický či pravděpodobnostní) proces, pro který platí, že pravděpodobnosti přechodu do následujícího stavu závisejí pouze na současném stavu, ne na předchozích stavech. Tato tzv. Markovovská vlastnost dovoluje proces znázornit stavovým diagramem, kde z každého stavu (uzlu grafu) vycházejí hrany možných přechodů do dalšího stavu s připsanou pravděpodobností. Pro konečný počet prvků lze přechodovou funkci vyjádřit i maticí.

Markovův řetězec je reprezentován posloupností diskrétních náhodných veličin $X_0, X_1, X_2 \ldots X_N$ s hodnotami ze spočetné množiny stavů. Tyto veličiny jsou indexovány množinou čísel $\mathbb{N}_0$. Využijme tedy vztahu 4 pro pozici robotu. Potom tedy x^t označuje pozici robotu v čase t a L_t označuje korespondující náhodnou proměnou. V případě kdy robot nezná svoji skutečnou pozici, nese robot přesvědčení o tom kde by se mohl nacházet. Funkce $Bel(L_t)$ označuje tedy víru v pozici v čase t. Pro příklad, kdy funkce $Bel(L_t = x^t)$ je pravděpodobnost, že robot přiřazuje k možnosti lokalizace v čase se rovná jeho skutečné pozici. Pravděpodobnost pozice robotu je aktualizována v dvou rozdílných typech.

- měření okolí za pomocí senzorické soustavy
- čtení odometrických dat

Markovovi lokalizační odhady posteriorní distribuce jsou provedeny přes všechny náhodné proměnné L_t na dostupných datech. Z čehož tedy dostaneme pravděpodobnostní vztah 7.

$$P(L_T = l|d) = P(L_T|d_0, \ldots, d_T) \tag{7}$$

Před odvozením dalšího přírustku je klíčová myšlenka v Markovově předpokladu. Pokud nyní známe polohu robotu, další měření není ovlivněno touto informací. Jinými slovy je lokalizace založená pouze na posledním stavu v prostředí.

V markovově lokalizačním algoritmu je $P(L_0 = x^0)$, která je inicializována pravděpodobností $Bel(L_0)$ reflektující znalost startovní pozice robotu. Tato distribuce může být inicializována náhodně ve dvou případech.

- pokud je pozice robotu vztažená k mapě neznámá, potom je rozložení pravděpodobnosti jednotné
- pokud je pozice přibližná, potom dosahuje pravděpodobnost Gausova rozložení

(Fox, 1999, 391-427)

3 Popis robotu

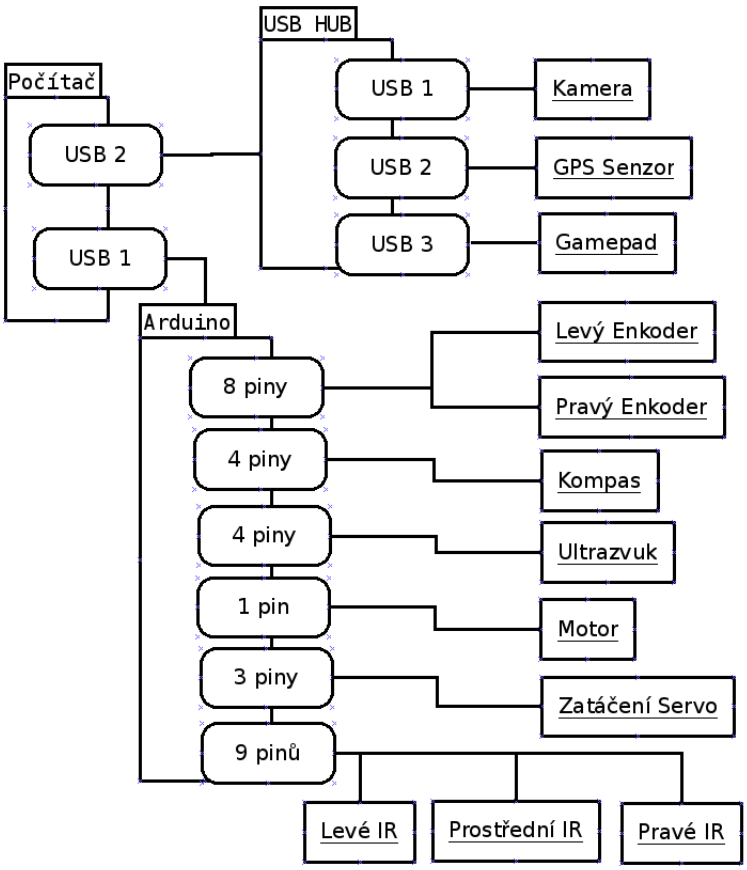
Na obrázku č. 8 vidíme upravený podvozek s názvem Losi 8ighT. Jedná se o přestavbu čtyřkolového podvozku s ackermanovým typem řízení, kde jsou poháněná všechna 4 kola. Od dodavatele se tento prototyp vyrábí pouze se spalovacím motorem, ale na požadavek byl osazen elektromotorem s převodovkou. Další úpravy a nadstavby byly zhotoveny a navrženy v rámci týmu Aistorm, kde se jedná především o uchycení elektrotechnických části, rozmístění a použití komponent na podvozku tak aby k nim byl optimální přístup a neměli vliv na jízdní vlastnosti podvozku. Podvozek je modelem závodního speciálu v poměru 1:8. Celkové rozměry i s provedenými úpravami jsou 495 mm pro délku, 308 mm pro šířku a 600 mm na výšku. Osazená kola mají průměr 325 mm.

Obrázek 8: Úprava podvozku Losi 8ighT

Tento podvozek disponuje pohonem 4x4 a ve vezené vybavě jsou zakomponovány tři infračervené senzory v přední části robota, jeden ultrazvukový senzor, kamera, kompas, GPSka, dvě NiMh baterie, počítač společnosti HP revolve 810 a desku arduino Mega 2050 s obvody umožňující regulaci a převod napětí. Přesné zapojení jednotlivých komponent ilustruje obrázek 9.

3.1 Senzorická soustava

V této podkapitole je popsána použitá senzorická soustava robota, vysvětleny fyzikální principy a konkrétní realizace na robota. Každý senzor je spojen s výpočetní

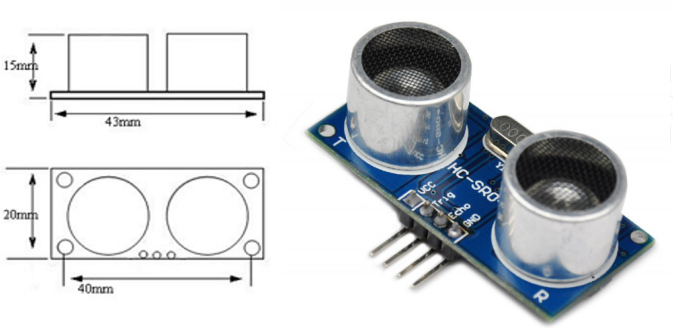


Obrázek 9: Schema zapojení robota

jednotkou, která data trasformuje a posílá pomocí sběrnice. Ve vyšších vrstvách software je informace použita v obslužných třídách a každou časovou periodu aktualizována.

3.1.1 Ultrazvukové senzory

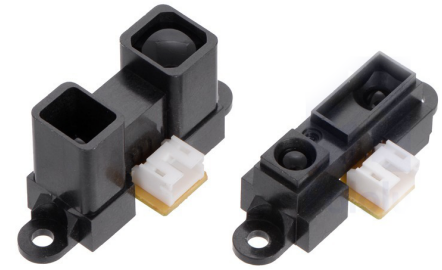
Moduly HC-SR04 (Obrázek 10) jsou kompaktní součástky o velikosti cca 44x20mm, které jsou již samostatně funkční a není již potřeba připojovat další součástky. Připojení do senzorické soustavy je realizováno pomocí výpočetní desky Arduino. Program po vyslání impulsu do modulu počká na zpětné odeslání pulzu od modulu, kdy funkce pulseIn() nám vrátí potřebný počet mikrosekund. S tímto číslem můžeme dále pracovat. Vezmeme v úvahu rychlost zvuku 346,3 $m * s^{-1}$ a to při teplotě suchého vzduchu 25°C. To znamená, že za 1 mikrosekundu urazí v metrech 346,3/1000000 což je 0,0003463 metru. Převedeno na cm to je 0,03463 cm/mikrosekundu. Vzhledem k tomu, že signál jde od čidla k předmětu, kde se odrazí a zase zpět, musíme tuto vzdálenost ještě vydělit číslem 2. Výsledek je. že se vzdálenost bude rovnat počtem mikrosekund násobených číslem 0,017315. Výsledek zašleme na sériový port, kde se každou vteřinu zobrazí.(Měříme vzdálenost, 2012)



Obrázek 10: ultrazvukový senzor

Zdroj: http://imall.iteadstudio.com/media/wysiwyg/Products/IM120628012_HC_SR04/Ultra-Sonic-dimen.JPG

3.1.2 IR- Infračervené senzory



Obrázek 11: použitý infračervený senzor

Zdroj: https://a.pololu-files.com/picture/0J6051.1200.jpg?4567baeed058eeddf63955f8fdf44bde

Použitý senzor SHARP 2Y0A02 F 44 (obrázek 11) má měřící dosah až 150cm. Jeho údaje o vzdálenosti jsou reprezentovány prostřednictvím analogového výstupu u kterého se mění velikost napětí v závislosti na vzdálenosti měřeného objektu.(Engel, 2013, str. 28)

Princip těchto snímačů spočívá v přítomnosti nebo nepřítomnosti světla na přijímači. Tuto změnu způsobí buď přítomnost, nebo nepřítomnost objektu (snímaného předmětu) v ozařovacím poli snímače. Tato změna je zaznamenána na výstupu přepnutím výstupního členu (NPN/PNP nebo relé). Difuzní senzor se skládá z vyhodnocovací a napájecí jednotky, vysílače a přijímače. Minimálně však vždy z vysílače a přijímače, které jsou umístěny do jednoho celku. Způsob, jakým jsou vysílač a přijímač vedle sebe umístěny, závisí na jejich určeném zorném poli dosahu. Výsledná charakteristika je tudíž vždy u každého typu odlišná. (Infračervené snímače v automatizaci, 2001)

Pro použití na robotu je nezbytně nutné hodnotu ze senzoru přečíst a převést ji pomocí A/D převodníku na digitální reprezentaci, která bude následně posílána s dalšími údaji do vyšších vrstev HW a SW vybavení.

3.1.3 Kamery

Osazené webové kamery jsou od společnosti Microsoft. Jedná se o typ LifeCam Cinema. Kamera má automatické ostření, filtry a úpravy expozic. Tento produkt je vhodný v poměru cena/výkon a díky svému válcovému tvaru je snadné kamery přichytit k robotu.

3.1.4 Odometrie

Odometrie je proces, který popisuje transformaci dat poskytnutých enkodéry na změnu pozice a orientace robotu. Enkodéry (obrázek **??**) kol jsou konstrukčně vloženy v zadní nápravě a je využito odečtu magnetického pole pootočením kola. Realizace odečtu signálů (magnetek umístěných v kole) probíhá pomocí relátka připojeného k desce Arduino, kde dochází k transformaci dat společně s kompasem a signály z podvozku do protokolu, který je následně odeslán přes rozhraní USB do vyšších vrstev softwarového vybavení ke zpracování. Samotná sonda je umístěná v obalu který je připevněn z vnější strany na závěs kola. Obal i uložení pro magnety byli zhotoveny za pomocí 3D tiskárny.

3.1.5 Kompas

Modul LSM 303 DLH 3D od společnosti pololu disponuje 3 osým kompasem a akcelerometrem. V použití s projektem BuggyMan je využito pouze kompasu pro určení otočení podvozku oproti globálnímu souřadnému systému. Kompas byl nejprve zkalibrován a následně došlo k opakovanému testování ve vnitřních prostorách budov a na volných prostranstvích. Odečtený směr se v jednotlivých měřeních shodoval s ručním kompasem v řádech stupňů. Statistické vyhodnocení přesnosti přesahuje možnosti této práce. Tento modul je nutné použít s deskou Arduino a předpřipravenými knihovnami pro správnou funkcionalitu.

3.1.6 GPS

Tento modul senzorického vybavení je realizován komerčním senzorem NL – 402U od firmy NaviLock. Disponuje rozhranim USB 1.1 a dokáže přijímat veškeré aktuálně vysílané frekvence z družic ke své lokalizaci. Dokáže příjímat signál GPS a Gallileo s přesností v řádech metrů při ideálních podmínkách. V přítomnosti překážek například budov nebo stromů se přesnost zhoršuje. Modul byl testován v rámci bakalářské práce „Návrh a implementace lokalizačního modulu pro autonomní mobilní robot“. (Koutský, 2013)

4 Metodika

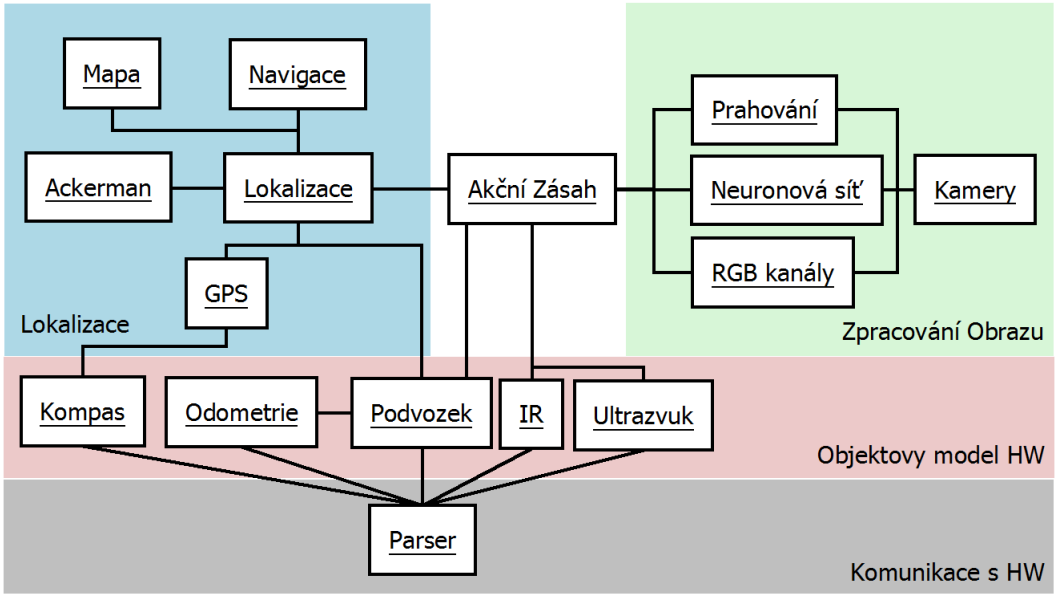
V této kapitole jsou uvedené kroky, které je nutné realizovat, aby byl dosažen stanovený cíl práce.

1. Navrhnutí architektury řídicí jednotky
   - vymodelování jednotlivých vazeb a určení jejich zodpovědnosti
   - navržení způsobu komunikace mezi vrstvami a jednotlivými moduly
   - navržení logovacího systém pro potřeby testování a vývoje funkcionality
2. Vyřešení funkcionality jednotlivých modulů
   - implementace a realizace pořizování dat
   - navržení a implementace detekce cesty
   - navržení a vypracování fůze dat mezi moduly které vyžadují sloučení dat
   - aplikace matematického popisu použitého podvozku pro potřeby lokalizace
   - agregace dat pro přepočet lokalizace
3. Řešení mapování a plánování cesty
   - vytvoření mapy
   - reprezentace mapy a zobrazení
   - plánování cesty
4. Implementace nejvyšší vrstvy řízení
5. Testování řešení

5 Praktická část

Po prostudování řešené problematiky je přistoupeno k návrhu software. Nejdříve je navržená minimální senzorická soustava k úspěšnému splnění cíle. Po návrhu jsou zhotoveny dílčí subprogramy ovládající jednotlivé komponenty. K tomu je použito programovacího jazyku Java a vývojového prostředí Eclipse s rozšířením Maven a AspektJ. Jednotlivé programy jsou spojeny do funkčního celku a obaleny třídami obsluhující jejich funkcionalitu. K nastavení jednotlivých komponent programu je využito konfiguračního souboru. Sběr prvnotních dat je prováděn pomocí teleoperace. Takto nasbíraná data jsou vyhodnocena a použitá k ladění programů. Ověření správného návrhu probíhá v arboretu Mendelovy univerzity.

5.1 Návrh architektury řídicí jednotky



Obrázek 12: Schema modelu systemu

Na obrázku 12 je vyobrazen návrh software a jeho vrstev. Celý návrh vychází z fyzického zapojení robotu, které je znázorněno na obrázku 9.

Program je vystavěn na knihovnách spring frameworku, který poskytuje prostředí pro rychlejší vývoj aplikací. Knihovny samotné využívají základní vývojové prostředí Eclipse a integrují do něj novou funkcionalitu v podobě funkcí pro práci s Subverison systémy, Maven repozitaři, AspektJ projekty a další.

Základní projekt obsahuje pouze zdrojové kódy jednotlivých projektů a pom.xml (Project Object Model) definující nadřazenost kořenového projektu. Právě pomocí POM souboru lze jednotlivé zdrojové kódy programu odvodit do samostatného projektu, který je svázán nadřazeným. Závislost projektů tedy kontroluje cyklické vazby, které jsou nežádoucí. Jednosměrná cesta závislostí vedoucí od

kořene k cíli projektu ukazuje funkcionalitu jiných projektů a vytváří tak možnost lépe strukturovat kód samotný.

Základní strukturou celého programového vybavení jsou třídy vlastní řídicí jednotky. Tyto třídy zavádí program do operační paměti a stanovují parametry pro komunikaci mezi moduly. Čtou konfigurační soubory. Vytváří vlastní pracovní prostor v adresářové struktuře běžícího operačního systému. Snaží se přebírat společnou funkcionalitu všech zapojených podprojektů, tak aby jednotlivé moduly řešili pouze vlastní logiku.

Program je nutné spustit se dvěma přepínači. Přepínač -x značí cestu k místu, kde se bude vytvářet běhová adresářová struktura programu. Tento přepínač uloží cestu do proměnných kořenové třídy *Settings* odkud si ji mohou všechny závislé moduly vyžádat a pracovat s tímto místem. S předpokladem, že se přepínač a ani jeho parametr se moc často měnit nebude, je přepínač opatřen vlastní značkou _{*date*}. Tato značka se automaticky při vytváření adresářů nahradí aktuálním datem a časem s přesností na minuty. Toto označení napomáhá k lepší orientaci v nasbíraných datech. Uvnitř této složky nalezneme dvě složky a tři soubory. První složka je s pořízenými fotografiemi a druhá poskytuje prostor pro uložení důležitých externích konfigurací jednotlivých modulů. Dále je zde soubor *conf.xrc*, který je popsán níže a logové soubory *init.log* shromažďující vypisy před samotným spuštěním hlavní smyčky a *logy.log* s nasbíranými naměřenými daty provedené jízdy. Snímky v adresáři s fotografiemi nesou jednoznačný identifikátor, který je shodný se záznamem v logovacím souboru.

Implementace a realizace sběru dat z robotu je sjednocená v souboru *logy.log* pracovního adresáře. V tomto prostoru jsou logy rozděleny do dvou souborů. První z nich nese informace o inicializaci a zavedení programu samotného a druhý z nich ponese textové informace o nasbíraných datech. Formát jednoho řádku je ve formátu „YYYY-MM-DD hh:mm:ss_[sss] Významnost Logger <-zařízení-> zpráva“. Tento formát zprávy zaručí jedinečnost záznamu napříč všemi posbíranými daty a v případě že by došlo ke kombinaci souborů, nedojde k nepřesnosti zprávy.

Pořizování dat z hardwarové části si bude řešit každý modul samostatně. Do modulu bude pouze dodán odkaz na sběrné místo všech informací a tak bude záležet pouze na zhotoviteli daného modulu co se v logovacím souboru přesně objeví.

Komunikace mezi jednotlivými moduly a vrstvami je zhotovená pomocí systému událostí. Ten je navržen do generického rozhraní, které se dělí na dvě základní části a to producenta a konzumenta. Proto aby třída mohla vysílat nějaké události je zapotřebí konzumenta/posluchače přidat a odebrat. Tuto funkcionalitu zajišťuje třída *IEventProducer*. Vyvolaná událost nese zprávu o změně stavu objektu v datové přepravce vlastního objektu a je předána registrovaným posluchačům implementující rozhraní posluchače *IEventListener*. Tímto způsobem je zajištěno doručení zprávy všem posluchačům události a možnost vlastní implementace reakce na vyvolanou událost.

Druhý přepínač -f vyžaduje cestu ke konfiguračnímu souboru založeného na bázi XML (Extensible Markup Language). V tomto souboru jsou strukturovaně uložené

informace o použitých modulech a komponentách ze kterých se samotný software skládá až za běhu programu. Soubor s příponou XRC (odvozený jazyk na bázi XML) tedy obsahuje definici robotu v párové značce <robot>. Software robotu se skládá z pomyslných zařízení z toho je odvozena párová značka <devices>. V této úrovni zanoření dokumentu můžeme definovat čtyři základní zařízení, které plní účel dle použité značky.

1. **aspect** je zařízení obalující volání všech veřejných metod. Zde můžeme nalézt podobnost s AOP (Aspektově Orientované Programování), kde pointcut metody je připojen okolo metody a zaznamenává její volání a návratovou hodnotu. V logovém souboru se jedná o položky s priority DEBUG a název zařízení je logAcpect.

2. **logger** je zařízení shromažďující správy z celého programu do jediného místa, kde se centrálně formátují a vkládají nejdříve do souboru init.log a následně do logy.log

3. **device** je zařízení nesoucí vlastní logiku, která se vykonává ve vlastních vláknech nebo slouží jako podpůrné třídy pro zpracování vyčtených dat ze senzorické soustavy, nebo transformaci dat z nižších vrstev software. Společným prvkem všech zařízení je možnost upravy chování jednotlivých tříd pomocí nepárové značky <property>, kde atribut name označuje proměnou a hodnota value nebo ref vyjadřuje vkládaný parametr do třídy.

4. **view** je speciální typ zařízení sloužící k vytvoření prezentační vrstvy programu. Každé view dostane ve vytvořeném grafickém okně svůj vlastní prostor pro prezentaci zpracovávaných dat. Pomocí značky <property> dojde k propojení device s příslušným view. View se zaregistruje jako posluchač událostí příslušného zařízení a odebírá vyslané novinky v podobě event systemu.

Takto definovaný software se pomocí XSLT transformace převede do Java Beans reprezentace. V souboru se stejným názvem ale už příponou xml. Ta se následně načte program se sestaví a spustí. Výhodou tohoto přístupu je jednoduchost a definice vlastních klíčových slov, kterých je pro rozsáhlý projekt potřeba málo.

XSL transformace definuje programovací jazyk založený na XML, který slouží k převodu XML dokumentů na jiné textové formáty. V současné době je nejobvyklejší použitím XSLT převod XML dokumentu na jiný XML dokumentů což pomáhá zmírnit následky nekompatibility návrhů XML dokumentů. XSLT nabízí tři odlišné modely programování: model založený na vzorech, procedurální a deklarativní model programování. První a nejjednodušší je model založený na vzorech. Tento model umožňuje vzít šablonu XML dokumentu a naplnit ji XSLT výrazy, které dynamicky naplňují odpovídající místa obsahem. (Skonnard, 2006)

Ukázku konfiguračního souboru a výstup z její transformace nalezneme v příloze nebo na datovém nosiči.

5.2 Řešení vybraných problémů podřízené vrstvy řídicí jednotky

5.2.1 Implementace a realizace sběru dat

Pořizování dat z Hardwarové části si bude řešit každý modul samostatně. Do modulu bude pouze dodán odkaz na sběrné místo všech informací a tak bude záležet pouze na zhotoviteli daného modulu co se v logovacím souboru přesně objeví.

Čtení dat ze zařízení graficky znázorňuje obrázek č. 9 a 12. Zařízení, které jsou připojeny přímo k PC pomocí sběrnice USB bude probíhat formou aktualizace ve vlastních vláknech tak, že se vyčte příslušná hodnota ze zařízení a vlákno samotné se na zadaný interval uspí. Po dobu intervalu budou vyčtená data vnitřně prezentována jako aktuální. Jak je vidět na obrázku č. 9 K samotné realizaci bude použito dvou zařízení, které mezi sebou komunikují. Arduino bude mít v hlavní smyčce programu čtení senzorických údajů na vyžádání dle předem stanoveného komunikačního protokolu. Arduino samotné bude k pc připojené přes USB. V prvé řadě jsou implementovány algoritmy pro sběr dat ze senzorické soustavy a jejich uložení na souborový systém. Dle modelu na obrázku 12 se jedná o první dvě vrstvy.

Komunikační protokol

Tento modul implementuje komunikaci všech implementovaných zařízení, které jsou připojeny k Arduinu. Jak je vidět na obrázku č. 9 Arduino je připojeno k řídicímu počítači pomocí sběrnice USB, ale komunikace samotná je pouze tunelována přes toto medium a vnitřně je implementována pomocí rozhraní COM portu. K tomuto účelu bylo nutné použití knihoven *javax.comm*, které tuto komunikaci již obstarávají na bázi bufferu.

Stávající knihovna byla tedy obalena do třídy *Connection* zajišťující navázání a ukončení spojení se zařízením. Třída přebírá parametry definující pořadí portu a přenosovou rychlost. S těmito parametry se pokusí navázat spojení. Protože se jedná o důležitý most pro přenos informací a příkazů je důležité v případě neúspěchu vše zdokumentovat do logovacího souboru a zastavit celý program. Spojení se tedy navazuje na začátku programu a při jeho ukončení. Stanovený komunikační protokol je dotazovacího typu a přesný význam jednotlivých zpráv je popsán následující tabulkou č. 1. V tabulce jsou uvedeny jen vybrané příkazy. Celý soubor s příkazy a jejich kódy je přiložen na datovém nosiči.

Tato třída slouží pouze jako prostředník mezi fyzickými a softwarovými zařízeními. Každá metoda, která poskytuje aktuální data provolává metodu v mutexu s patřičným kódem zprávy. Po odeslání zprávy je ještě ve stejné metodě implementováno čekání na naplnění bufferu příslušnými daty. Data jsou přenášená po jednotlivých bytech a výsledné číslo se musí zpětně seskládat. K tomuto účelu je využita číselná konverze mezi soustavami, kde se jednotlivé byty převedou do binární reprezentace následně se zřetězí a vypočítá se číslo posílané. Aby bylo zachyceno číslo jakéhokoli rozsahu jsou všechny přečtená čísla datového typu long (v implementaci Javy 1.7 je rozsah $2^{63} - 1$). Protože jednotlivé moduly mohou vyžadovat

Tabulka 1: Tabulka komunikačního protokolu s HW vybavením robotu

DEC	**Příkaz**	**Popis**	**Počet vracených bytů [B]**
16	Pololu Heading	hodnota kompasu	2
40	Ultrasound Front	Vzdálenost v cm	1
59	IR front Left	Vzdálenost v cm	1
59	IR front Mid	Vzdálenost v cm	1
59	IR front Right	Vzdálenost v cm	1
65	Encoder B getData relative	počet tiků enkoderu B	4
74	Reset Enkoder B	vynulování	0
85	Motor Increment	zvýšení rychlosti	0
86	Motor Decrement	snižení rychlosti	0
90	Motor Stop	Zastavení	0
149	Steering Increment	zatoč o jeden krok doleva	0
150	Steering Decrement	zatoč o jeden krok doprava	0
154	Steering reset	srovnání kol	0

pro svojí funkcionalitu i jiný datový typ než long, poskytuje tato třída i sadu metod pro konverzi.

Velkou výhodou tohoto provedení je snadné nahrazení jinou třídou, která implementuje rozhraní jednotlivých tříd používající komunikaci. Nahrazením můžeme docílit snadné záměny za jiný komunikační protokol, jiné přenosové medium, nebo jednoduchého simulačního prostředí podvozku a senzorické soustavy, kde data jsou vyčítána z pořízených logovacích souborů. Každý záznam je směřován do připraveného souboru v adresářovém stromu operačního systému.

Pro komunikaci byly vyzkoušeny přístupy asynchroního volání, kdy se žádost o data provedla a už se nečekalo na výsledek. Ten přišel do systémového bufferu až byl zpracován a vyhodnocen jednotkou Arduino. Použitý přístup musel implementovat rozhraní Javy *SerialPortEventListener*, kam jsou směřovány tyto události. Velký problém nastal v případě rychlejšího plnění bufferu, kde se musela udržovat fronta poslaných příkazů směrem k Arduinu. Tenhle problém vyřeší metoda čtení z bufferu. V synchronizované metodě posílání zprávy k HW se po odeslání provede operace čtení. Čtení v tomto okamžiku uvázne v čekací smyčce dokud se buffer nenaplní požadovanými byty. Synchronizace v Javě 1.7 funguje na principu semaforu a tudíž se všechny požadavky skládají do fronty a jsou zpracovávány v pořadí v jakém na metodu přišli.

Sběr obrazových dat

Modul kamer implementuje zpracování obrazové informace přímo ze zařízení připojeného k řídicí jednotce. Třída implementuje sběr snímků z kamer popsaných v kapitole senzorická soustava. O přenos informace se stará knihovna *com.github.sarxos.webcam*. Do této knihovny se v inicializační fázi předá číslo kamery připojené k operačnímu systému a rozlišení, které je požadováno po kameře. Kamery jsou v OS (Operační Systém) indexovány od nuly. Tato informace je důležitá, protože vezený počítač vlastní vestavěnou kameru a proto připojená kamera má vždy index jedna. Property nastavující index a formát výstupního obrazu otestuje přítomnost požadovaného hardwarového zařízení a schopnost zprostředkovat požadovaný stream. Pokud zařízení je schopno zpracovat takový výstup otevře se proud dat. V opačném případě se vyvolá vyjímka programu. Při plné rychlosti posílají kamery až 30 snímků za vteřinu což není nutné vzhledem k rychlosti pohybu robotu. Pro sběr snímků běží tento modul ve vlastním vlákně, kde v parametru interval je nastavena perioda pořizování snímků z kamer. Tyto snímky jsou logovány do souboru a uloženy na disk. Po záloze snímků je provedená hloubková kopie objektu a vyvolána událost. Kopie jsou distribuovány do všech posluchačů, protože kdybychom předávali původní objekt, tak bychom jsme si následným zpracováním poškodili vstupní snímek který ještě čeká na zápis někde ve frontě operačního systému.

5.2.2 Datové moduly

Modul podvozek je datovým nosičem aktuálního nastavení a vlastností podvozku při zavolání metod se stav aktualizuje nebo v případě závislostí na jiných projektech je stav periodicky aktualizován. Tento modul drží informace o natočení natočení řídicí nápravy a zařazené rychlosti.

Modul Odometrie je stejně jako předchozí pouze nosičem informací o aktuálním stavu enkodérů a ujeté vzdálenosti.

Modul IR (Infra Red)je dynamický modul udržující informaci o všech zapojených IR senzorech na platformě. Tento modul v konfiguračním souboru má zvláštní párovou property, která o proti ostatním má pouze jeden parametr a to název. Uvnitř párové značky jsou schovány další property označující pořadí a kód daného senzoru. Tato property se v xslt transformaci převede na mapu která je modulu nastavena při jeho inicializaci.

Modul Ultrasound vychází ze stejné představy jako modul IR a proto pro něj platí stejná pravidla.

Modul Kompas Reprezentuje zapojené pololu LSM303 a periodicky si vyčítá hodnotu prostřednictvím Arduina a komunikačního modulu. Hodnota vyčtená ze senzoru je šířená pomocí systému posluchačů.

Modul GPS

Modul GPS (Global Positioning System) pracuje na bázi tunelovaného spojení Sériového portu (COM) přes rozhraní USB (Universal Serial Bus). Po spuštění zavádějícího programu je potřeba předat senzoru NL-402U konfigurační soubor s nastavením filtrace snímaných dat a formátu výpisu. Nastavení čísla připojeného portu a periodu s jakou mají být data zaznamenávány do systému.

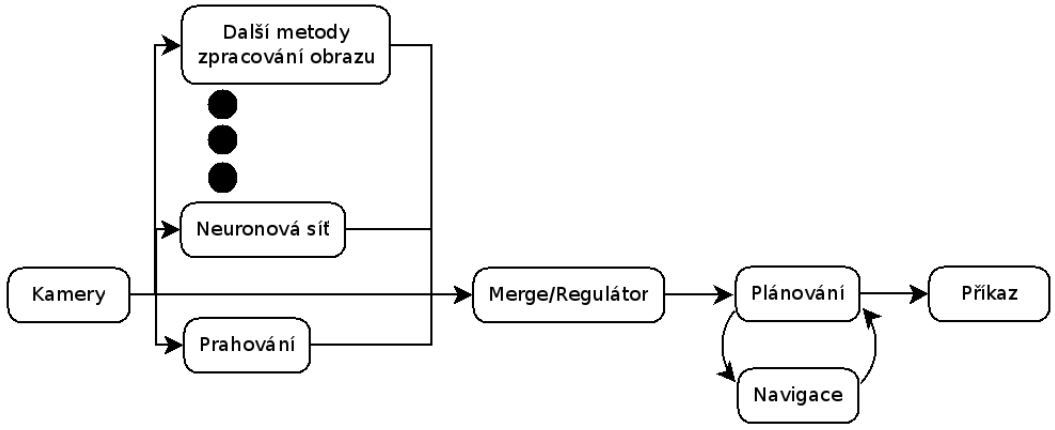
O nastavení konfiguračního souboru pro použitou jednotku pojednává bakalářská práce „Návrh a implementace lokalizačního modulu pro autonomní mobilní robot“ (Koutský, 2013).

Hlavní smyčka tohoto vlákna tedy pracuje s externími knihovnami pro komunikaci na sériové lince. Po přečtení celého řádku dat je programově zjištěna přítomnost klíčových slov. Poté dojde ke zpracování řetězce a převodu naměřené polohy do formátu desetinného čísla. Vytvořená událost je předložená všem posluchačům a zaznamenána do centrálního logovacího souboru.

V průběhu práce a testovacích jízdách arboretem Mendelovy univerzity byli ošetřeny chybové stavy jako je například ztráta signálu jednoho ze systémů GPS, Gallileo nebo obou z nich. V případě ztráty signálu jsou zpracovány pouze dostupné informace a chybová část je zaznamenána do logu.

5.2.3 Detekce cesty v obraze

V této podkapitole bude vysvětlen princip a základní funkcionalita algoritmů, které se podílejí na detekci cesty, rozhodování a zpracování informací nezbytných k průjezdu po cestě. Na obrázku č. 13 je vidět schéma logiky, která má udržet robot na cestě.



Obrázek 13: Schéma principu techniky jizdy po cestě

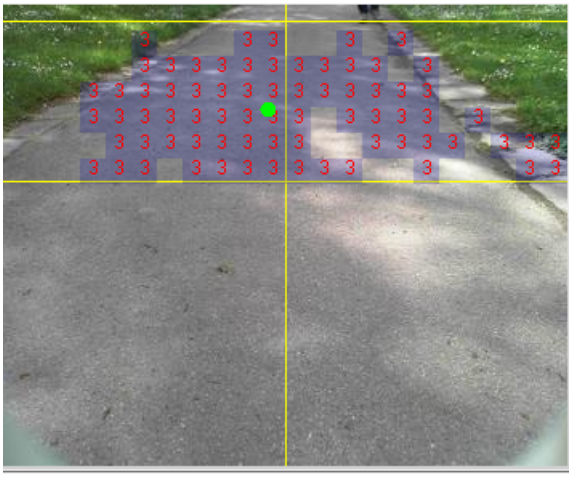
Detekce cesty je realizována na základě real-time zpracování obrazu z vezené kamery. K detekci cesty je použito nízkého rozlišení kamery pro menší výpočetní

nároky. Dále je zpracovávaná pouze ta část obrazové informace, která umožní detekovat střed cesty, tak aby robot měl co největší prostor pro manévrování. Ke zpracování obrazu je použito více metod. Použitím více metod se minimalizuje dopad špatně určené cesty na základě nepříznivých podmínek prostředí ve kterém se robot pohybuje.

Neuronová síť

V tomto modulu je obraz segmentován do matice hodnot, kde na výstupu je v každé buňce uloženo číslo kategorie.

- čistá cesta
- cesta s nečistotami
- okolí cesty
- jiné nerozpoznané okolí.



Obrázek 14: Grafický výstup naučené neuronové sítě

Na obrázku 14 je vidět grafický výstup modulu neuronové sítě. Tento výsput zobrazuje segmenty s nalezenou cestou a na základě nich vypočítává odchýlení středu obrazu od středu cesty.

Pro lokalizaci cesty je použito vícevrstvé neuronové sítě. Jako vstup je použit výřez z nasnímaného obrázku o velikosti 16x16 pixelů a ten je předán k ohodnocení neuronové síti založené na Feed-forward schématu. Toto schéma je založeno na propojení neuronů každý s každým. A proto výsledné výpočty se zjednodušují na práci s maticemi. Vstupní data jsou předány 256 vstupním neuronům. Výsledky jsou zpracovány v dalších dvou vrstvách obsahující 75 a 30 neuronů, které na výstup generují ohodnocení.

Tento modul se při prvním průchodu a inicializaci učí ze vstupních dat. Data jsou modulu předložená pomocí property nastavující cestu ke složce, kde se nachází

přes 40 výřezů správně klasifikovaných dat. K učení je použit přístup učení s učitelem. To je docíleno metodou back-propagation, kdy část dat prohlásíme za trénovací a část dat za testovací. (Hammerschmiedt, 2013)

Tento postup je vhodný použít pak li že hledáme novou konfiguraci neuronové sítě tak aby se co nejlépe přizpůsobovala předkládaným datům. Z hlediska zpětného trasování uvažování programu je důležité použít funkci pro uložení matic s váhami, aby byl výsledek optimálně na zaznamenaných vstupních datech stejný nebo co nejvíce podobný. K tomu nám při zavadění programu slouží property, která definuje soubor s uloženými váhami. Pokud soubor není dostupný, nebo definující property není uvedená v konfiguračním souboru je neuronová síť znovu natrénovaná.

Při práci s tímto modulem bylo vytvořeno několik souborů a složek uchovávající nastavení pro různé prostředí a světelné podmínky.

Pro snížení výpočetní náročnosti implementují moduly zpracovávající obraz rozhraní, které předkládá hladinu a velikost okolí výseče ze zdrojového snímku. Pro snímek o velikosti 352x288 je nutno použít algoritmus klasifikace 396-krát. Díky implementaci rozhraní se výpočetní nároky snížily na 22 dotazů za snímek na modul neuronové sítě při předpokladu že se výseč zúží pouze na jediný řádek. S každým přidaným řádkem roste náročnost lineárně. Výsledek z klasifikovaného výřezu je dále zpracován k nalezení středu cesty. První metoda nalezne střed nejdelší řádkové sekvence v matici pro každý řádek a ten následně zprůměruje. Druhá metoda vychází z představy průměru nalezených bodů v řádku. Jsou li výstupy obou metod pro řádek shodné, změní se v průměr vážený, kde shodným místům je přiřazena větší váha.

Prahování

Detekce cesty je založena na podílu jasové složky v barevném rozložení jednotlivých pixelů. Obraz se tedy projde a vyprahuje dle hodnoty jasu. Dá se předpokládat že obraz bude obsahovat šum. Pro odstranění šumu je použit nejvhodnější klasifikátor nebo filtr. V očištěném obrazu je střed cesty nalezen jako kombinace nejdelšího zpracovaného segmentu a středu jejich extrémů.

Obrázek 15: Postupné zpracování obrazové informace

Zdroj: framework aistorm

Prahování obrázku probíhá až na základě události vyvolaného z modulu obsluhující Kamerové zařízení. Ze samotné události se extrahuje původní snímek, který je zaznamenán v logu a uložen bezpečně na HDD(Hard Disc Drive), vytvoří se jeho kopie a pomocí třídy *BufferedImage* se převede na černobílou reprezentaci. Při vzniku kopie je tedy použit parametrický konstruktor se třemi parametry šířka, výška a konstanta BufferedImage.TYPE_BYTE_BINARY. Výstup barevné konverze je ilustrován na obrázku č. 15 vlevo.

Tato konstanta označí kreslící plátno jako neprůhledné a při vykreslování barevného vstupu na toto plátno si vytáhne nejbližší barvy v paletě barev. Ta je určena třídou *IndexColorModel* a výsledný index je uložen. Přibližování a ztráta alfa nebo barevných složek může mít za následek, v závislosti na barvy v mapě barev třídy *IndexColorModel*.(Oracle BufferedImage, 2014)

Očištění dat od šumu probíhá pomocí klasifikátoru a je vyobrazen jako prostřední snímek na obrázku č. 15. Metody klasifikace založené na vzdálenosti využívají skutečnosti, že pokud jsou dva záznamy klasifikovány do stejné třídy, tak musí mít něco společného, tj. musí si být podobné. Vlastní klasifikace pak spočívá ve zvolení vhodné metriky podobnosti záznamů. Analyzovaný záznam je klasifikován do třídy, která obsahuje mu nejbližší záznamy.

V praxi se používá klasifikace podle k nejbližších sousedů (odtud název k nearest neighbours, KNN). Tato metoda předpokládá, že existují trénovací záznamy, které jsou správně oklasifikované. Vlastní klasifikace nového záznamu pak probíhá tak, že je vybráno k trénovacích záznamů, které jsou nejbližší klasifikovanému záznamu. Analyzovaný záznam je klasifikován do třídy, do které náleží nejvíce z vybraných trénovacích záznamů. KNN používá pro určení blízkosti záznamů vzdálenost vektorů v Euklidovském prostoru (jsou tedy nutné číselné atributy s uspořádáním) dle rovnice 8, kde x a y reprezentují souřadnice bodů. (Rychlý, 2005)

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{8}$$

Cesta na třetím snímku obrázku č. 15 je vypočtena na základě poloviny nalezeného nejdelšího řetězce. V případě obrázku tedy polovina nejdelší bílé sekvence. Tato informace je vložená do view prezentační vrstvy řídicí jednotky dále logována a zpracovávána.

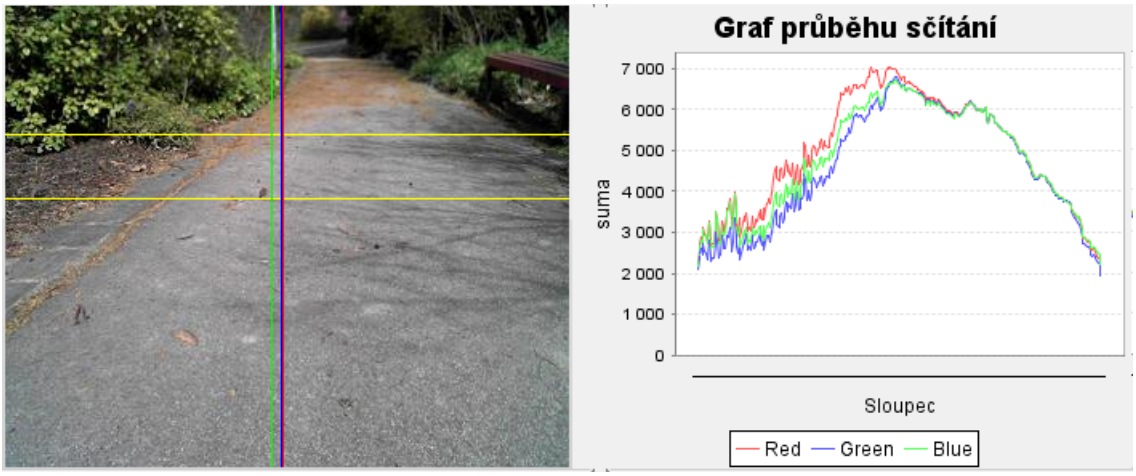
Stejně jako u předešlého modulu (neuronové sítě) se nelze spoléhat na existenci ukázkové cesty. Na cestě se mohou povalovat různé předměty nebo za slunečního svitu se na cestu mohou promítat stíny osob a věcí. Což má za následek nepřesný odhad nalezené cesty. Ke snížení výpočetní náročnosti je i zde implementován interface výřezu obrazu a shoda průměru cesty krajních hodnot.

Barevné kanály

Na obrázku č. 16 je myšlenka lokalizace cesty v obrazu založená na barevném podílu jednotlivých složek ve zpracovávaném pixelu. (Věchet et al., 2007) Obraz je tedy

segmentován do matice pixelů. V této matici jsou sčítány ve sloupcích jednotlivé barevné kanály. Průběh jednotlivých kanálů je vynesen do grafu, který vidíme na pravé části obrázku 16. K uhlazení výsledného signálu je použito plovoucího filtru. Ten v property přebírá šířku prohledávaného okolí. Jako střed cesty tato metoda určuje maximální hodnotu z celého pásma. Pro asfaltový, zámkový a kamenitý povrch je použito modrého kanálu jako výsledku hledání. Tento výsledek je přepočítán na procentuální vyjádření vzdálenosti od středu obrazu a dále posílán a zpracováván dalšími moduly.

I v tomto případě je pro snížení výpočetní náročnosti použit pouze výřez z obrazu.



Obrázek 16: Zpracování barevných kanálů v obraze

Zdroj: framework aistorm

Fůze dat z algoritmů pro detekci cesty

Modul regulátoru podle schématu na obrázku č. 13 je odpovědný za agregaci dat získaných z předchozích modulů zpracovávající obraz. Protože výstup z neuronové sítě, prahování a barevných kanálu je zcela odlišný, je nutné data přepočítat na společnou reprezentaci. Reprezentace detekované cesty je v procentuálním vyjádření vzdálenosti od poloviny obrázku směrem k jeho kraji, kde levý kraj zastupuje hodnotu -100 % a pravý kraj 100 %. Ke sjednocení informací je v tomto modulu implementován synchronizační vzor bariera z důvodů zpracování totožné zprávy algoritmy s rozdílnou časovou a prostorovou náročností. Výsledná událost, kterou vyvolává zpracování těchto dat, nese informace o cestě a jejím středu pro další moduly. Pro to aby mohl tento modul fungovat správně je nutno vytvářet barieru už při vytváření události nového snímku. V implementaci programovacího jazyku Java je použito třídy *OwnBarrier*. Tato třída obaluje cyklickou barieru z dokumentace Javy. Při vytváření je zde nutno uvést na kolik vláken bariera čeká a co se bude dělat v případě

uvolnění bariery. Počet vláken na barieru se určí pomocí tříd registrujících se jako posluchači událostí modulu kamery a současně musejí implementovat rozhraní *IOdchylka*. Po uvolnění bariery se provede kód uvedený ve třídě *Release*. Tato obslužná třída projde všechny vlákna zachycená na barieře a vyžádá si výsledky algoritmů.

Protože výsledná odchylka od středu cesty nemusí záležet pouze na jednom mechanismu detekce je navrženo fůze, která ze všech odchylek vypočítá jedno výsledné číslo. První a nejjednodušší variantou je skládat vypočtené odchylky pomocí váženého aritmetického průměru. Ten zajistí kompenzaci špatně určené cesty na určitý kompromis mezi všemi použitými metodami. Druhý návrh fůze dat bude hledat nejmenší rozdíl mezi jednotlivými dvojicemi odchylek. Pro detekci pomocí neuronové sítě a prahováním je vypočítávána i váha jednotlivých algoritmů. Ta je založená na počtu nalezených bodů/segmetů v obraze. V případě návrhu prahování nebo neuronové sítě to tedy bude procentuální podíl mezi nalezenou cestou a zbytkem obrazu.

Souběžně se zpracováním snímku tento modul integruje možnost zavést umělou chybu v závislosti na změřených datech ze senzorické soustavy infračervených senzorů. V případě že se překážka přiblíží pod nějakou stanovenou mez, robot se jí začne intuitivně vyhýbat. Pak li že je překážka čelně proti robotu. Robot zastaví a počká na uvolnění cesty.

5.3 Implementace nejvyšší vrstvy řídicí jednotky

5.3.1 Matematický model pohybu

Relativní navigace je realizována třídou starající se o výpočty trajektorie robotu tak, aby došlo k nalezení robotu na cestě za pomoci numerického modelu. Pro pohyb robotu a výpočty spojené s relativní navigací je použit modul implementující Ackermanovo řízení, který z předzpracovaných snímků za pomoci kamery, vypočítá přírůstky.

Tento modul tedy slouží jako numerický model pohybu robotu v lokálním kontextu souřadného systému. Modul disponuje funkcemi pro dopřednou úlohu. Ta vychází ze znalosti natočení řídicí nápravy (φ_{SA}) a ujeté vzdálenosti odečtené z enkodéru kol. Změna orientace robotu($\Delta\omega_r$) se spočítá pomocí rovnice 13 a přírustky Δx a Δy podle rovnic 10 a 11. (Věchet et al., 2007)

$$\Delta\omega_r = \frac{\text{ujetá vzdálenost} * tan\varphi_{SA}}{\text{vzdálenost náprav}} \tag{9}$$

$$\Delta x = \frac{\text{vzdálenost náprav}}{tan\varphi_{SA}} * (1 - cos\Delta\omega_r) \tag{10}$$

$$\Delta y = \frac{\text{vzdálenost náprav}}{tan\varphi_{SA}} * sin\Delta\omega_r \tag{11}$$

Tyto vypočítané přírůstky jsou klíčové pro modul lokalizace, který je popsán dále v textu. Modul tedy nepotřebuje běh ve vlastním vlákně, protože je závislý na

informačním toku jiných modulů. Nicméně po každém zavolání výpočtu je vyvolána událost, která je distribuována dále do systému.

Další funkcí implementovanou v rámci tohoto modulu je zpětná úloha, kdy ze znalosti počáteční P_0 a koncové pozice P_1 i s orientací se snažíme odvodit jízdní parametry. Výpočet natočení řídicí nápravy je v rovnici 12 a ujetá vzdálenost je v rovnici 14. Změna orientace ($\Delta\omega_r$) je vypočítána na základě vztahu 13. (Věchet et al., 2007)

$$R = \frac{\Delta x^2 + \Delta y^2}{2\Delta x}, \text{ kde } \Delta x = |P_{0x} - P_{1x}|, \Delta y = |P_{0y} - P_{1y}| \tag{12}$$

$$\Delta\omega_r = arctan\frac{2\Delta x\Delta y}{\Delta y^2 - \Delta x^2} \tag{13}$$

$$\text{ujetá vzdalenost} = \Delta\omega_r R \tag{14}$$

Inverzní přepočty ze známých pozic jsou uplatněny v modulu plánování, kde spolu s modulem navigace se snaží přiblížit k dalšímu bodu v naplánované trase.

5.3.2 Modul Lokalizace robotu

V tomto modulu, kde je implementována vyšší logika lokalizace robotu založená na dvou metodách. Výsledek obou metod je vykreslen na obrázku 17. V levé části je lokalizace vykreslená v doprovodu mapy, aby byl vidět globální kontext. V pravé části je mapa odebrána z důvodů přehlednějšího rozdílu mezi těmito metodami. První z nich (Modrá) využívá poslední přečtené známé pozice z GPS senzoru. Tuto pozici agreguje s aktuálním stavem kompasu a pro vykreslení do mapy je celá událost zanesena jako nová instance třídy Waypoint. Ta je zřetězena do seznamu, který je pomocí dvojtého bufferu vykreslován na výstup.

Obrázek 17: lokalizace pomocí GPS (modrá), lokalizace pomocí odometrie (magenta)

Druhá metoda využívá numerického modelu Ackermanova typu podvozku (Magenta). Tento model plánuje nebo zaznamenává úlohu lokalizace do lokálního

souřadného systému robotu. Pro správnou funkcionalitu tohoto modulu je nezbytné iniciovat v property konfiguračního souboru údaje o startovní pozici.

Modul po nastavení všech parametrů funguje na základě posluchače událostí podvozku. Zde se provádí změna akční veličiny pro nastavení zatočení kol. Tento parametr je klíčový spolu s ujetou vzdáleností pro numerický model. Ten dokáže na základě těchto parametrů vypočítat přírůstky $\Delta x, \Delta y, \Delta\theta$ v metrech a stupních.

Takto spočítaná informace je sama o sobě nevypovídající, protože v ní chybí globální kontext pozice robotu ve známé mapě. Toho je docíleno pomocí transformačních vztahů v rovině pro translaci a rotaci kolem osy X viz rovnice 16 a 15.

$$M_r = \begin{bmatrix} cos\varphi & -sin\varphi & 0 & 0 \\ sin\varphi & cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$M_t = \begin{bmatrix} 1 & 0 & 0 & posuvX \\ 0 & 1 & 0 & posuvY \\ 0 & 0 & 1 & posuvZ \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

Z rovnic jsou patrné neznámé posunu a otočení. PosunX získáme jako přeponu trojúhelníku vypočítanou z přírůstků. Ostatní posunutí jsou nulová. Uhel otočení φ získáme pomocí trigonometrie jako $\Delta x / posuvX$. Získání konečného přírůstku v globálním měřítku je realizováno pomocí násobení matic viz rovnice 17, kde poslední matice značí bod v souřadném systému před posunem. Výsledné přírůstky ve správném souřadném systému jsou v matici M, kde $\Delta x = m_{0,0}$ a $\Delta y = m_{1,0}$.

$$M = M_t * M_r * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{17}$$

Tento postup by se musel pro získání každého konečného bodu v mapě od výchozí pozice opakovat. S každým krokem by bylo nutné připočítat dvě nové matice translace a rotace čímž by se značně zvedla časová a prostorová náročnost. Z toho důvodu je použito přepočtu vztažené k poslední pozici viz rovnice 18 a 19. Souřadnice jsou ve formátu latitude a longitude čímž dojde k přerušení vazby závislé na měřítku mapy a jejím přepočtům v modulu lokalizace. Konstanta 6378137 v rovnicích je vztažená pro systém WGS84 a znamená poloměr matematického tělesa reprezentující zemi v metrech.

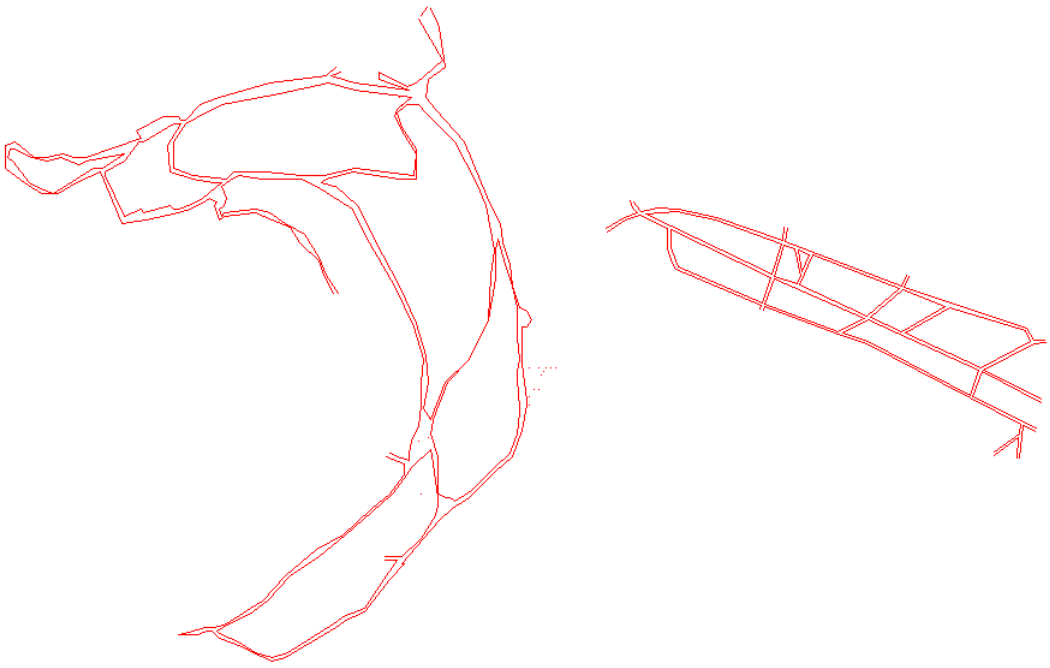
$$lat_{\text{poslední}} = lat_{\text{aktualní}} + (\frac{180}{\pi} * \frac{\Delta x}{6378137}) \tag{18}$$

$$long_{\text{poslední}} = long_{\text{aktualní}} + \frac{\frac{180}{\pi} * \frac{\Delta y}{6378137}}{cos(long)} \tag{19}$$

5.3.3 Modul Navigace

Absolutní navigace je realizována pomocí třídy, která umí zpracovat vstupní soubor ve formátech RNDF, MDF (význam jednotlivých souborů je popsán níže), vytvoří pole bodů kterými robot musí bezpodmínečně projet a tím rozdělí úlohu na dílčí řešené části bez nutnosti hledání optimální cesty ze startovní pozice do cíle. Tato třída poskytuje svému okolí vždy aktuální souřadnice, které ještě nebyli splněny. a jako posluchač událostí vyčkává na události lokalizačního modulu, jestli se robot přiblížil svému cíli. Zároveň poskytuje plánovacímu modulu API(Application Programming Interface), kde pomocí dotazu se zamýšlenými souřadnicemi modul odpoví jestli se plánovaná dráha přibližuje a jak moc výrazně. Pokud se robot přiblíží k bodu pod určitou mez je bod odebrán ze seřazené fronty a všechny výpočty po odebrání směřují k dalšímu bodu. Pak li že fronta bodů je vyčerpána pošle se signál k zastavení a ukončení celého programu.

5.3.4 Modul mapování



Obrázek 18: Mapové podklady testovaných parků

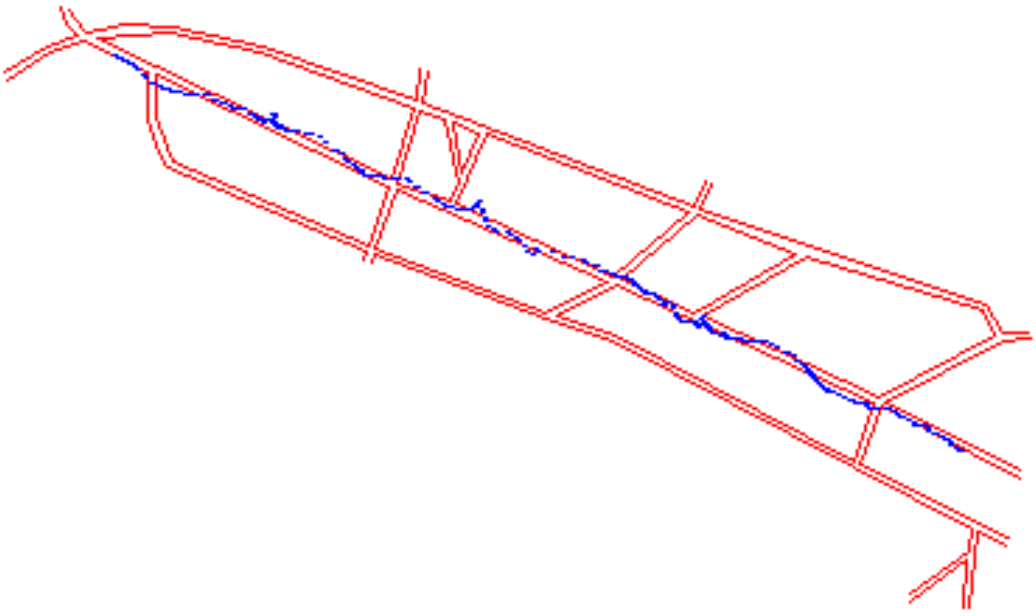
Modul pro práci s mapou je spíše vizualizačním nástrojem. Pro jeho správnou funkcionalitu je třeba mu nastavit property s cestami k mapovému podkladu a definici mise. Mapové podklady testovaných parků můžeme vidět na obrázku 18. V levé části obrázku je vykreslen podklad pro arboretum Mendelovy univerzity a v pravé části je park Palackého sadů v Písku. Tento modul využívá dvojtého bufferu pro

rychlé vykreslování okna na obrazovku a jeho základní velikost je 800 na 600 obrazových bodů. Mapa samotná je načtena ze souboru, kde je uveden seznam úseček a polygonů ve dvojicích desetiných čísel GPS souřadnic. První z nich je Latitude a druhé Longitude. Vykreslování pozice robotu se odehrává na základě poslouchání událostí z modulu obstarávající lokalizaci. Tyto informace jsou agregovány a vykresleny do mapy. Mapu tvoří statická třída *Grafika* a všechny prvky na mapě jsou potomky *Grafického Objektu*.

Do tohoto modulu je přidán panel se zaškrtávacími tlačítky, který udržuje viditelné pouze zaškrtnuté informace. Jako jsou cesty, kontrolní místa, lokalizace na základě GPS a numerického modelu.

5.4 Testování řešení

Cílem testování navrženého řešení je ověření funkcionality software v reálném prostředí městských parků např. arboretum Mendelovy univerzity v Brně. K testování může být užito i jiného parku s podobným charakterem (viz obrázek 19). Dosažené výsledky každého testu jsou uloženy na sdíleném disku robotické skupiny Aistorm. Na základě zpracování jednotlivých měření jsou navržena vylepšení, které zpřesní nebo zrychlí běh robotu. Cíl testování je naplněn s několikanásobným projetím vytyčené cesty autonomní konfigurací.



Obrázek 19: GPS lokalizace v Palackého sadech během soutěže Robotem Rovně

Program je vyvíjen a laděn na stolním počítači HP dc7800 s operačním systémem Windows 7. Protože použitá platforma mobilního robotu neumožňuje vést tak velký počítač ani nedisponuje dostatečným napájením je celé řešení přeneseno na laptop od společnosti HP revolve 810 s operačním systémem Windows 8.1. Tento počítač je použit jako řídicí jednotka robotu a pro sběr dat z podvozku a během ladění programu. Pro prvotní sběr dat je použito manuálního ovládání pomocí USB gamepadu. Pro účely teleoprace je napsán modul manuálního ovládání, ze kterého jsou vyčítaná data posílána ihned ke zpracování podvozku. Každý signál z ovládání je logován a proto soubor shromažďující logy obsahuje i rozhodnutí operátora.

Posbíraná data jsou zpětně zpracovávaná třídou ReadLog. Zde je využito navrženého systému logování do několika úrovní a funkcionality napsané řídicí jednotky. V balíčku, kde je implementován logger, je uložena logika vzniku logu. Což je klíčové pro zpětné zpracování. Logový soubor je řádek po řádku zpracováván a nepodstatné věci jsou opomíjeny. Na každém řádku logu je uvedena časová značka, zařízení, a zpráva. Třída ReadLog využije názvu zařízení a rozdělí řádek právě touhle značkou. Dále rozdělí pravou část dle mezer a patřičné informace z logu vypreparuje do připravených seznamů.

Takto připravené seznamy čekají na „dummy“ implementace zařízení, které jsou potomky jejich ostrých verzí a díky dědičnosti jsou překryty pouze metody klíčové pro sběr dat. Tímto způsobem je docíleno, že navržené algoritmy pracují v offline režimu a celý výstup z programu lze ladit pohodlně.

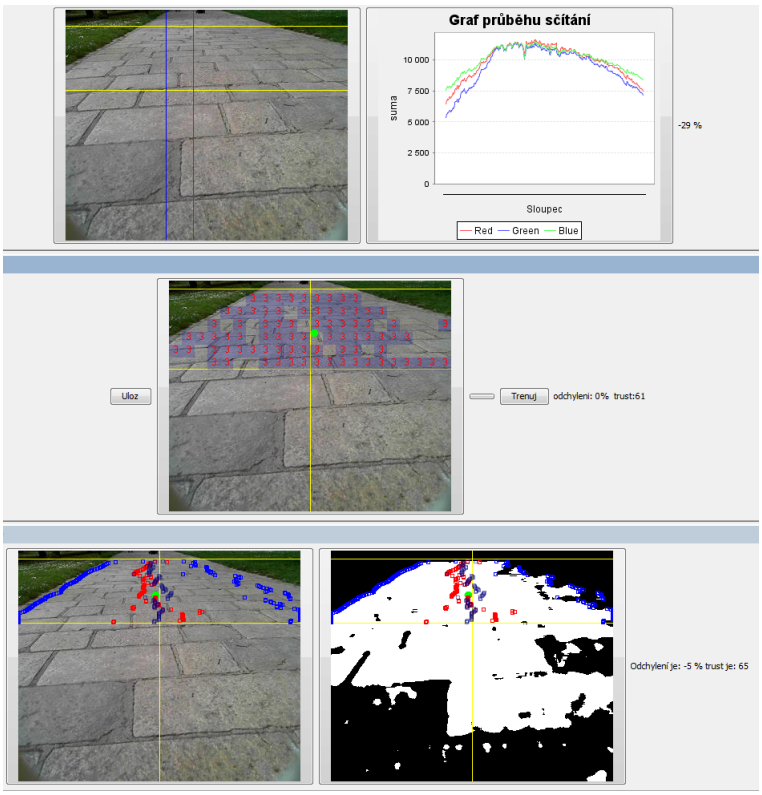
Obrázek č. 20 ukazuje grafický výstup algortimů pro detekci cesty. Zde si můžeme povšimnout výseče zpracovávaného obrazu. Na základě vypočítané vzdálenosti jednotlivých algoritmů je potom určen akční zásah pro podvozek, aby se robot udržel na cestě.

Po vyladění programu na testovacích datech je přistoupeno k autonomnímu testování. Na základě znalostí získaných z dat je připraven konfigurační soubor obsahující nezbytné konstanty určující chování celého systému. Testování autonomní jízdy probíhalo spuštěním programu a vyzkoušení reakce podvozku na snímanou cestu. Pak následovalo zařazení prvního rychlostního stupně.

Pokud robot sjel z nějaké příčiny ze sledované cesty byl program zastaven čímž došlo k oddělení běhového prostředí a následnému sběru dat do nové složky.

Během testování se ukázalo že automatický filtr použitých kamer je nedostačující a v obraze se objevují za slunečného počasí ostrá místa na kterých robot nedokáže správně detekovat cestu. Z toho důvodu byl zkonstruován přípravek, který před objektiv kamery přichytí polarizační filtr. Tento přípravek je zhotoven pomocí technologie 3D tisku a je obarven matnou černou barvou aby nedocházelo k průsvitu přípravku a odrazům z vnitřního prostředí polarizačního filtru.

Během testování byla experimentálně upravována hladina výřezu snímku, tak aby robot dostatečně vnímal své okolí a na širokých cestách byly v záběru i okraje cest. Testování samotné začínalo vždy na hranici křižovatky v různých částech arboreta, tak aby robot měl před sebou cestu bez křižovatek, které jsme se snažili projet rovně. Řízení založené na snímcích z kamer vykazovalo největší spolehlivost



Obrázek 20: Grafický výstup modulů pro detekci cesty

na otevřeném prostranství za jakéhokoli počasí. Největší problémy při autonomní jízdě způsobovaly místa pod stromy, kde pronikali ostré paprsky slunce. Díky stínění korun okolních stromů zde vykazovala i značnou nepřesnost vezená GPSka.

Při testování modulu lokalizace na základě výpočtů z odometrie se objevuje chyba v ujeté vzdálenosti. Z tohoto důvodu jsou enkodéry po dosažení nastavené hranice periodicky resetovány do nulového stavu, aby přírůstky chyby byli co možná nejmenší. Skutečnost táhnutí podvozku k jedné straně je kompenzována v řízení založeném na zpracování obrazu. V lokalizaci za pomocí odometrie je tato systémová chyba nevyhodnocená což má za následek odchýlení robotu od cesty.

Zde v testování se nejvíce uplatnil konfigurační soubor na bázi XML, který zajistil příslušnou škálovatelnost programu a rychlé nastavení v případě potřeby. Zejména nastavení příslušných cest k datovým souborům, které byli v každém OS (Operačním Systému) uloženy na rozdílných místech.

6 Zhodnocení

6.1 Shrnutí

Práce se zabývá problematikou vývoje software pro platformu mobilního kolového robotu. Pro implementaci aplikace byl zvolen programovací jazyk Java s využitím jeho rozšíření subversion, Maven, AspektJ, Swing a další. Naprogramovaná aplikace je odladěná pro prostředí Windows 7 a 8.1, kde je nainstalována 32-bitová Java.

V práci je použito komerčně dostupných komponent kompasu, kamery, GPS lokátoru, podvozku a dílů navržených, nebo upravených pro tuto práci.

V oddílu se současným stavem řešené problematiky je čtenář uveden do problematiky lokalizace robotu a současně jsou zde představeny podobné robotické jednotky. Na základě popisu jednotlivých projektů je navrženo vlastní schéma aplikace, které je uplatněno v praktické části řešení. Dále autor popisuje použitou senzorickou soustavu a jejich fyzikální principy a vlastnosti. Z tohoto přehledu je vytvořen objektový model pro nižší vrstvy aplikace modelující příslušné komponenty.

V základu praktické části autor popisuje výstavbu software na konfiguračním souboru založeném na bázi XML. Nastavení jednotlivých částí programu je popsáno v kapitole zabývající se návrhem architektury a následně je řešená funkcionalita jednotlivých modulů a vrstev aplikace.

V další části textu jsou popsány vyřešené problémy s komunikací mezi jednotlivými moduly. Zde je navrženo systému událostí a reakcí na ně. S tímto modelem souvisí zpracování implementace a sběru dat ze senzorické soustavy robotu.

Pro autonomní jízdu robotu autor uvádí metody zpracování obrazové informace, které byli použity pro detekci cesty. V textu je navržena fůze dat z jednotlivých metod tak, aby se robot snažil udržet na nalezené cestě. Tento akční zásah do regulované soustavy pohonů robotu je zaznamenán a použit v modulech určených pro lokalizaci.

K lokalizaci robotu je využito matematického modelu podvozku a několika souřadných systémů k výpočtu pozice za pomocí odometrie a senzorického čtení dat. Orientace robotu v mapě je vyřešená pomocí datového souboru určující kontrolní body, kterými je nutné projet. Po odebrání všech kontrolních bodů je trasa považována za projetou.

V poslední části práce je uvedeno testování vyhotoveného řešení. Celá aplikace byla testována v arboretu Mendelovy univerzity a při soutěžních jízdách v parku Paladského sadů v Písku. Veškerá zaznamenaná data i se zdrojovými kódy jednotlivých modulů autor přikládá k této práci na datový nosič.

6.2 Diskuse

V práci je předložen vícevrstvá modulární architektura řídicí jednotky mobilního robotu, která je schopná zajistit autonomní pohyb kolového robotu po zpevněných cestách městských parků. Celkový koncept programu umožňuje rychlou implemen-

taci nových algoritmů lokalizace, plánování či mapování prostou záměnou použitého modulu za nově vytvořený. Z autonomního, v práci popsaného, robotu se tak stává univerzální platforma umožňující snadný vývoj algoritmů souvisejících s činností mobilních robotů. Tento fakt považuje autor práce za jeden z hlavních přínosů.

Za přínos lze také považovat vlastní technické řešení vytvořeného programu, který je implementován jako vícevláknový. Tento přístup dopomáhá optimálnímu rozložení výpočetní zátěže na všechna jádra řídicí jednotky. Další výhodu představuje celistvost představeného řešení, které je tvořeno moduly Maven. Tímto odpadá nutnost spouštění jednotlivých podprogramů a řešení dodatečné komunikace mezi moduly. Naopak prostor pro zlepšení vidí autor práce v návrhu grafického rozhraní programu. Aktuální rozložení prvků zobrazuje jednotlivé moduly neuceleně ve vlastních podoknech. Ta jsou navíc optimalizována spíše pro vyšší rozlišení pracovních stanic používaných během vývoje programu s odpojeným robotem. Možné vylepšení tedy spočívá v návrhu grafického uživatelského rozhraní, které by informovalo operátora o stavu robotu na monitoru připojeném k řídicí jednotce během autonomní jízdy ve venkovním prostředí.

6.3 Závěr

Cílem této diplomové práce bylo navrhnout a implementovat řídicí software kolového mobilního venkovního robotu. Tento cíl byl úspěšně splněn. Navržený software zajišťující autonomní pohyb robotu po parku je považován za hlavní výstup práce. Správnost návrhu vícevrstvé architektury řízení a funkčnost jednotlivých modulů prokazují desítky provedených testů v arboretu Mendlovi univerzity a také úspěšná účast na robotické soutěži Robotem rovně v květnu 2015. Autor práce počítá s dalším rozšiřováním návrhu řídicí jednotky v rámci doktorského studia.

7 Reference

ARBot [online]. 2008 [cit. 2015-01-25]. Dostupné z: http://www.arbot.cz/post/2013/10/14/ARBot-hardware-en.aspx .

Bayes Nets [online]. 2007 [cit. 03.05.2015]. Dostupné z: http://www.bayesnets.com/.

BILGIN, Esme. *Kalman Filter for Dummies* [online]. 2009 [cit. 2014-11-29]. Dostupné z: http://bilgin.esme.org/BitsBytes/KalmanFilterforDummies.aspx.

DIEZ, David, Christopher BARR a Mine CETINKAYA-RUNDEL. OpenIntro statistics [online]. 2012, 426 s. [cit. 10.02.2015]. Second Edition. Dostupné z: https://drive.google.com/file/d/0B-DHaDEbiOGkZ1o5VVYwYXVKVjg/view.

DURRANT-WHYTE, Hugh a Tim BAILEY. *Simultaneous Localisation and Mapping (SLAM): Part i The Essential Algorithms* [online]. 2006. Dostupné z: http://www.cs.berkeley.edu/p̃abbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf .

ENGEL, Michal. 2013. VYUŽITÍ SENZORŮ PRO MĚŘENÍ VZDÁLENOSTI JAKO ČIDLA PRO POČÍTÁNÍ OSOB [online]. Brno. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=69249.

FOX, Dieter, Wolfram BURGARD a Sebastian THRUN. 1999. Markov localization for mobile robots in dynamic environments [online]. [cit. 2015-05-10]. Dostupné z: https://www.jair.org/media/616/live-616-1819-jair.pdf.

HAMMERSCHMIEDT. Analýza obrazu pro potřeby řízení mobilního robota. Brno, 2013. Diplomová. Mendelova univerzita v Brně. Vedoucí práce Radovan Kukla.

HERMAN David, Tomas ONDRACEK a Filip ORSAG. Quido - the robot that won the Robotour 2011 competition [online]. 2011 [cit. 2015-04-12]. Dostupné z: http://www.fit.vutbr.cz/ ihajek/other/robosense-posters/poster_abstract-roboauto.pdf.

Infračervené snímače v automatizaci. 2001. MM Průmyslové spektrum. Praha: Vogel Publishing, 2001(3, 05.03.2001). ISSN 1212-2572. Dostupné také z: http://www.mmspektrum.com/clanek/infracervene-snimace-v-automatizaci.html.

KOUTSKÝ, Filip. Návrh a implementace lokalizačního modulu pro autonomní mobilní robot. Brno, 2013. Bakalářská. Mendelova univerzita v Brně. Vedoucí práce ing. Vít Ondroušek Ph. D. .

MARCHETTI, Luca, Giorgio GRISETTI a Luca IOCCHI. DIPARTIMEN TO DI INFORMATICA E SISTEMISTICA UNIVERSITA A "LA SAPIENZA". A Comparativ e Analysis of P article Filter based Lo calization Metho

ds [online]. 2012 [cit. 03.05.2015]. Dostupné z: http://www2.informatik.uni-freiburg.de/g̃risetti/pdf/marchetti06robocup.pdf .

Měříme vzdálenost s HC-SR04 [online]. 2012 [cit. 2015-01-26]. Více zde: http://arduino8.webnode.cz/news/lekce-9-merime-vzdalenost-s-hc-sr04/ .

NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN - technická literatura, 2007, 248 s. ISBN 80-730-0141-1..

ORACLE. Class BufferedImage [online]. 2014 [cit. 2015-02-20]. Dostupné z: http://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html #TYPE_BYTE_BINARY.

PAVLENKO, Peter. Uživatelské rozhraní pro řízení pozemního robota ve venkovním prostředí. Brno, 2014. Dostupné z: http://www.stud.fit.vutbr.cz/ xpavle00/technicka_sprava.pdf. Bakalářská. Vysoké učení technické. Vedoucí práce Beran Vítězslav..

Particle filter. Wikipedie [online]. 2014, 2014-12-15 [cit. 2015-01-29]. Dostupné z: http://en.wikipedia.org/wiki/Particle_filter.

Recursive Bayesian estimation. Wikipedie [online]. 2015, 2015-1-6 [cit. 2015-01-29]. Dostupné z: http://en.wikipedia.org/wiki/Recursive_Bayesian_estimation.

RIISGAARD, Søren a Morten Rufus BLAS. *SLAM for Dummies* [online]. 2005, 127 s. [cit. 25-11-2014]. Dostupné z: http://ocw.mit.edu/courses/aeronautics-and-astronautics/16412jcognitive-roboticsspring2005/projects/1aslam_blas_repo.pdf .

RICHTER, Miloslav. *Kalmanův filtr* [online]. 2008, 2008-10-17 [cit. 2014-11-26]. Dostupné z: http://www.uamt.feec.vutbr.cz/řichter/vyuka/0910_mpov/tmp/kalman_filter.html.cs.

ROBOAUTO A. S. Roboauto [online]. 2012 [cit. 2015-04-12]. Dostupné z: http://www.roboauto.cz/foswiki/bin/view/Projekt/WebHome?redirectedfrom=Main.WebHome.

RYCHLÝ, Marek. FIT VUT. Klasifikace a predikce [online]. 2005 [cit. 2015-02-20]. Dostupné z: http://www.fit.vutbr.cz/ rychly/public/docs/classification-and-prediction/xhtml/classification-and-prediction.xhtml.

SENA ESTEVES, João, Adriano CARVALHO a Carlos COUTO. UNIVERSITY OF MINHO. Generalized Geometric Triangulation Algorithm for Mobile Robot Absolute Self-Localization [online]. 2008 [cit. 21.04.2015]. Dostupné z: www.researchgate.net/profile/Carlos_Couto4/publication/4056103_Generalized_geometric_triangulation_algorithm_for_mobile_robot_absolute_self-localization/links/00b4952286628b4da1000000.pdf .

Simultaneous localization and mapping. Wikipedie [online]. 2014, 21 November 2014 [cit. 2014-11-25]. Dostupné z: http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping.

SKONNARD, Aaron a Martin GUDGIN. XML: pohotová referenční příručka : referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším. 1. vyd. Praha: Grada, 2006, 342 s. Průvodce (Grada). ISBN 80-247-0972-4. .

SMEKAL, Zdeněk a Petr SYSEL. Číslicové filtry [online]. 2004, 130 s. [cit. 18.04.2015]. Dostupné z: http://www.researchgate.net/profile/Robert_Vich/publication/47091716_slicov_filtry_/links/0c96052efc299c7e66000000.pdf.

STANDFORD UNIVERSITY. Junior: The Stanford Entry in the Urban Challenge [online]. 2005 [cit. 2015-04-12]. Dostupné z: http://robots.stanford.edu/papers/junior08.pdf.

ŠVEHLA, Ondřej. Realizace podpůrného mapového software pro autonomní mobilní robot [online]. Brno, 2013 [cit. 2015-02-18]. Dostupné z: https://is.mendelu.cz/auth/lide/clovek.pl?zalozka=7;id=33158;studium=50398;zp=41242;download_prace=1. Bakalářská. Mendelova univerzita v Brně. .

REKLEITIS, Ioannis. MCGILL UNIVERSITY. A Particle Filter Tutorial for Mobile Robot Localization [online]. 2008 [cit. 05.05.2015]. Dostupné z: http://www.cim.mcgill.ca/ÿiannis/particletutorial.pdf .

VĚCHET, S.; KREJSA, J.; ONDROUŠEK, V.; The Development of Autonomous Racing Robot Bender, článek v časopise Engineering Mechanics, Brno, Engineering Academy of the Czech Republic, Vol.14, No.4, 2007, pp. 277-287, ISSN: 1802-1484 .

Výzkumná skupina robotiky Robo@FIT: Toad [online]. 2014 [cit. 2015-01-25]. Dostupné z: http://www.fit.vutbr.cz/research/groups/robo/eqp.php.cs .

Seznam obrázků

8 Přílohy

8.1 Příklad konfiguračního souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<robot>
 <devices>
 <aspect name="logAspect"
    class="cz.mendelu.aistorm.log.AspectLog">
    <property name="logger" ref="robotLogger" />
  </aspect>
  <logger
   name="robotLogger"
    class="cz.mendelu.aistorm.log.RobotLogger"
    path="PahtToInitLog">
  </logger>
  <device name="Kamery"
    class="cz.mendelu.aistorm.device.Kamery">
         <property name="leva" value="1" />
         <property name="formatIndex" value="4" />
         <property name="interval" value="1000" />
         <property name="logger" ref="robotLogger" />
   </device>
   <device name="Prahovani"
         class="cz.mendelu.aistorm.prahovani.
PrahovaniDeviceRefactor">
         <property name="logger" ref="robotLogger" />
         <property name="heightOfLine" value="100" />
         <property name="heightFromLine" value="20" />
         <property name="kamery" ref="Kamery" />
         <property name="neighbors" value="5" />
   </device>
   <view name="PrahovaniView"
         class="cz.mendelu.aistorm.prahovani.view.PrahovaniView">
         <property name="prah" ref="Kamery"/>
         <property name="adevice" ref="Prahovani" />
         <property name="logger" ref="robotLogger" />
   </view>
   <device name="parser"
         class="cz.mendelu.aistorm.deviceParser.
SkritecekSynchroni">
         <property name="interval" value="1000" />
```

```
        <property name="logger" ref="robotLogger" />
        <property name="comPort" value="COM3" />
    </device>
    <device name="ackerman"
        class="cz.mendelu.aistorm.navigation.driver.
NavigationAckerman">
        <property name="interval" value="1000"/>
        <property name="logger" ref="robotLogger" />
    </device>
    <device name="Podvozek"
        class="cz.mendelu.aistorm.devicePodvozek.Podvozek">
        <property name="logger" ref="robotLogger" />
     <property name="wheelBase" value="36" />
     <property name="skritecek" ref="parser"/>
    </device>
    <device name="lokalizace"
      class="cz.mendelu.aistorm.lokalizace.Lokalizace">
     <property name="logger" ref="robotLogger" />
     <property name="interval" value="1000" />
     <property name="ackerman" ref="ackerman" />
     <property name="podvozek" ref="Podvozek" />
     <property name="latStart" value="49.211672" />
     <property name="longStart" value="16.614476" />
     <property name="omegaStart" value="60" />
  </device>
  <device name="Regulator"
    class="cz.mendelu.aistorm.regulator.AkcniZasah">
    <property name="logger" ref="robotLogger" />
    <property name="kamery" ref="Kamery"/>
    <property name="chassy" ref="Podvozek"/>
    <property name="ultrasound" ref="Ultrazvuky"/>
    <property name="IR" ref="IRka"/>
    <property name="emergencyIrPravy" value="50" />
    <property name="emergencyIrLevy" value="50" />
  <property name="emergencyUltrasound" value="50"/>
 </device>
 </devices>
</robot>
```

8.2 Datové CD

Na přiloženém datovém nosiči jsou obsaženy následující položky:

- zdrojové kódy aplikace
- elektronická podoba této práce
- použité obrázky
- posbírané logy z testování
- videoukázka funkčního projektu