TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autonomní robot pro mapování vnitřních prostor

Autonomous robot for mapping of interiors

Bakalářská práce

Autor: **Zbyšek Zapadlík**

Vedoucí práce: Doc. Ing. Mgr. Václav Záda, CSc.

Konzultant: RNDr. Miroslav Kulich, Ph.D.

V Liberci 15. 5. 2013

(místo tohoto listu se zde nachází zadání práce)

Prohlášení

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

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

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

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

Datum

Podpis

PODĚKOVÁNÍ:

Tímto děkuji panu Doc. Ing. Mgr. Václavu Zádovi, CSc. za vedení mé práce, poskytnutí zázemí a za velký prostor, jenž mi byl umožněn.

ABSTRAKT:

Cílem této práce je zvolit systém pro mapování a lokalizaci mobilního robota v neznámém prostředí a prakticky ho vyzkoušet na reálné platformě. Předpokládá se užití laserového měřícího senzoru a robotické platformy z předchozího projektu [1]. Takto komplexní úloha vyžaduje úpravu řídícího softwaru mobilního robota, vytvoření funkcí pro získání dat z laserového měřícího senzoru a konečně výběr a aplikaci vhodných algoritmů pro tvoření mapy okolí a lokalizaci platformy v ní. Závěrem se zhodnotí kvalita dosažených výsledků a navrhne se možné vylepšení.

KLÍČOVÁ SLOVA:

Lokalizace; mapování; mobilní robot; SLAM; Simultánní lokalizace a mapování; Laserový skener; Laserový senzor; skenování; LabView; MatLab;

ABSTRACT:

The aim of this work is to choose a system for mapping and localization of mobile robot in an unknown environment and apply it on a real platform. It is expected to use laser measurement sensor and robotic platforms from a previous project [1]. Such complex task requires a modification of the control software of mobile robot, a function for extracting data from the laser measurement sensor and finally selecting and applying appropriate algorithms for creating maps and localization of the platform in it. Finally, evaluate the achieved results and propose possible improvements.

KEYWORDS:

Localization ; mapping; mobile robot; SLAM; Simultaneous localization and mapping; Laser scaner; Laser senzor; skenování; LabView; MatLab;

OBSAH

ÚVOD

Potřebná vlastnost mobilního robota je určit svojí pozici v okolním prostředí s takovou přesností, aby byl schopen vykonávat činnost, pro kterou je určen. Pokud je okolní prostředí neznámé, robot musí simultánně tvořit mapu a lokalizovat v ní svou pozici. Takovýto proces se označuje jako „SLAM“, což je zkratka anglického výrazu „Simultaneous Localization and Mapping“.

Tato práce zčásti vychází ze „SLAM kurzu“ [3], určeného pro základní seznámení s EKF (Extended Kalman Filter) a SLAM, který je každoročně přednášen na univerzitě Toulouse ve Francii. Součástí tohoto kurzu je série videí o celkové délce 8,5 hodin a písemný dokument popisující problematiku SLAM a teoretické pozadí EKF [4]. Další součástí kurzu jsou kódy pro MATLAB, které byly v rámci této práce dále upraveny.

Byly zvoleny následující prostředky pro daný úkol:

Hardware – byl použit laserový měřící senzor, který je popsán v kapitole 1. Mobilní robot je použit z předchozího bakalářského projektu [1] a upraven pro aplikaci s výše zmíněným senzorem. Celý hardware je stručně popsán v kapitole 2.

Software – pro řízení mobilního robota byl v rámce práce vytvořen program v prostředí LabView. Kódy samotného algoritmu SLAM v jazyce prostředí MATLAB jsou použity z výše popsaného kurzu, bylo ale třeba doplnit několik funkcí pro reálné použití a propojení s řídícím programem robota. Podrobný popis se nachází v kapitole 4.

1. LASEROVÝ MĚŘÍCÍ SENZOR

Následující údaje byly čerpány z anglického manuálu k danému výrobku [5] a popisu protokolu komunikace [6].

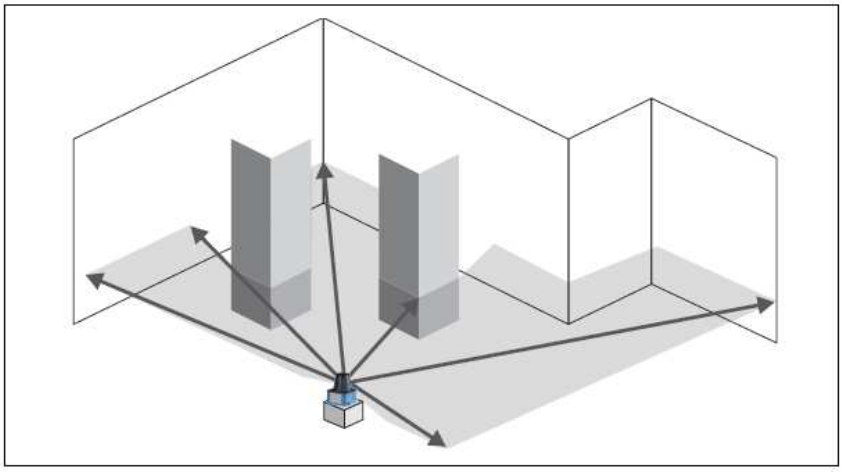
1.1 ZÁKLADNÍ VLASTNOSTI

TiM310 je optoelektrický laserový skener, který snímá svůj přilehlý perimetr v jedné rovině za pomoci laserových paprsků. Výstupem takovéhoto senzoru jsou data v polárních souřadnicích, udávající vzdálenosti k objektům v jeho okolí, jak znázorňuje Obr. 2. Rozsah snímání je 270° a dosah 4 m při denním světle a při odrazu od přirozeného povrchu, který má reflektivitu větší jak 50% (např.: bílá stěna domu). Rozlišení je jedna hodnota vzdálenosti na jeden stupeň, tzn. 271 hodnot.

Obr. 1 Optoelektrický laserový skener, převzato z [5]

TiM310 používá technologii HDDM (High Definition Distance Measurement), což je technologie, kterou vlastní firma SICK a je založená na měření doby letu vyslaného paprsku. Paprsek putuje od senzoru k objektu, kde se odrazí od jeho povrchu a vrátí se zpět k senzoru. Tento pulzující laserový paprsek je rozmítaný do roviny pomocí rotujícího zrcátka spojeného s enkodérem (snímač natočení). Na každou hodnotu, tedy na 1°, připadá 84 podružných měření, které jsou průměrovány.

Senzor dokáže snímat své okolí s frekvencí 15 Hz.



Obr. 2 Znázornění snímání okolí v jedné rovině, převzato z [5]

1.2 FORMÁT KOMUNIKACE

Senzor je opatřen kabelem s elektrickými vodiči napevno připevněným k vlastnímu tělu. Tento kabel obsahuje napájecí vodiče (10 - 28 V DC) a také signálové vodiče – několik vstupů a výstupů, jejichž funkci lze určit v nastavovacím softwaru.

Dále senzor disponuje konektorem pro připojení USB kabelu. V našem případě bylo použito právě tohoto rozhraní pro načítání dat.

Načítání dat ze senzoru může probíhat pouze v ASCII formátu dvěma způsoby:

1) **PC se dotáže na jeden telegram,**

   a to následujícím řetězcem:

   <STX>sRN{SPC}LMDscandata<ETX>

2) **PC pošle příkaz, aby senzor posílal data permanentně.** Pro zrušení je třeba poslat příkaz o zastavení posílání.

   Realizováno za pomoci následujícího řetězce:

   <STX>sEA{SPC}LMDscandata{SPC}1<ETX>

Pro zastavení je příkaz identický, až na předposlední znak – nula místo jedničky.

Startovací znak „<STX>“ odpovídá hodnotě 02 hexadecimálně, tzv. mezera „{SPC}“ hodnotě 20 hexadecimálně a ukončovací znak „<ETX>“ hodnotě 03 hexadecimálně.

Následně senzor pošle do PC sérii dat v normalizované textové podobě, ze kterých se následně vyextrahují ta potřebná. Každý údaj je oddělen mezerou, a proto byl v této práci pro extrakci využit jednoduchý postup – od 27. údaje začínají data *skenu* a těchto dat je 270 – takže pouze tyto informace byly převedeny z hexadecimálního tvaru do číselné podoby.

1.3 KOMUNIKAČNÍ ROZHRANÍ

Pro spojení senzoru a nadřazeného řídicího systému, tedy PC, bylo využito rozhraní USB 2.0 s následujícími parametry pro komunikační kanály, tzv. *pipes* (uvedené hodnoty jsou v hexadecimálním tvaru):

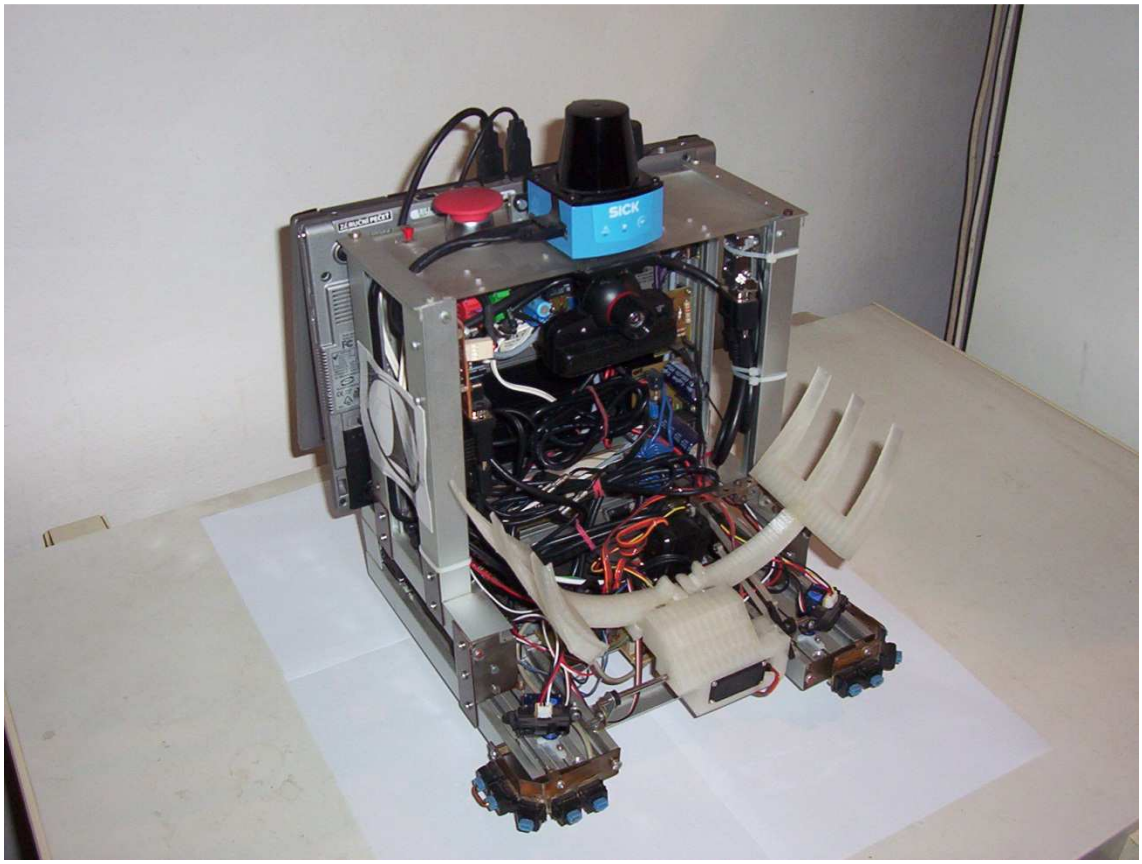
Control pipe: 0 (default)

Bulk in pipe: 81

Bulk out pipe: 02

2. POHYBOVÁ PLATFORMA

Jako pohybová platforma byl využit *Autonomní robot na soutěž BEAR RESCUE 2012* z předchozího bakalářského projektu [1]. Následně bude uveden stručný popis vlastností a vybavení tohoto robota, podrobnému popisu se věnuje výše zmíněná práce.

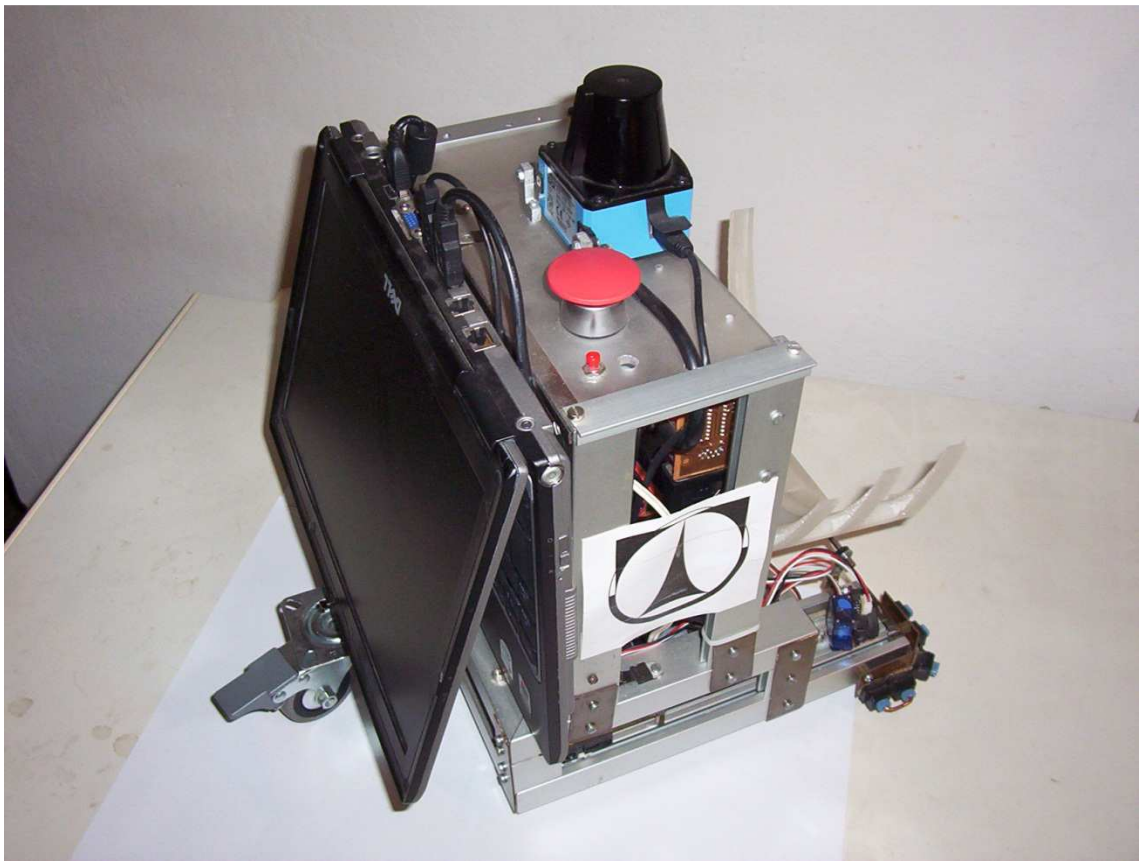


Obr. 3 Fotografie robotické platformy

2.1 MECHANICKÝ SUBSYSTÉM

Základem robota je dvoukolový diferenciálně řízený podvozek, jehož konstrukce je tvořena převážně z hliníkových profilů a spojovacích plechů. Některé části jsou vytištěné z plastu na 3D tiskárně.

Dva stejnosměrné motory s převodovkou a enkodéry zajišťují při napájení 12 V a otáčkách 170 ot/min točivý moment 0.15 Nm a maximální výkon 4,22W.



Obr. 4 Fotografie robotické platformy

2.2 SENZORICKÝ SUBSYSTÉM

Robot je vybaven kromě laserového měřícího senzoru popsaného v kapitole 1. dalšími senzory, mezi něž patří čtveřice infračervených senzorů vzdálenosti s rozsahem 10-80 cm. Dva jsou umístěny fixně v zadní části robota, další dva jsou připevněny na modelářských servech, pomocí níž lze měnit snímanou oblast senzoru. Kamera připevněná z vnitřku na vrchní desku je určená pro *Playstation 3* (PS3 EYE). Při rozlišení 320x240 pixelů zvládá pořizovat 120 snímků za sekundu. Dále je robot po obvodu osazen taktilními senzory, které mají jistící funkci.

Ze senzorického systému robota bylo pro tuto úlohu vyjma laserového měřícího senzoru použito pouze enkodérů.

2.3 ŘÍDICÍ SYSTÉM

Jako řídicí systém byl použit notebook umístěný svisle na podvozku. Bylo třeba provést úpravu displeje – jeho otočení – takže je na jeho obrazovku vidět když má sklopené víko. Panty notebooku byly téměř zrcadlově identické, a proto stačilo tyto panty prohodit a kabely k wifi anténě a k samotnému displeji protočit. Notebook má následující parametry:

DELL Latitude D420

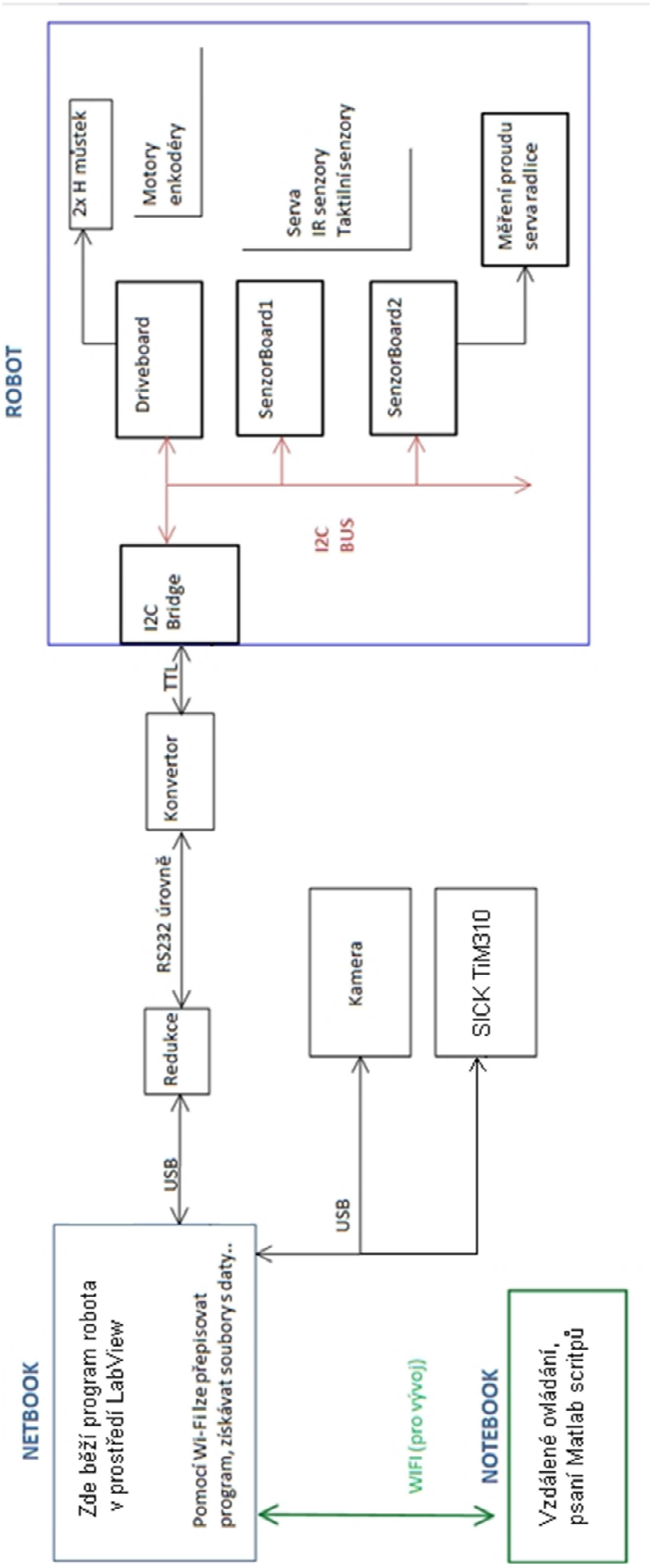
U2500 dual core @ 1.20 GHz

790 MHz, 2 GB RAM

2.3.1 ELEKTRONIKA

Elektronika robota je složena ze čtyř hlavních desek, každá obsahuje jeden mikrokontrolér ATmega 8. Desky jsou nazvány následovně: I2CBridge , DriveBoard, SenzorBoard a SenzorBoard2.

Netbook komunikuje prostřednictvím redukce USB↔RS232 a převodníkem napěťových úrovní (MAX232) s deskou I2CBridge, která přeposílá pakety požadované desce (podle adresy) přes I2C sběrnici. Informaci o natočení (resp. rychlosti otáčení) dávají kvadraturní enkodéry na každém z motoru. Blokové schéma celé elektroniky je uvedeno na Obr. 5.



Obr. 5 Blokové schéma elektroniky

3. SOFTWAROVÉ PROSTŘEDKY

Pro řízení mobilního robota slouží program vytvořený v prostředí LabView od firmy National Instruments. Kódy samotného algoritmu SLAM jsou použity z (v úvodu popsaného) kurzu [3] v jazyce prostředí MATLAB, bylo ale třeba doplnit několik funkcí a provést úpravy pro reálné použití.

Použité operační systémy:

Notebook na robotické platformě

– Microsoft Windows XP 32 bit SP3

Notebook pro vývoj a programování

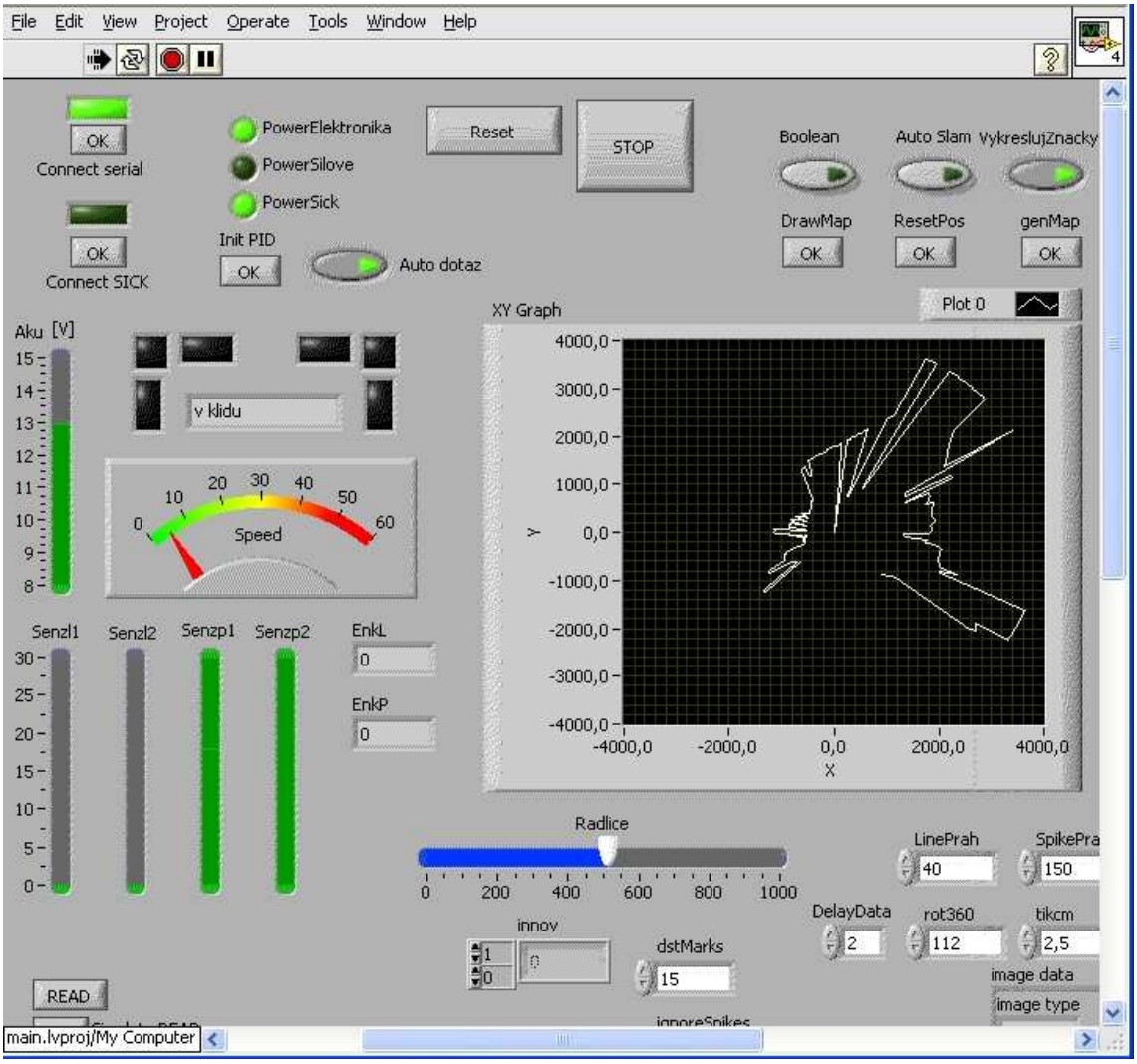
– Windows 7 Home premium 32 bit

Vývoj probíhal dle následujících bodů:

- Notebook pro vývoj byl připojen k notebooku na mobilním robotu přes program REAL VNC *viewer*, který umožnuje připojit se přes síť na plochu systému Windows
- Přes tuto vzdálenou plochu byl vyvíjen řídící program v LabView
- Na notebooku pro vývoj běžel MATLAB, jehož kořenový adresář byl síťový disk notebooku na mobilním robotu. Po úpravě m-souborů (skriptů MATLABu) se po restartování řídicího programu a zavolání příkazu „clear functions“ v příkazové řádce MATLABu, běžící na mobilním robotu, tyto soubory aktualizovaly.

3.1 LABVIEW

LabView je platforma návrhového a vývojového prostředí pro grafický programovací jazyk od firmy National Instruments. V rámci této práce byl vyvinut program, který má na starosti řízení pohybové platformy. Hlavní obrazovka tohoto programu je znázorněna na Obr. 6.



Obr. 6 Hlavní obrazovka programu sloužícího pro řízení pohybové platformy

Program byl dělen na dvě hlavní „while“ struktury, což znamená, že tyto struktury běží současně ve dvou oddělených vláknech. První obsluhuje události grafického uživatelského prostředí, druhá struktura obsluhuje periodický příjem a odesílání dat jak z laserového měřícího skeneru, tak z robotické platformy.

Komunikaci obsluhovanou programem můžeme rozdělit na komunikaci mezi robotem a počítačem a komunikaci mezi skenerem a počítačem.

3.1.1 Robot – PC

Tuto funkci zajišťují pomocné dynamické knihovny, jež byly v rámci této práce vytvořeny v jazyce C#. Tyto knihovny slouží pro poskytnutí funkcí, které upraví parametry do formátu potřebného pro poslání přes sběrnici. Každá funkce volaná z prostředí LabView má následující formát:

```
public byte[] Prikaz(int parametr1, int parametr2,…)
```

Výstupem je tedy vždy posloupnost bytů, které budou odeslány po sériové lince a vstupem jsou parametry dané funkce. Funkce může být dvojího typu, a to buď příkaz, nebo dotaz. Příkaz přepíše určité proměnné v adresované desce, nebo provede určitou akci. Dotaz naopak vyvolá přenos od adresované desky – ta odešle připravená data (např.: předchozím příkazem) – a předá libovolný počet dat. Tyto data následně přijdou v příjmové události. Tento postup je nutný z důvodu hierarchie *master-slave*, jelikož *slave* po I2C sběrnici nemůže zahajovat přenos.

3.1.2 Skener – PC

Byla vytvořena „subVI“ komponenta pro jednoduché nastavení režimu skeneru. Komponenta je uvedena na následujícím obrázku.

Obr. 7 Komponenta umožňující ovládání senzoru TiM310

VISA zdroj je reference na iniciované zařízení. Typ příkazu může nabývat hodnot:

0 – posílej data permanentně

1 – pošli data jednou

2 – přestaň posílat

3.1.3 Další funkce programu

Program realizuje několik dalších funkcí, jako je například oddělené zapínání napájecích okruhů, které umožňují celý systém úsporně napájet, což může být využito pro „stand-by“ režim při vzdáleném ovládání robota. Napájecí okruhy jsou děleny na napájení elektroniky, napájení serv a napájení skeneru. Dále program umožňuje přímé řízení podvozku pomocí klávesnice. Rychlost jízdy lze nastavit ukazatelem.

Program dále zobrazuje důležité informace o stavu robotické platformy, jako je napětí akumulátoru, informace z infračervených a taktilních senzorů a v neposlední řadě také data ze skeneru v komponentě „XY graf“.

3.2 MATLAB

MATLAB (MATrix LABoratory) je interaktivní programové prostředí a skriptovací programovací jazyk. MATLAB byl vyvinut společností MathWorks a umožňuje počítání s maticemi, vykreslování 2D i 3D grafů funkcí, implementaci algoritmů, počítačovou simulaci, analýzu a prezentaci dat a také mnoho dalších funkcí, které byly postupem času přidány ve formě „toolboxů“.

LabView dokáže volat skripty MATLABu tak, že po otevření projektu, který obsahuje komponentu „MATLAB script“ spustí jádro a příkazovou řádku MATLABu. Při zavolání výše zmíněné komponenty odešle skript do příkazové řádky. Toto jádro běží i při neaktivním řídicím programu. Pro zobrazování dat mapovacího algoritmu a výsledné mapy bylo použito finálně MATLABu. Původně se tato data zobrazovala v bitmapě v řídicím programu, do které probíhalo vykreslování za pomocí dynamické knihovny napsané v jazyce C#. Od tohoto řešení bylo upuštěno z důvodu velké výpočetní náročnosti.

4. SIMULTÁNNÍ LOKALIZACE A MAPOVÁNÍ

Simultánní lokalizace a mapování je často označována zkratkou „SLAM“. Informace v této kapitole byly čerpány a rovnice převzaty ze „SLAM kurzu“ [4].

SLAM je součástí tzv. *mapově orientované navigace* [2], která je zhruba dělena do tří procesů: lokalizace, učení mapy a plánování cesty. Třetím procesem se tato práce nezabývá.

Přístupů jak řešit SLAM je několik:

1. **Monte Carlo lokalizace** – na principu způsobu rozdělení hustoty pravděpodobnosti popisující odhad pozice robota
2. **Kalmanův filtr** – na principu odhadu plovoucího průměru pozice robota
3. **Markovova lokalizace** – na principu výpočtu pravděpodobnostního rozdělení na všech možných pozicích v mapě

Kalmanův filtr je v kybernetice pro úlohy lokalizace a řízení nejvíce používaný algoritmus, svá uplatnění má také ale v oblastech zpracování signálů a také v ekonomice. V našem případě bylo zvoleno (v souladu se SLAM kurzem) *rozšířené* verze Kalmanova filtru (zkratka EKF).

4.1 OBJEKTY VE SLAMu

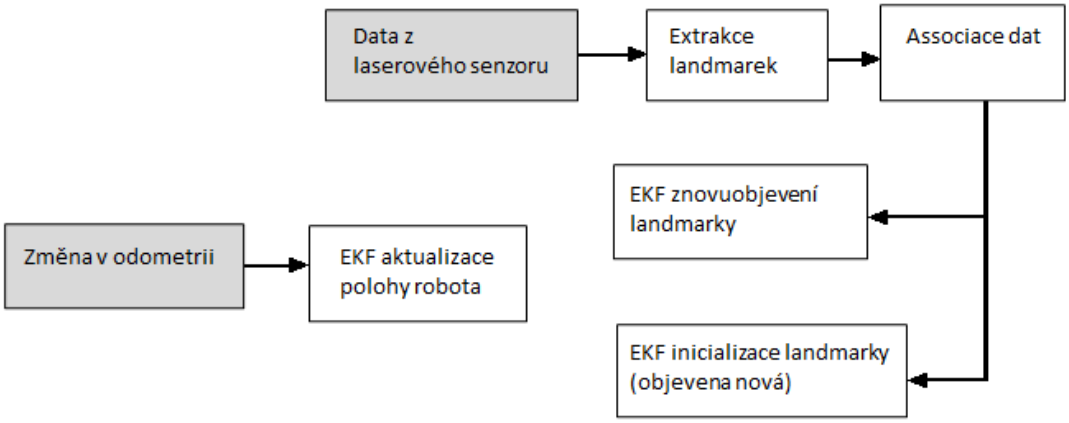
- Mapa
- Robot
- Laserový měřící senzor v roli *exteroceptivního* senzoru
- *Enkodéry* = snímače ujeté dráhy každého kola robota, v roli *proprioceptivního* senzoru
- *Landmarky* = význačné body v okolí, orientační body vyextrahované podle zvoleného algoritmu
- *Měření* = Každá dvojice exteroceptivní senzor – *landmark* definuje takovéto jedno *měření*. Je definováno dvěma hodnotami - úhlem a natočením.

4.2 PRINCIP SLAM

Pro funkci SLAM musí být robot vybaven alespoň jedním *exteroceptivním* senzorem, např.: kamerou, laserovým skenerem nebo sonarem, a dále je vhodné, aby robot mohl alespoň přibližně měřit vlastní pohyb, v případě kolového podvozku třeba snímače otáček jednotlivých kol, nebo akcelerometr či gyroskop, tedy potřebuje *proprioceptivní* senzor. SLAM sestává z následujících tří kroků:

- **Robot objeví význačné znaky ve svém okolí**, nazývané jako *landmarky*. Takovéto *landmarky* jsou nalezeny v okolí pomocí exteroceptivního senzoru, u něhož je známá chyba měření. Poloha robota má nyní také určitou nejistotu, takže je třeba vzít v úvahu obě tyto nejistoty a promítnout nalezené *landmarky* do mapy. Tento postup se nazývá *inverzní měřící model*.
- **Robot změní svojí polohu** a z dat proprioceptivních senzorů za pomocí *matematického pohybového modelu* vypočítá změnu své polohy a bere v potaz systematickou i náhodnou chybu těchto senzorů, která s každým dalším pohybem zvyšuje nejistotu polohy robota. I přes snahu o minimalizaci těchto chyb se vždy bude zvyšovat nejistota polohy až za mez použitelnosti.
- **Robot znovuobjeví landmarky, které již viděl.** Provede korekci vzhledem k rozdílu v poloze *landmarek* objevených již dříve a *landmarek* objevených nyní. Velikost korekce bude udána nejistotami poloh obou *landmarek* a také nejistotou polohy samotného robota. Korekcí je myšlena úprava poloh a snížení jejich nejistot. Pro výpočet polohy landmarky, kterou již viděl, se použije takzvaného *přímého měřícího modelu*.

Pro správnou funkci SLAM procesu je nutné udržovat všechny polohy a jejich nejistoty co nejaktuálnější, aby docházelo k patřičné propagaci nejistot a jejich korekcí.



Obr. 8 Posloupnost akcí při SLAM

Na Obr. 8 můžeme vidět schéma procesu SLAM. Po příjmu dat z laserového skeneru dojde k extrakci *landmarek* a následně musí být asociovány s databází uvnitř systému. Tak je možné určit, jestli jde o nově objevenou *landmarku* či ne. Při příjmu dat z *enkodérů* proběhne aktualizace polohy robota, a tím se zvýší nejistota jeho pozice.

4.3 POHYBOVÉ A MĚŘÍCÍ MODELY

4.3.1 Pohybový model

Robot se pohybuje v závislosti na řídícím signálu **u** a poruše **n** a aktualizuje tím svůj stav $\mathcal{R}$ (tedy polohový vektor – souřadnice x a y a úhel).

$$\mathcal{R} \leftarrow f(\mathcal{R}, \mathbf{u}, \mathbf{n})$$

V našem případě jsou řídícím signálem data z proprioceptivního senzoru, tedy z enkodérů. V obecném případě může být tímto signálem například pouze informace o tom, jakou vzdálenost by robot „měl“ dle příkazu ujet. Porucha **n** odpovídá chybě, která vzniká měřením pomocí enkodérů.

4.3.2 Přímý měřící model

Robot $\mathcal{R}$ objevuje *landmarku* $\mathcal{L}_i$. Tato *landmarka* byla již jednou změřena senzorem $\mathcal{S}$. Jinak řečeno, tato funkce převede známý stav *landmarky* $\mathcal{L}_i$ (tedy její polohu – x a y) na měření $\mathbf{y}_i$ (tedy na informaci ve stejném formátu, jaký poskytuje senzor – vzdálenost a úhel). Senzor $\mathcal{S}$ je udáván z toho důvodu, aby byla definována jeho relativní poloha a orientace vůči robotu.

$$\mathbf{y}_i = h(\mathcal{R}, \mathcal{S}, \mathcal{L}_i)$$

4.3.1 Inverzní měřící model

Následně se vypočítá stav nově objevené *landmarky* $\mathcal{L}_i$. Měření $\mathbf{y}_i$ bylo provedeno senzorem $\mathcal{S}$ z polohy robota $\mathcal{R}$.

$$\mathcal{L}_j = g(\mathcal{R}, \mathcal{S}, \mathbf{y}_j)$$

V nejlepším případě bude, v rámci jednoho měření, funkce *g( )* inverzní k funkci *h( )*. Pokud by měření neobsahovalo všechny stupně volnosti stavu *landmarky*, pak inverze neplatí a *g( )* nelze definovat. Takovýto případ nastane pokud exteroceptivním senzorem je například kamera, která nám neposkytne informaci o vzdálenosti, ale pouze o směru *landmarky*.

4.4 EKF PRO SLAM

V EKF-SLAM je mapa tvořena „velkým“ vektorem, definujícím stavy robota a objevených *landmarek*, u robota tedy polohou a natočením, u *landmarek* pouze polohou. Tato mapa, obvykle nazývána stochastická mapa, je spravována EKF za pomocí procesů predikce (když se robot pohne) a korekce (když senzory znovuobjeví landmarku, která již byla zmapována). Pro progresivní tvoření mapy je EKF vybaven krokem, který má za úkol přidat nově objevené *landmarky* do vektoru mapy. Inicializace *landmarky* je realizována výpočtem stavu *landmarky* z inverzní měřící funkce, jejích Jakobiánů a údaji o pozici senzoru a o jeho měření. Také musí být vypočítány kovariance a křížové

kovariance se zbytkem mapy. Tato data jsou pak následně přidána do stavového vektoru a do kovarianční matice. Tabulka 1 uvádí podobnosti mezi EKF a EKF pro SLAM.

Událost	**EKF - SLAM**	**EKF**
Robot se pohne	Pohyb robota	EKF predikce
Senzor detekuje novou *landmarku*	Inicializace *landmarky*	Zvětšení stavu
Senzor detekuje známou *landmarku*	Korekce mapy	EKF korekce
Zmapovaná *landmarka* je neplatná	Vymazání *landmarky*	Redukce stavu

Tabulka 1

Doplňující poznámka: Kovariance je střední hodnota součinu odchylek obou náhodných veličin X a Y od jejich středních hodnot.

$$\sigma(x, y) = \mathrm{E}\left[(x - \mathrm{E}[x])(y - \mathrm{E}[y])\right] = \mathrm{E}\left[xy\right] - \mathrm{E}\left[x\right]\mathrm{E}\left[y\right]$$

4.5 MAPA

Mapa je tvořena „velkým“ stavovým vektorem, uchovávající informaci o stavu robota a *landmarek*,

$$\mathbf{x} = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix}$$

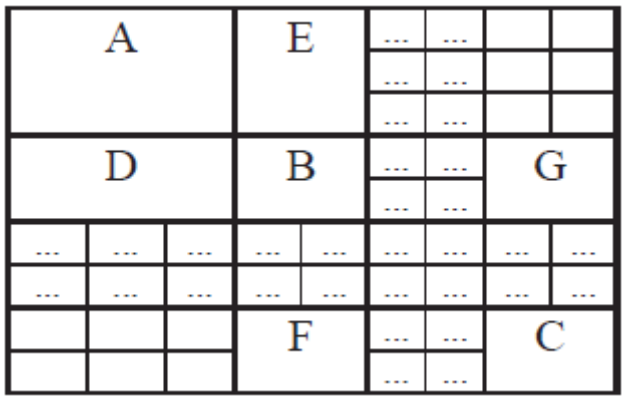
kde $\mathcal{R}$ je stav robota (tedy jeho poloha a natočení) a $\mathcal{M} = (\mathcal{L}_1, \cdots, \mathcal{L}_n)$ je posloupnost o velikosti *n* obsahující jednotlivé stavy *landmarek* (tedy jejich polohy). V EKF je mapa modelována za pomocí Gaussova rozložení, používající plovoucí průměr a kovarianční matici stavového vektoru, označené jako $\overline{\mathbf{x}}$ a P.

$$\overline{\mathbf{x}} = \begin{bmatrix} \overline{\mathcal{R}} \\ \overline{\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \overline{\mathcal{R}} \\ \overline{\mathcal{L}}_1 \\ \vdots \\ \overline{\mathcal{L}}_n \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{M}} \\ \mathbf{P}_{\mathcal{M}\mathcal{R}} & \mathbf{P}_{\mathcal{M}\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{R}\mathcal{L}_n} \\ \mathbf{P}_{\mathcal{L}_1\mathcal{R}} & \mathbf{P}_{\mathcal{L}_1\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{L}_1\mathcal{L}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{\mathcal{L}_n\mathcal{R}} & \mathbf{P}_{\mathcal{L}_n\mathcal{L}_1} & \cdots & \mathbf{P}_{\mathcal{L}_n\mathcal{L}_n} \end{bmatrix}$$

Kde $\mathbf{P}_{\mathcal{R}\mathcal{R}}$ je kovarianční matice pozice robota, $\mathbf{P}_{\mathcal{R}\mathcal{M}}$ a $\mathbf{P}_{\mathcal{M}\mathcal{R}}$ jsou tzv. *křížové* kovariance, tedy kovariance mezi pozicí robota a *landmarkami*. Zbývá matice $\mathbf{P}_{\mathcal{M}\mathcal{M}}$, která obsahuje po diagonále kovariance pozic *landmarek* a zbytek jsou kovariance mezi jednotlivými *landmarkami*.

Cílem EKF-SLAM je udržet mapu (tedy vektory $\overline{\mathbf{x}}$ a P) neustále aktuální.

Rozměry vektorů: V našem případě, kde mapa je dvoudimenzionální prostor, je poloha robota udána souřadnicemi **x**, **y** a **úhlem** a poloha každé *landmarky* je udána pouze souřadnicemi **x** a **y**, protože význačný bod v mapě nemá orientaci. Rozměry celé kovarianční matice jsou pak pro větší názornost zobrazené na Obr. 9.



Obr. 9 Znázornění rozměrů kovarianční matice **P**, převzato z [7]

Matice A, tedy kovariance pozice robota, má rozměr 3x3. Matice E a D, tedy kovariance mezi *landmarkami* a robotem, mají rozměry 2x3 a 3x2 a jsou vzájemně transponované. Matice B a C jsou kovariance poloh *landmarek* a mají rozměr 2x2. Matice F a G jsou *křížové* kovariance mezi jednotlivými *landmarkami*.

4.6 INICIALIZACE MAPY

Na počátku je mapa neobsahuje žádné *landmarky*, takže $n = 0$ a $\mathbf{x} = \mathcal{R}$. Pokud uvažujeme startovní pozici robota jako počátek mapy, nejistota počáteční pozice je nulová, pak platí:

$$\overline{\mathbf{x}} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

4.7 POHYB ROBOTA

Pro klasické EKF máme následující funkci, kde **u** je kontrolní vektor a **n** je vektor poruchy.

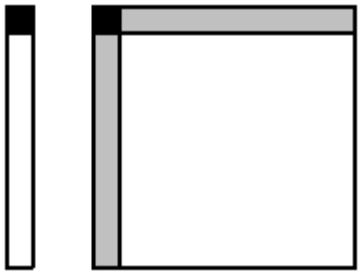
$$\mathbf{x} \leftarrow f(\mathbf{x}, \mathbf{u}, \mathbf{n})$$

Pak rovnice pro predikční krok vypadají následovně (vektor šumu promítneme do kovarianční matice, a do změny stavu ho zanášet nebudeme):

$$\overline{\mathbf{x}} \leftarrow f(\overline{\mathbf{x}}, \mathbf{u}, 0)$$
$$\mathbf{P} \leftarrow \mathbf{F}_{\mathbf{x}}\mathbf{P}\mathbf{F}_{\mathbf{x}}^{\top} + \mathbf{F}_{\mathbf{n}}\mathbf{N}\mathbf{F}_{\mathbf{n}}^{\top}$$

$$\mathbf{F}_{\mathbf{x}} = \frac{\partial f(\overline{\mathbf{x}}, \mathbf{u})}{\partial \mathbf{x}} \qquad \mathbf{F}_{\mathbf{n}} = \frac{\partial f(\overline{\mathbf{x}}, \mathbf{u})}{\partial \mathbf{n}}$$

Na Obr. 10 je vlevo znázorněn stavový vektor **x** a vpravo kovarianční matice **P**. Černou a šedou barvou jsou zvýrazněny části, které pohyb robota ovlivní. Černá část vektoru **x** odpovídá stavu robota $\overline{\mathcal{R}}$, černá část kovarianční matice je kovariance pozice robota $\mathbf{P}_{\mathcal{R}\mathcal{R}}$ a šedé pruhy odpovídají křížovým kovariancím $\mathbf{P}_{\mathcal{R}\mathcal{M}}$ a $\mathbf{P}_{\mathcal{M}\mathcal{R}}$ (tedy kovariancím mezi pozicí robota a všemi *landmarkami*).



Obr. 10 Aktualizované části vektoru **x** a matice **P**, převzato ze [4]

Po zjednodušení dostaneme následující rovnice EKF, které budou použity pro predikci při pohybu robota:

$$
\begin{aligned}
\overline{\mathcal{R}} &\leftarrow f_{\mathcal{R}}(\overline{\mathcal{R}}, \mathbf{u}, 0) \\
\mathbf{P}_{\mathcal{R}\mathcal{R}} &\leftarrow \frac{\partial f_{\mathcal{R}}}{\partial \mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{R}} \frac{\partial f_{\mathcal{R}}}{\partial \mathcal{R}}^{\top} + \frac{\partial f_{\mathcal{R}}}{\partial \mathbf{n}} \mathbf{N} \frac{\partial f_{\mathcal{R}}}{\partial \mathbf{n}}^{\top} \\
\mathbf{P}_{\mathcal{R}\mathcal{M}} &\leftarrow \frac{\partial f_{\mathcal{R}}}{\partial \mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{M}} \\
\mathbf{P}_{\mathcal{M}\mathcal{R}} &\leftarrow \mathbf{P}_{\mathcal{R}\mathcal{M}}^{\top}
\end{aligned}
$$

Kde **N** je kovarianční matice chyby n pro pohyb robota.

4.8 OBJEVENÍ ZNÁMÝCH LANDMAREK

Zde je uvedena sada rovnic, definující aktualizaci stavu kovarianční matice **P** a stavového vektoru $\overline{\mathbf{x}}$.

$$
\begin{aligned}
\overline{\mathbf{z}} &= \mathbf{y}_i - h_i(\overline{\mathcal{R}}, \mathcal{S}, \overline{\mathcal{L}}_i) \\
\mathbf{Z} &= \begin{bmatrix} \mathbf{H}_{\mathcal{R}} & \mathbf{H}_{\mathcal{L}_i} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{L}_i} \\ \mathbf{P}_{\mathcal{L}_i\mathcal{R}} & \mathbf{P}_{\mathcal{L}_i\mathcal{L}_i} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{\mathcal{R}}^{\top} \\ \mathbf{H}_{\mathcal{L}_i}^{\top} \end{bmatrix} + \mathbf{R} \\
\mathbf{K} &= \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{L}_i} \\ \mathbf{P}_{\mathcal{M}\mathcal{R}} & \mathbf{P}_{\mathcal{M}\mathcal{L}_i} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{\mathcal{R}}^{\top} \\ \mathbf{H}_{\mathcal{L}_i}^{\top} \end{bmatrix} \mathbf{Z}^{-1} \\
\overline{\mathbf{x}} &\leftarrow \overline{\mathbf{x}} + \mathbf{K}\overline{\mathbf{z}} \\
\mathbf{P} &\leftarrow \mathbf{P} - \mathbf{K}\mathbf{Z}\mathbf{K}^{\top}
\end{aligned}
$$

Kde **R** je kovarianční matice chyby měření. Vektor $\overline{\mathbf{z}}$, tzv. *inovace*, je hodnota udávající jak velký je rozdíl mezi předpokládanými a skutečnými pozicemi. Matice **Z** má stejný význam, až na to, že se týká kovarianční matice (tedy rozložení nejistot).

Objevení známých *landmarek* probíhá v EKF často postupně po jedné. Jeden krok pak závisí pouze na stavu robota $\mathcal{R}$, senzoru $\mathcal{S}$, a jedné *landmarky* $\mathcal{L}_i$ .Pro aktualizaci je potřeba projet v cyklu všechny objevené *landmarky*. Pak měřící model definuje následující vztah,

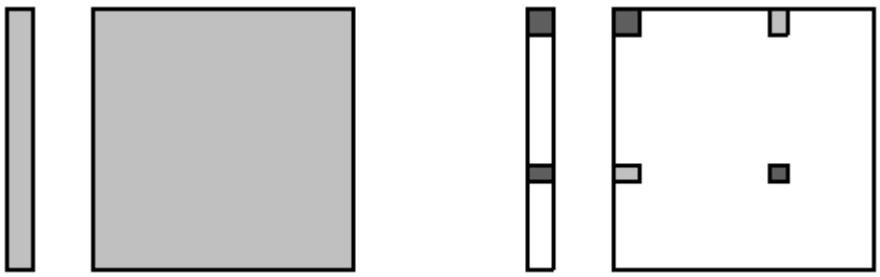
$$\mathbf{y}_i = h_i(\mathcal{R}, \mathcal{S}, \mathcal{L}_i) + \mathbf{v}$$

ve kterém nezáleží na žádné jiné *landmarce* než na $\mathcal{L}_i$. Pak je Jakobián $\mathbf{H_x}$ řídká matice:

$$\mathbf{H_x} = \begin{bmatrix} \mathbf{H}_{\mathcal{R}} & 0 & \cdots & 0 & \mathbf{H}_{\mathcal{L}_i} & 0 & \cdots & 0 \end{bmatrix}$$

$$\mathbf{H}_{\mathcal{R}} = \frac{\partial h_i(\overline{\mathcal{R}}, \mathcal{S}, \overline{\mathcal{L}}_i)}{\partial \mathcal{R}} \qquad \mathbf{H}_{\mathcal{L}_i} = \frac{\partial h_i(\overline{\mathcal{R}}, \mathcal{S}, \overline{\mathcal{L}}_i)}{\partial \mathcal{L}_i}.$$

Na Obr. 11 můžeme vidět aktualizované části stavového vektoru a kovarianční matice **P**. Vlevo je vidět, že je ovlivněn celý stav a to díky Kalmanovu zesílení **K**. Vektor *inovace* ale ovlivní pouze stav robota, stav jedné landmarky, kovariance jejich pozic a vzájemnou kovarianci. Toto vyplývá z řídkosti matice $\mathbf{H_x}$.

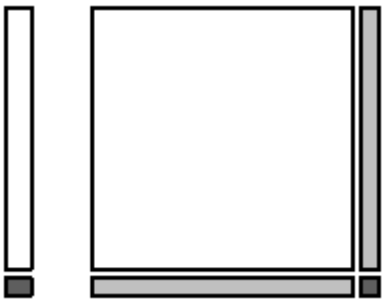


Obr. 11 Znázornění aktualizace stavového vektoru a kovarianční matice, převzato ze [4]

4.9 INICIALIZACE LANDMAREK

Inicializace *landmarek* proběhne, pokud robot změří význačný bod, který nebyl nikdy ještě zmapován. Tato operace zvyšuje velikost stavového vektoru a kovarianční matice. Pokud známe všechny stupně volnosti popisující stav nové *landmarky*, stačí nám invertovat měřící funkci *h( )*, a vypočítat ho ze stavu robota $\mathcal{R}$, senzoru $\mathcal{S}$, a z měření $\mathbf{y}_{n+1}$.

$$\mathcal{L}_{n+1} = g(\mathcal{R}, \mathcal{S}, \mathbf{y}_{n+1})$$



Obr. 12 Znázornění částí, které jsou přidány do stavu, převzato ze [4]

Jako první vypočítáme průměr *landmarky* a Jakobián funkce.

$$\begin{aligned}
\overline{\mathcal{L}}_{n+1} &= g(\overline{\mathcal{R}}, \mathcal{S}, \mathbf{y}_{n+1}) \\
\mathbf{G}_{\mathcal{R}} &= \frac{\partial g(\overline{\mathcal{R}}, \mathcal{S}, \mathbf{y}_{n+1})}{\partial \mathcal{R}} \\
\mathbf{G}_{\mathbf{y}_{n+1}} &= \frac{\partial g(\overline{\mathcal{R}}, \mathcal{S}, \mathbf{y}_{n+1})}{\partial \mathbf{y}_{n+1}}
\end{aligned}$$

Dále vyjádříme kovarianci pozice *landmarky* $\mathbf{P}_{\mathcal{L}\mathcal{L}}$ a její *křížové* kovariance se zbytkem mapy $\mathbf{P}_{\mathcal{L}\mathbf{x}}$.

$$\begin{aligned}
\mathbf{P}_{\mathcal{L}\mathcal{L}} &= \mathbf{G}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{R}}\mathbf{G}_{\mathcal{R}}^{\top} + \mathbf{G}_{\mathbf{y}_{n+1}}\mathbf{R}\mathbf{G}_{\mathbf{y}_{n+1}}^{\top} \\
\mathbf{P}_{\mathcal{L}\mathbf{x}} &= \mathbf{G}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathbf{x}} = \mathbf{G}_{\mathcal{R}}\left[\mathbf{P}_{\mathcal{R}\mathcal{R}}\ \mathbf{P}_{\mathcal{R}\mathcal{M}}\right]
\end{aligned}$$

A konečně aktualizujeme celý stav a kovarianční matici.

$$\overline{\mathbf{x}} \leftarrow \begin{bmatrix} \overline{\mathbf{x}} \\ \overline{\mathcal{L}}_{n+1} \end{bmatrix}$$
$$\mathbf{P} \leftarrow \begin{bmatrix} \mathbf{P} & \mathbf{P}_{\mathcal{L}\mathbf{x}}^{\top} \\ \mathbf{P}_{\mathcal{L}\mathbf{x}} & \mathbf{P}_{\mathcal{L}\mathcal{L}} \end{bmatrix}$$

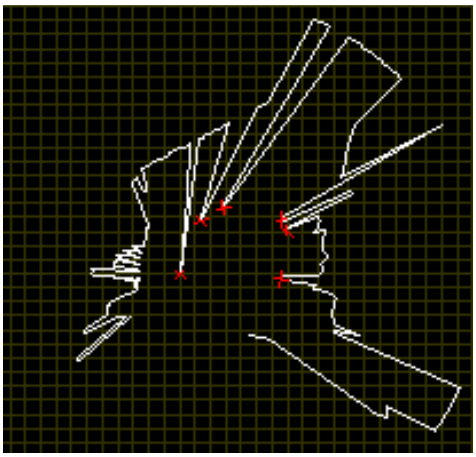
4.10 EXTRAKCE LANDMAREK

Způsobů jak nalézt *landmarky* v senzorových datech je několik. Zvolení správného způsobu záleží na prostředí, v jakém se robot pohybuje a také na dostupném výpočetním výkonu.

4.10.1 SPIKE LANDMARKY

Extrakce tzv. „Spike“ *landmarek* probíhá tak, že se v datech najdou extrémy. Tento způsob je velice jednoduchý a je vhodný pro prostředí, kde jsou spousty objektů umístěny v prostoru. Naopak je velice nevhodný např. do prázdných místností nebo koridorů, kde jsou rovné stěny.

Nalézt extrémy v našich senzorových datech v polárním tvaru můžeme např. pomocí diference následujícího a předchozího údaje. Pokud je tato diference větší než práh, můžeme tento bod považovat za *landmarku*. Na Obr. 13 můžeme vidět *landmarky* označené červenými křížky, které byly vyextrahovány dle tohoto způsobu. Tento způsob je poměrně výpočetně nenáročný, cenou je ale nízká robustnost.



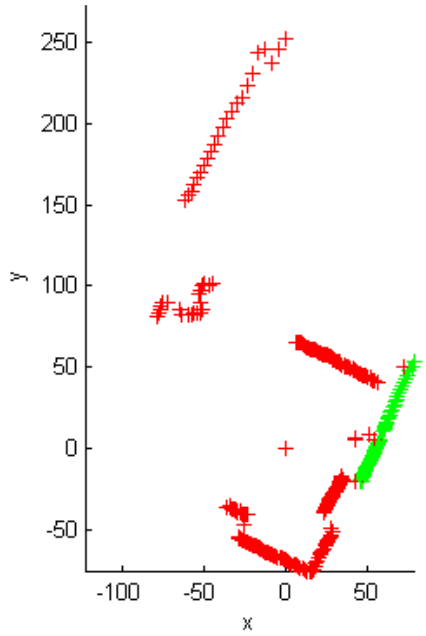
Obr. 13 „Spike“ *landmarky*

4.10.2 RANSAC ALGORITMUS

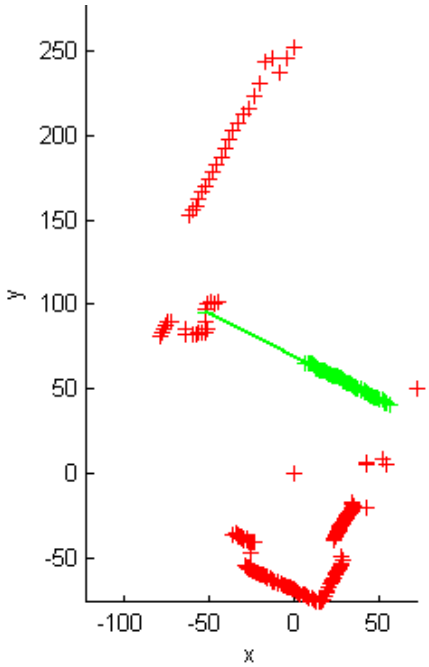
Druhou možností jak vyextrahovat *landmarky* ze senzorových dat je v nich hledat úsečky. To umožní tzv. „RANSAC“ algoritmus (z angl. Random Sample Consensus). Využívá náhodného umísťování úseček do senzorových dat a iteračního výpočtu jak moc se tato úsečka shoduje s daty. Pokud se úsečka shoduje dostatečně (to určí prahová konstanta), tak se data ležící na úsečce odstraní a provede se znovu celý algoritmus. Takto lze nalézt více úseček (tedy stěn) v jednom skenu. Tento postup je patrný z Obr. 14 a z Obr. 15.

Úsečky jsou pro reprezentaci ve 2D prostoru převedeny na body, a to tak, že se nalezne pomocí trigonometrie průsečík přímky, která vede počátkem souřadného systému a je kolmá k převáděné úsečce. Tento přístup je díky iteračnímu charakteru podstatněji výpočetně náročnější nežli extrakce „Spike“ *landmarek*.

V rámci práce byl použit „RANSAC-Toolbox“ pro Matlab [9], jehož použití popisuje tento text [8].



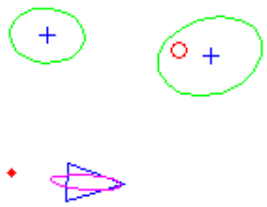
Obr. 14 První aplikace RANSAC algoritmu



Obr. 15 Druhá aplikace RANSAC algoritmu

5. REALIZACE

Finální program zobrazuje současný stav systému v „plot“ funkci Matlabu, jednoduchý případ můžeme vidět na Obr. 16. Jedná se o polohu robota (modrý trojúhelník), pravděpodobnost jeho pozice (červená elipsa), pozice pouze z odometrie (červená tečka), pozice *landmarky* (červený kroužek), změřená pozice *landmarky* (modrý křížek) a konečně pravděpodobnost její pozice (zelená elipsa).



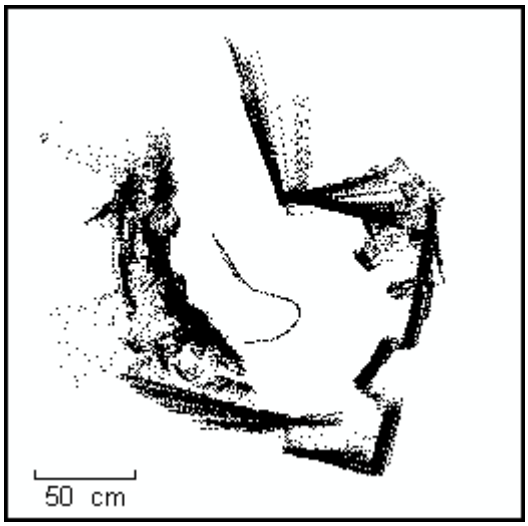
Obr. 16 Grafický výstup programu v akci

Robot tedy změří nějakou *landmarku*, tedy význačný bod v mapě, porovná se zapamatovanou pozicí a následně (s ohledem na pravděpodobnosti) provede korekci jak pozice robota, tak *landmarky* a i *landmarek* v okolí. Takovýto proces je opakován pro každý znovuobjevený význačný bod v mapě. Z obrázku je také vidět, že *landmarka* vpravo nebyla nyní změřena. Pro správnou funkci EKF nemusí být objeveny všechny význačné body, ale musí jich být dostatek, pro potřebně „důveryhodné“ vstupní informace.

Pokud je tento proces úspěšný, tedy známe pozici robota v průběhu pohybu, stačí nám vykreslovat data ze senzoru do rastru. Tím získáme mapu blízkého okolí robota. Následující dva obrázky jsou právě takovouto mapou, která byla získána na robotické platformě popsané v rámci této práce po krátké jízdě v obývacím pokoji a v menší místnosti.

Obr. 17 Výsledná mapa po jízdě v obývacím pokoji

V obrázku křivka v prostoru označuje trajektorii pohybu robota. V Obr. 17 jsou patrné stěny a židle uprostřed. Robot se pohyboval po koberci, což je velmi nevhodný povrch pro navigaci pouze z *odometrie*, ale díky korekci EKF bylo dosaženo lepšího výsledku. Výsledek není dokonalý, s každým dalším posunem robota dochází k mírně chybnému posunu (rotaci), jež je vidět např. na spodní stěně, která by měla být zobrazena jako čistá rovná čára. Pro získání těchto map bylo použito extrakce „Spike“ landmarek.



Obr. 18 Další výsledná mapa po jízdě v malé místnosti

ZÁVĚR

Mobilní robot byl upraven o laserový měřící senzor, byl vyvinut kompletní ovládací program pro jeho řízení a dále byly upraveny a úspěšně aplikovány algoritmy realizující simultánní lokalizaci a mapování, jejímž výsledkem je mapa, ze které lze usoudit, že uvedený laserový skener je pro tuto aplikaci použitelný.

Vyzkoušena byla extrakce jak „Spike“ *landmarek*, tak extrakce za pomocí „RANSAC“ algoritmu. Finálně byla použitelná pouze „Spike“ extrakce, a to z toho důvodu, že výpočetní prostředky nestačily „RANSAC“ algoritmus v Matlabu zpracovávat v reálném čase. Použití „RANSAC“ algoritmu (nebo kombinace obou algoritmů) by ale jistě přineslo mnohem uspokojivější výsledek, a to vzhledem k tomu, že vnitřní prostory budov jsou charakteristické rovnými plochami, které by tento algoritmus mohl výhodně používat pro korekci polohy. „Spike“ *landmarky* se projevily jako nepříliš stabilní orientační body v prostředí interiérů.

Je zde také velký prostor pro vylepšení celého systému (např. formou diplomové práce), a to například v oblasti optimalizace rychlosti celého programu pro možnost použít dokonalejší extrakci *landmarek*, nebo v oblasti asociace *landmarek*, tedy postupu, při němž se určuje k jaké uložené *landmarce* odpovídá nově objevená *landmarka*.

SEZNAM POUŽITÝCH ZKRATEK A VYSVĚTLIVKY

SLAM	*Simultánní lokalizace a mapování*
EKF	*Rozšířený Kalmanův filtr*
Skener	*Laserový měřící senzor v roli exteroceptivního senzoru, v našem případě TiM310*
Enkodéry	*snímače ujeté dráhy každého kola robota, v roli proprioceptivního senzoru*
Landmarky	*význačné body v okolí; orientační body vyextrahované podle zvoleného algoritmu*
2D/3D	*dvou/tří dimenzionální prostor*
Exteroceptivní	*přijímající podněty z vnějšího prostředí*
Proprioceptivní	*vnímající pohyb / polohu svého těla*

SEZNAM SOUBORŮ NA PŘÍLOŽENÉM CD

Bakalářská práce Zbyšek Zapadlík.docx (.pdf) – elektronická podoba této práce

SLAM course.pdf - SLAM kurz [4]

Labview projekt - Projekt řídícího programu pro prostředí Labview

Matlab skripty – Složka obsahující m-file, volané řídícím programem

KrytonComm (+ .dll) – Projekt a knihovna pro komunikaci s robotickou platformou pro Visual Studio 2010.

SEZNAM POUŽITÉ LITERATURY

[1] Valšík, M.- Zapadlík, Z.. Autonomní robot na soutěž BEAR RESCUE 2012. Bakalářský projekt, TUL-FM, 2012.

[2] NEHMZOW, U. 2003. *Mobile Robotics, A Practical Introduction*. Springer-Verlag London.

[3] Joan Solà. INTERACTIVE COURSE ON EKF AND SLAM [online]. [cit. 2013-05-04]. Dostupné z: http://www.joansola.eu/JoanSola/eng/course.html

[4] Joan Solà. Simulataneous localization and mapping with the extended Kalman filter [online]. [cit. 2013-05-04]. Dostupné z: http://www.joansola.eu/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf

[5] SICK pdf document services [online]. [cit. 2013-05-04]. Dostupné z: https://www.mysick.com/saqqara/pdf.aspx?id=im0042231

[6] SICK telegram listing [online]. [cit. 2013-05-04]. Dostupné z: http://www.rakurs.su/files/SICK/VMS/LMS1xx_LMS5xx_TiM3xx_JEF300_JEF500%20English.pdf

[7] Søren Riisgaard and Morten Rufus Blas. SLAM for Dummies [online]. [cit. 2013-05-04]. Dostupné z: http://www.core.org.cn/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring-2005/9D8DB59F-24EC-4B75-BA7A-F0916BAB2440/0/1aslam_blas_repo.pdf

[8] Marco Zuliani. RANSAC for Dummies [online]. [cit. 2013-05-04]. Dostupné z: http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf

[9] Marco Zuliani. RANSAC-Toolbox [online]. [cit. 2013-05-04]. Dostupné z: https://github.com/RANSAC/RANSAC-Toolbox

[10] Guido Zunino. Simultaneous Localization and Mapping for Navigation in Realistic Environments [online]. [cit. 2013-05-04]. Dostupné z: http://www.nada.kth.se/utbildning/forsk.utb/avhandlingar/lic/020220.pdf