Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Distribuovaná riadiaca infraštruktúra mobilného robota

Bakalárska práca

2020

Martin Slimák

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Distribuovaná riadiaca infraštruktúra mobilného robota

Bakalárska práca

Študijný program:	Aplikovaná Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2020

Martin Slimák

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Slimák
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Distribuovaná riadiaca infraštruktúra mobilného robota
Distributed Control Infrastructure of a Mobile Robot

Anotácia: V priebehu uplynulých 11 rokov vznikli v robotickej skupine tri platformy mobilných robotov: Smelý Zajko, ktorý je určený predovšetkým na súťaž RoboTour a úlohy v exteriéri, Kocúr Mikeš, ktorý plní lokalizačné, mapovacie a jednoduché manipulačné úlohy v interiéri a menší Janko Hraško, robot s pohonom všetkých 4 kolies so stereovidením a ramenom, určený na malé maniulačné úlohy. Postupne dozrel čas na novú revíziu hardvérovej výbavy s cieľom vytvorenia verzatilnej platformy s vysokým podvozkom, dostatočne modulárnej a využiteľnej tak v interiéri i exteriéri a rozšíriteľnej o nadstavbu s manipulátormi. Otvorí nám to cestu k zapájaniu sa do aktivít, ktoré boli doteraz pre nás nedostupné. To je zároveň príležitosť pre vytvorenie distribuovanej modulárnej multiplatformovej riadiacej softvérovej infraštruktúry novej generácie, ktorá oživí pripravovaný hardvér. Súčasťou frameworku, ktorý v práci študent vytvorí, bude možnosťladenia, logovania a analyzovania logov, jednotný adresný priestor, vzor subscriber, alternatívne získavanie údajov na požiadanie, priama podpora aj nízkoúrovňových platforiem, riadenia v reálnom čase, rozličné topológie prepojenia a vysoká efektivita.

Literatúra: Illah Reza Nourbakhsh and Roland Siegwart: Introduction to Autonomous Mobile Robots, MIT Press, 2004.
Howie Choset et al: Principles of Robot Motion, Theory, Algorithms, and Implementations, MIT Press, 2005.

Kľúčové slová: mobilný robot, distribuovaná architektúra, riadenie

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 15.10.2019

Dátum schválenia: 15.10.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

V tejto práci sa budeme venovať mobilným robotom. Spravíme prehľad existujúcich robotických systémov a ich využívanie. Hlavným cieľom robota Maťo je, aby bol schopný zúčastniť sa na robotických súťažiach. Spomenieme aké druhy pohonu sa použíivajú pri stavbe robotov a prečo sa používajú. Zhrnieme aj používané senzory a ich výhody a nevýhody. Objasníme prečo sme použili dané technológie pri stavbe robota Maťo. Navrhneme robota od konštrukcie až po framework, ktorý bude riadiť a spravovať dáta a komunikáciu pre akékoľvek pripojené moduly. Podrobne popíšeme celý framework, ktorý sme naprogramovali a na záver zhrnieme ako dopadol v testovaní a čo sa dá v budúcnosti ešte zlepšiť.

Abstract

The aim of this work will focus on mobile robots. We will make an overview of existing robotic systems and their use. The main goal of the robot Mato is to be able to participate in robotic competitions. We will mention what types of drive are used in the construction of robots and why they are used. We will also summarize the sensors used and their advantages and disadvantages. We will explain why we used the given technologies in the construction of the Maťo robot. We will design a robot from construction to a framework that will control and manage data and communication for any connected modules. We will describe in detail the whole framework that we have programmed and in the end we will summarize how it turned out in testing and what can be improved in the future.

Kľúčové slová: mobilný robot, distribuovaná architektúra, riadenie

Obsah

Zoznam obrázkov

Úvod

Robotika sa v dnešnej dobe rozmohla natoľko, že robotov vidíme skoro v každej oblasti života. Každý robot slúži na nejaký účel a každý účel vyžaduje špecifické vlastnosti, vybavenie, senzorické a pohybové schopnosti robota. Moja práca sa bude zaoberať robotom Maťo, ktorý je určený na navigáciu a presun v exteriéri a v budúcnosti po doplnení robotického ramena aj na menšie manipulačné práce. Robota Mato som navrhol, skonštruoval a sfunkčnil ako súčasť tejto bakalárskej práce. Mojimi cieľmi pri stavbe nového robota bolo priniesť nielen nové možnosti zapájania hardwaru, ako sú nové senzory, ďalšie pohonné systémy alebo komplexnejšie moduly, ako je robotické rameno, ale aj navrhnúť a naimplementovať riadiacu architektúru založenú na rýchlych a efektívnych interných riadiacich procesoch.

Každý robot potrebuje spoľahlivý systém, ktorý ho bude riadiť a ktorý bude zjednodušovať prácu programátorovi, aby sa nemusel zaťažovať programovaním všetkých tried, ale mal by sa venovať hlavne riešeniu problémov ktoré napomôžu robotovi úspešne prekonať prekážky a splniť jeho účel. V mojej práci sa budem zaoberať softvérovým a hardvérovým riešením pre robota a budem pritom vychádzať nielen z existujúcich riešení, ktoré sú vo svete používané, ale aj riešení z projektov, ktoré boli určené priamo na používaný hardvér a na konkrétneho robota.

Hlavnými prínosmi tejto práce z informatického hľadiska bude nová distribuovaná infraštruktúra, ktorá bude zapojená v sieti nad TCP/IP. Hoci je inšpirovaná už existujúcimi riešeniami ako je robotický operačný systém (ROS) alebo riadiaci framework robota Mikeš, svojimi vlastnosťami je oveľa vhodnejšia pre typ úloh a spôsob práce v podmienkach robotickej skupiny na KAI ako ROS a v porovnaním s Mikešom prináša užitočné abstrakcie, čím odstraňuje zbytočné duplicity kódu, pričom zároveň rozširuje možnosti. Najväčšími výhodami nového frameworku, ktorý zabezpečuje komunikáciu medzi modulmi budú: modularita s čo najväčšou nezávislosťou jednotlivých kompo-

nentov, jednotné rozhranie všetkých modulov, vysoká efektívnosť, platformová nezávislosť, transparentná abstrakcia nad distribuovaným multi-platformovým výpočtovým prostredím, logovanie a analýza dát a prehrávanie behov zo záznamu. Všetky časti projektu budú v podobe open-source, aby v mojej práci mohol ktokoľvek pokračovať, alebo využiť vo svojich vlastných projektoch.

Nové možnosti zapojenia hardwaru sú nevyhnutnosťou pre to, aby robotické projekty mohli napredovať a snažiť sa dosiahnuť úspech v nových disciplínach. robotické sútaže sú veľmi náročné a vyžadujú množstvo znalostí vo všetkých smeroch či už z informatiky alebo strojníctva a elektrotechniky.

V mojej práci skonštruujem robota tak, aby bol univerzálny a vedel sa nielen zúčastniť hneď niekoľkých robotických súťaží, ale aby bol použiteľný v ďalších záverečných prácach, vo výučbe, vo výskume, alebo pri verejných podujatiach a prezentáciách. V každej zo súťaží musí robot autonómne splniť zadanú úlohu bez toho, aby niečo poškodil alebo porušil dopredu dohodnuté pravidlá sútaže. Na týchto súťažiach sa zúčastňuje mnoho tímov z rôznych krajín a konajú sa vždy na nových miestach, preto sú dobrou príležitosťou na výmenu a porovnanie informácií a technológií.

V nasledujúcich kapitolách sa budem zaoberať mobilnými robotmi a ich rozdelením podľa druhu využitia, analýze existujúcich systémov, kde spomeniem doteraz používané frameworky a spomeniem aj využívanie robotov v rámci súťaží. Ďalej spomeniem ako som navrhol a skonštruoval robota, ktorý spĺňa podmienky na kvalifikáciu na súťaž a vysvetlím ako funguje môj framework. Na záver ukážem pomocou testov, že môj framework funguje.

Kapitola 1

Mobilné roboty

V tejto časti opíšem základné aspekty robotických mobilných systémov. Pri stavbe robota je potrebné poznať, analyzovať jednotlivé spôsoby a možnosti hardvérových a softvérových riešení, z ktorých pre konštruovaného robota vyberieme najvhodnejšiu kombináciu architektúry, technológií a algoritmov.

1.1 Pohon a riadenie mobilného robota

Vo všeobecnosti existuje množstvo spôsobov pohybu robotov v závislosti od vlastností rôznych prostredí: od automatických ponoriek a plachetníc, lietajúcich dronov, cez humanoidov a robotov s rôznym počtom nôh, až po autonómne automobily a roboty, ktoré sa pohybujú po chodbách v interíeri alebo aj v náročnejšom teréne v exteriéri. Zameriame sa len na kolesové roboty, ktoré sa pohybujú v rovine. [1]

Z hľadiska počtu kolies sa najviac vyskytujú roboty s dvomi, tromi alebo štyrmi kolesami. Dvojkolesové typicky vyžadujú udržiavanie rovnováhy, ktoré je v podmienkach vonkajšieho offroad prostredia príliš náročné, štvorkolesové sa buď otáčajú šmykom, čo vedie k nepresnostiam pri pohybe, alebo vyžadujú zložité riadenie ako v prípade automobilov (Ackermann steering). Hoci popularitu získavajú pohonné systémy so všesmerovými kolesami, ani tie nie sú pre naše účely ideálne, keďže vyžadujú pevný a rovný podklad, tiež majú obmedzenú presnosť a nízku rýchlosť, efektivitu a vysokú cenu. V mobilnej robotike sú asi napoužívanejšie trojkolesové roboty s dvomi hnacími a jedným voľne otočným kolesom (tzv. castor wheel). Tento druh pohonných systémov sa označuje ako diferenciálny pohon v rovine.

1.1.1 Diferenciálny pohon v rovine

Najzákladnejším typom robota, ktorý sa pohybuje v priestore, je diferenciálny pohon v rovine. Je založený na dvoch hnacích kolesách, ktoré riadia rýchlosť a smer jazdy a môžu spôsobovať zatáčanie robota. Umožňujú riadiť celkovú rýchlosť pohybu i uhlovú rýchlosť zatáčania. Jednotlivé pohyby sa dosahujú otáčaním kolies po bokoch rozličnými rýchlosťami. V prednej alebo zadnej časti robota musí byť tretie oporné koliesko, ktoré udržuje robota v rovine. Tento typ pohonu je použitý na oboch robotoch Smelý Zajko aj robot Mikeš. Tento typ pohonu bude použitý aj v novom robotovi. Ak sa hnacie kolesa pohybujú rovnako rýchlo, tak robot ide priamo dopredu alebo dozadu, ak je rýchlosť kolies rôzna, tak platí tato rovnosť

$$\frac{r_1}{r_2} = \frac{d_1}{d_2},$$

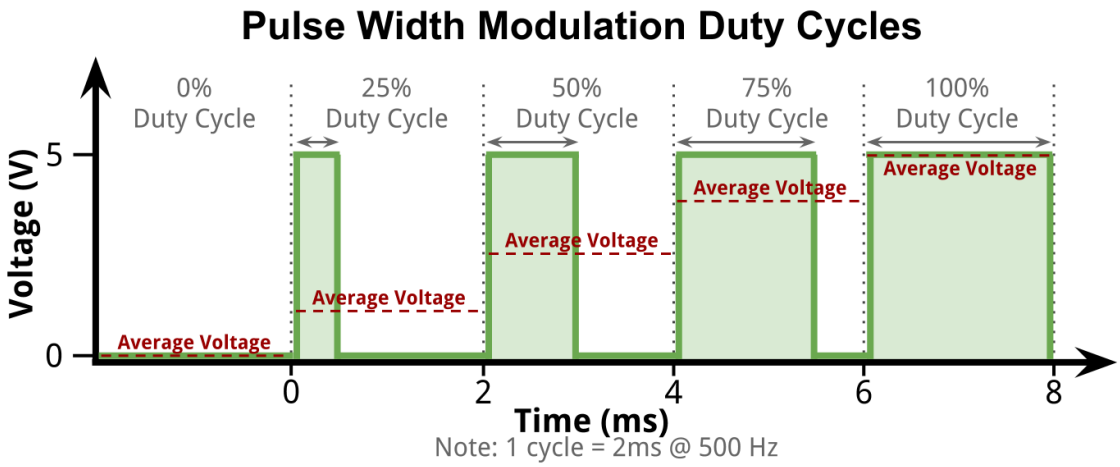
kde d_i je dĺžka po ktorej koleso prešlo a r_i je polomer oblúku, ktorý koleso prešlo.

1.2 Motory v mobilných robotoch

Aby sme mohli nazvať niečo robotom, tak sa ten robot musí hýbať, k tomu potrebuje motory a v robotike sa používa hneď niekoľko druhov a ja tu spomeniem najmä 2 najpoužívanejšie, a to sú jednosmerné motory a krokové motory.

1.2.1 Jednosmerný motor

Jednosmerný motor k pohybu vyžaduje jednosmerný prúd, ktorý pomocou prerušovaných kontaktov na komutátore (brushed DC motor), resp. pomocou externého radiča (brushless DC motor) rozdeľuje na jednotlivé cievky a sila, akou sa motor otáča, závisí od pripojeného napätia, smer otáčania závisí od polarity napätia. Keďže regulovanie napätia pri vysokom prúde vyžaduje náročné obvody, v robotike sa väčšinou používa náhrada riadením pomocou PWM (pulse width modulation) signálu (obr. 1.1). V určitej pomerne vysokej frekvencii (napr. 2000Hz) sa neustále zapína a vypína prívod napätia k motorom, pričom výsledná rýchlosť potom závisí od pomeru kladného signálu a nulového signálu, ktorý spína prívod napätia. Vďaka tomu môžeme motor zapnúť na 30 percentný výkon pomocou pulzu, kde je 30 percent periódy kladný signál a 70



Obr. 1.1: Riadenie jednosmerného motora pomocou PWM signálu. Prevzaté z: [3]

percent periódy pulzu nulový signál. Vďaka tomu je energia využívaná iba na pohyb motora.

1.2.2 Krokový motor

Krokový motor využíva tiež jednosmerné napätie, ale je skonštruovaný tak, aby všetku energiu, ktorú má k dispozícii, použil na udržanie sa v aktuálnej pozícii. Bežný krokový motor má na jednu otočku 200 pozícií, na ktorých vie držať. Medzi pozíciami sa pohybuje vďaka prepínaniu napätia medzi (typicky) 4 rôznymi cievkami (unipolar stepper motor). Napätie do prívodných 4 vodičov generuje radič servo motorov, ktorý ovládame z mikrokontroléra tak, že každý pulz dlhý približne 2 mikrosekundy signalizuje radiču posun motora o jeden krok. To znamená,že 100 percent energie sa využíva aj na to, aby stál na mieste. Sila motorov je vždy rovnaká a rýchlosť akou sa otáčajú, závisí od frekvencie pulzov za sekundu.

1.3 Senzory mobilných robotov

Neoddeliteľnou súčasťou každého autonómneho robota je schopnosť detegovať jeho okolie vďaka senzorom. Senzory vedia detegovať veľa typov údajov. Poznáme senzory zisťujúce prítomnosť prekážok v blízkom okolí vysielaním zvukových alebo svetelných vĺn ako sú napríklad ultrazvukové senzory a lidar, ďalej poznáme senzory, ktoré vedia zistiť údaje o pohybe robota v priestore vďaka akcelerácii alebo používajú iné zariadenia

s ktorými komunikujú (GPS). Pokročilejším druhom senzorov je spracovanie obrazu z jednej alebo viacerých kamier.

V tejto časti opíšem senzorické systémy, ktoré sú relevantné pre stavbu robota navrhovaného v tejto práci tak, aby vyhovoval zamýšľaným aplikáciám, napríklad účasti na súťažiach uvedených v stati 2.3.

1.3.1 Ultrazvukový senzor HC SR04

Ide o jeden z najpoužívanejších ľahko dostupných senzorov na meranie vzdialenosti. Medzi výhody patrí okrem nízkej ceny aj široký uhol v ktorom zachytáva prekážku. Nevýhodou ultrazvukových senzorov je nízka rýchlosť (rýchlosť šírenia zvuku vo vzduchu je oproti rýchlosti svetla nízka), vzájomná interferencia a zachytávanie falošných odrazov. Senzor sa riadi jednoduchým signálnym protokolom - na pine TRIG mikrokontrolér vyšle 10 μsec pulz a po dokončení merania senzor odpovedá pulzom na pine ECHO, ktorého dĺžka je úmerná nameranej vzdialenosti (distance = pulse_length/58) [6].

1.3.2 GY-87 - viacúčelový senzor s kompasom

Kombinovaný senzor obsahuje zabudovaný gyroskop a akcelerometer (MPU6050), vďaka čomu vieme korigovať hodnotu jeho zabudovaného kompasu HMC5883L podľa aktuálneho naklonenia robota. Súčiastka dokáže okrem aktuálneho linárneho a uhlového zrýchlenia merať aj teplotu a atmosferický tlak prostredia a vypočítavať momentálne natočenie vzhľadom na štartovnú polohu. Knižnica na obsluhu súčiastky obsahuje aj kód na kompenzáciu smeru kompasu podľa naklonenia [7].

1.3.3 GPS

Každý robot, ktorý sa pohybuje na dlhšie trasy vo vonkajšom prostredí by mal mať v sebe zabudovaný aj modul systému globálnej satelitnej navigácie (Global Navigation Satellite System (GNSS)). V súčastnosti používa robotická skupina 3 modely.

- Navilock NL-320U (sníma iba satelity GPS)
- Navilock NL-8022MU

- GM-3N - GNSS.

Líšia sa v presnosti a schopnosti prijímať signál, posledné dva dokážu spracovať aj signál z novších satelitov, ako je napríklad Galileo, GLONASS, či Beidou. Modul pracuje ako samostatný počítač, ktorý cez sériovú linku (pripojenú na USB port) pravidelne vysiela detekovanú pozíciu a ďalšie stavové informácie zakódované do protokolu NMEA 0183 [2].

1.3.4 ZED mini

ZED mini je zariadenie triedy RGB-D, ktoré ma v sebe zabudované 2 kamery, ktoré zaznamenávajú obraz a z neho následne vyhodnocujú dáta. Táto stereo kamera poskytuje množstvo funkcií, ako napríklad sledovanie pozície a pohybu kamery v priestore, hĺbkovú informáciu v spoločnom zornom poli kamier vo forme point cloudu alebo 2D obrázka hĺbkovej mapy. Ku kamere je dostupný SDK s implementovanými funkciami ako nahrávanie videa, detekcia plochy a veľa ďalších. Kamera využíva IMU senzor (akcelerometer a gyroskop) na zlepšenie výsledkov výpočtu a presnosti výstupných údajov. Kamera je schopná sledovať svoju pozíciu v priestore aj s natočením vďaka porovnávaniu po sebe idúcich dvojíc snímkov, ktoré sú zachytené dvoma kamerami. Vďaka tomu prepočítava svoju polohu aj natočenie v priestore. Nevýhodou zariadenia (napr. v porovnaní so zariadeniami Intel Realsense) je, že výpočet neprebieha priamo v zariadení ZED mini, ale vyžaduje spracovanie na výkonnom počítači s NVIDIA GPU. Naopak výhodou sú výborné parametre zariadenia (snímkovacia frekvencia, rozlíšenie) a najmä, že hĺbkovú i polohovú informáciu vypočítava z pasívne snímaného obrazu, čo je nevyhnutnou podmienkou vo vonkajšom prostredí, ktoré môže byť presvetlené slnečným žiarením, s čím si napr. kamera s IR projektor Intel RealSense D435 nemusí poradiť. V prípade Intel T265 zasa ide o jednoúčelové zariadenie s pasívnym stereom, ktoré ale zasa nevracia hĺbkovú mapu a neobsahuje farebnú kameru.

1.4 Riadiace počítače pre mobilné roboty

Spracovanie obrazu, alebo stereovidenia, strojové učenia a zložitejšie navigačné algoritmy si vyžadujú veľký výpočtový výkon. Mobilné roboty však majú obmedzené rozmery a nosnosť, preto vyžadujú špecializované riadiace počítače s dostatočným výko-

nom, ale malými rozmermi, hmotnosťou a spotrebou. Naopak riadiace jednotky, ktoré riadia nízke vrstvy - motory, jednoduché senzory musia fungovať v reálnom čase, ale nevyžadujú veľký výpočtový výkon. Využitie zložitejších počítačov, na ktorých beží vyšší operčný systém (napr. Ubuntu) je v tomto prípade nevhodné, keďže bežne používané operačné systémy nie sú real-time operačnými systémami a teda nie je možné zabezpečiť korektné časovanie vstupných a výstupných riadiacich signálov.

1.4.1 Jetson TX2

Jetson TX2 je malý počítač, ktorý vydáva firma Nvidia. Je určený predovšetkým na riešenie úloh, pri ktorých sa využíva umelá inteligencia. Jeho operačný systém tvorí Linux. Tento modul je použitý na Smelom Zajkovi a je na ňom spustená neurónová sieť na rozpoznávanie chodníkov z obrazu.

1.4.2 Raspberry Pi 4

V robotovi Maťo pouzijem aj počitač Raspberry Pi 4, ktorý ma 8 GB operačnej pamäte. Je to nová generácia, ktorá je oproti predchádzajúcim modelom oveľa výkonnejšia vďaka quadcore 62 bitovému procesoru Cortex-A72 [8].

1.4.3 Arduino Nano

Arduino je platforma, ktorá je veľmi rozšírená po celom svete a je základom väčšiny robotických projektov. Arduino budem používať na riadenie oboch krokových motorov robota. Arduino Nano Strong v sebe má mikroprocesor ATmega328, ktorý pracuje na frekvencii 16MHz. Arduino je určené na menšie robotické projekty, ktoré sa často využívajú na školách a aj v priemysle alebo inteligentných domácnostiach. Niektoré výstupy z Arduina vedia generovať pulzy v rôznych frekvenciách [15].

1.5 Softvérové technológie

1.5.1 Jazyk C, GCC kompilátor a make build system

Celý projekt bude realizovaný v programovacom jazyku C, pretože je to jeden z najrýchlejších jazykov aké sa dajú použiť v robotických systémoch a výpočtová rýchlosť

je prioritná pre aplikácie robotických projektov, ktoré sa musia rozhodovať v reálnom čase s minimálnou reakčnou dobou. C je najrozšírenejším jazykom pre robotiku a je kompatibilný so všetkými knižnicami, ktoré budem používať. Výhodou jazyka C je, že sa prirodzene a jednoducho integruje aj s existujúcim kódom napísaným v jazyku C++ a naopak. Program písaný v jazyku C sa najlepšie kompiluje pomocou GCC. Pomocou príkazu Make sa vytvorí súbor, ktorý sa dá spustiť. Program Make využíva súbor s pravidlami o zostavení projektu Makefile, ktorý môže byť buď jednoduchý s priamymi príkazmi na vykonanie, alebo aj zložitý, využívajúci premenné, automatické dosadzovanie hodnôt a zotavovacie pravidlá. Jeho výhodou je, že podľa dátumov modifikácii zdrojových i cieľových súborov i medzivýsledkov spustí iba tie kroky zostavenia, ktoré sú v danej situácii nevyhnutné.

1.5.2 ZED sdk

V súčasnosti používa kamera ZED mini súbor softvérových nástrojov ZED SDK 3.2 a ten ponúka knižnice a ďalšie nástroje na prácu s kamerou ZED mini. V mojej práci budem používať najmä nástroje spracovávania dát na určenie polohy robota. Program na určovanie polohy robota sa skladá z výpočtu natočenia kamery a súradníc, ktoré hovoria informáciu o polohe robota v priestore a druhá časť vykresľuje dáta.

1.5.3 Arduino IDE a súbor knižníc

Vďaka tomu, že oba motory budú ovládané z jedného Arduina, tak nemôžem použiť hardvérové budenie procesora pomocou OCR1A a OCR1B, ako tomu je pri jednosmerných motoroch, pretože ich frekvencia je vždy rovnako dlhá a rýchlosť krokových motorov sa musí líšiť vo frekvencii pulzov čo nám nedovoľuje využitie len jedného časovača. Namiesto toho budem generovať PWM pulzy softvérovo pomocou jedného časovača. Dĺžka pulzu musí byť väčšia, ako 2.5 mikrosekundy. Okrem pulzov musí Arduino generovať aj ďalší signál, ktorý udáva smer otáčania motora. Tieto dva výstupy sa posielajú do ovládacích jednotiek MB450A a tá ich pretransformuje do mikrokrokov, aby bolo otáčanie motorov plynulé a nebolo zasekané.

1.5.4 PLink - spoľahlivá sériová komunikácia v prostredí Linux z konzoly

Plink je utilita príkazového riadku, ktorá je súčasťou terminálového programu PUTTY. Slúži ako textový terminál na pripojenie napr. na sériový port. Jej výhodou je, že korektne nainicializuje sériový port v prostredí OS Linux podľa zadaných komunikačných parametrov. Dá sa využiť na ladenie komunikácie: zobrazenie údajov vysielaných z GPS alebo Arduina.

1.5.5 Knižnica kolekcií pre jazyk C - GTK Glib 2.0

Knižnica Glib obsahuje množstvo užitočných funkcií a dátových typov a niektoré z nich budem využívať v mojom programe. Využijem obojsmerný spájaný zoznam, hešovaciu tabuľku, obyčajné pole a najmä funkcie, ktoré pracujú s týmito dátovými štruktúrami. Knižnice Glib sú navrhnuté v nízkej úrovni, aby čo najviac využívali procesor a zvýšili tak efektivitu programu. Najčastejšie využívaným dátovým typom je GArray ktoré najprv treba vytvoriť pomocou `g_array_new()`, dáta sa vkladajú pomocou `g_array_append_val()` prvky poľa sa čítajú aj zapisujú pomocou `g_array_index()` [13].

Kapitola 2

Analýza existujúcich systémov a technológií

2.1 Mobilné roboty v robotickej skupine na KAI

Na katedre aplikovanej informatiky sa v súčasnosti nachádzajú dva robotické projekty.

Smelý Zajko

Prvým projektom je Smelý Zajko, ktorý vznikol z veľkej časti z diplomových prác študentov v priebehu niekoľkých rokov. Používa framework ROS, ktorý zabezpečuje komunikáciu modulov a ich správu. Na navigáciu používa lidar, ktorý zisťuje prítomnosť prekážok v ceste, otáčkové senzory na jednosmerných motoroch, GPS, kompas a aj mobilný telefón na zaznamenávanie snímkov, na ktorých rozpoznáva chodník a trávu vďaka neurónovej sieti. Účelom Smelého Zajka je zúčastniť sa na robotickej súťaži robotour kde zožal už nejeden úspech. Robot Smelý Zajko má nákladný priestor na 5L sud piva. Má pomerne silné hardvérové vybavenie - jeden až dva výkonné počítače s NVIDIA GPU - Jetson TX2, riadiaci notebook, niekoľko jednočipových mikropočítačov, kompas s kompenzáciou naklonenia, kvalitnú jednotku GPS, niekoľko ultrazvukových snímačov, kvalitný lidar značky Hokuyo a sterovidenie ZED mini.

Robot Mikeš

Ako druhý vznikol robot Mikeš, ktorý už v sebe nemal framework ROS, ale mal vlastnú verziu frameworku, ktorá sa starala o správu modulov. Robot Mikeš k navigácii používa

lidar, ktorý zisťuje prítomnosť prekážok v blízkom okolí. Robot Mikeš bol vytvorený prevažne pre súťaž SICK Robot Day, do ktorej sa pokúsil zapojiť dvakrát, raz len virtuálne, raz aj reálne. V týchto súťažiach je okrem navigácie a doručovania nákladu potrebná aj manipulácia v prostredí, takže Mikeš bol vždy vybavený aj nejakou nadstavbou, ktorá mu umožňovala aktívne interagovať s prostredím. Využíva novšiu verziu rovnakého podvozku Parallax Mount and Wheel kit s jednosmernými motormi a otáčkovými senzormi, ale na rozdiel od Smelého Zajka má skromnejší výpočtový výkon - nevyžaduje notebook ani nedisponuje zariadeniami GPU. Jeho riadenie je založené na počítači Raspberry Pi 4 a dolné vrstvy riadi jednočipový mikropočítač Arduino Nano.

2.2 Analýza frameworkov

Smelý zajko

Robot Smelý Zajko bol pôvodne naprogramovaný v C++ a program bežal na jeho jednom počítači - notebooku, ktorý si vozil so sebou. Neskôr vznikla potreba integrovania viacerých systémov a v súlade s trendami v robotických skupinách v akademickej sfére bol jeho program prepísaný do komunikačného frameworku ROS. Hlavná sila systému ROS je jeho široké využitie a dostupnosť ROS-ových balíčkov pre všetky bežné typy senzorov, aktuátorov, algoritmov a úloh. Nevýhodou však je pomerne rýchla frekvencia vydávania nových verzií, ktoré prinášajú potrebu portovania kódu pre nové verzie API, veľká závislosť na určitých verziách knižníc, či operačného systému, čo zasa koliduje s inými modulmi, ktoré využívajú najnovšie verzie balíkov CUDA, KERAS a podobne. Naviac, systém ROS používa pomerne zložitý a nepraktický zostavovací systém "catkin make", ktorý sa občasnému používateľovi vie postarať o hodiny, dni až týždne neplánovanej "zábavy". Preto bolo vyhodnotené, že pre naše potreby prináša systém ROS viac problémov a práce ako úžitku. Hoci posledná funkčná verzia frameworku na robotovi Smelý Zajko stále využíva ROS, z viacerých modulov bola závislosť na ROS odstránená, bežia samostatne. Aktuálnym cieľom je využiť výsledok tejto bakalárskej práce vo všetkých robotoch našej skupiny a preportovať ich moduly tak, aby bežali pod novým frameworkom celkom nezávislým od ROSu, ktorý vznikol v tejto práci."Základom frameworku ROS sú nezávisle bežiace moduly, pričom každý beží v samostatnom procese. Moduly produkujú dáta na kanáloch, ktorým sú priradené adresy v globálnom

mennom priestore. Jednotlivé moduly sa vďaka týmto globálnym adresám zdrojov dát dostávajú k jednolivým dátam. Veľkou výhodou je, že ROS funguje v distribuovanom prostredí a automaticky preposiela údaje medzi rôznymi výpočtovými uzlami, ak moduly bežia na rôznych počítačoch. [10] [5] [4] [11] [12].

Robot mikes

Robot Mikeš nepoužíva ROS aby bolo jeho používanie jednoduchšie. Moduly si pamätajú svoje posledné aktuálne dáta, tie ukladajú do premennej, ktorá slúži len na kopírovanie týchto dát každému modulu, ktorý ich chce získať. Každý modul si musí pamätať komu ma posielať posledné aktualizované dáta. Tieto dáta sa vždy musia kopírovať z posledných validných dát. Každý modul ma zadefinované či je aktívny alebo pasívny, teda či má alebo nemá vlastne vlákno. Každý aktívny modul musí doručiť dáta všetkým subscriberom, ktorí sú prihlásení na odber dát. Pasívne moduly spracovávajú dáta priamo vo vlákne modulu, ktorý odosiela dáta. V tomto návrhu sa musia dáta stále kopírovať čo spomaľuje celý systém. [18] [9]

Janko Hraško

Robot má samostatný pohon všetkých štyroch kolies, je vybavený robotickým ramenom a starším typom stereovidenia, ktoré sa dá použiť na rozpoznávanie statickej scény. Robota má radič s jednočipovým mikropočítačom a miniaturizovaným počítačom Gumstix Overo. Bol využitý v diplomovej práci [16], kde jeho úlohou bolo vyhľadávať a zbierať červené kocky, vykladať kocky rôznej farby na seba, alebo riešiť plánovacie manipulačné úlohy.

2.2.1 ROS

Podobne distribuované systémy na riadenie robotov používajú vo veľkej miere v každom odvetví robotiky. Robotika je veľmi rozšírená najmä v priemysle, kde je lacnejšou a efektívnou pracovnou silou a prináša veľa pozitívnych vlastnosti. Ukážkovým príkladom je ROS pretože v priemysle sa používa jeho variant ROS-industrial, ktorý ponúka viac nástrojov a knižníc, ktoré ponúkajú možnosti vytvoriť si model robotov, následne s nimi pracovať a implementovať nove metódy a postupy. Aj tie najväčšie spoločnosti, ktoré vyrábajú robotické ramená, ako je ABB alebo Fanuc podporujú ROS. ROS bol

navrhnutý v roku 2007 na univerzite študentami, ktorí chceli docieliť spoluprácu a priblížiť robotiku širšiemu okruhu ľudí. Všetky knižnice, ktoré ROS ponúka, sú vytvorené expertmi v danej oblasti a sú vďaka ROSu používané kýmkoľvek bez toho, aby musel vedieť čo dané knižnice robia. Hoci ROS je veľmi silný nástroj a ponúka veľké množstvo hotových modulov, dôkladné zoznámenie sa s frameworkom (ktoré je nevyhnutné, ak má byť výsledný program správny) si vyžaduje viac času, ako je v typických záverečných prácach k dispozícii. Ďalšou nevýhodou ROS je, že je závislý na verzii operačného systému Ubuntu, ktorá sa mení každé dva roky [14].

2.3 robotické súťaže

Jednou z hlavných motivácií pre stavbu nového robota je, aby sa mohol zapojiť (aspoň postupne) do viacerých robotických súťaží. Východiskom sú skúsenosti, ktoré naša robotická skupina za 10 rokov získala s ostatnými robotmi - mal by mať robustnejšiu a presnejšiu konštrukciu, väčšie kolesá a vyšší podvozok a to by mu malo umožniť pohybovať sa aj v náročnejšom teréne mimo cesty a chodníky. V tejto stati uvádzame hlavné súťaže, nad ktorými sme uvažovali pri zbieraní predpokladov pri stavbe robota.

2.3.1 Robotour

Súťaž Robotour je súťaž, v ktorej sa robot pohybuje po chodníkoch či už v parku alebo na ulici. Zmyslom súťaže je odštartovať robota, ktorý sa bude autonómne pohybovať v priestoroch súťaže pričom sa má previesť náklad medzi dvomi stanoviskami. Na tejto súťaži je, ako náklad použitý 5 litrový sud piva. Aby robot splnil úlohu, musí sa autonómne navigovať v priestore a chodiť len po chodníku na prvé stanovisko kde zastane a dá obsluhe signál, že sa môže nakladať náklad. Po naložení nakladu musí sám detegovať, že má sud naložený a následne sa musí dostať na druhé stanovisko. Keď sa tam dostane, tak vydá zvukový signál, aby obsluha vedela, že môže vyložiť sud. Robot potom deteguje, že už nemá naložený náklad a vyberie sa naspäť na štart. Celá súťaž má 5 kôl a z toho prvé kolo je len cvičné.

2.3.2 Roboorienteering

V súťaží Roboorienteering je úlohou robota aby rozniesol 5 golfových loptičiek na stanoviská rozmiestnené v areáli s ľahším terénom v akomkoľvek poradí. Stanovísk je viac, ako golfových loptičiek a môže si vybrať do ktorých ich odnesie. Lietajúce roboty majú namiesto golfových loptičiek pingpongové loptičky. Súťaž má definované štartovacie miesto odkiaľ robot vyštartuje a aj ukončovacie miesto kam musí robot prísť. Táto súťaž má 4 kolá a prvé kolo je len cvičné. Na tejto súťaží sa naša robotická skupina ešte nezúčastnila.

2.3.3 Robocup@home

Robotická súťaž RoboCup@Home spočíva v tom, aby sa robot autonómne pohyboval v domácom prostredí, kde musí pomáhať ľudom s bežnými úlohami, ako je varenie, manipulácia s menšími objektami, interakcia s ľuďmi, vyhľadávanie objektu v priestore a mapovanie prostredia a nesmie pritom nič poškodiť ani nikoho ohroziť. Táto súťaž je spomedzi vyššie spomenutých najnáročnejšia a vyžaduje veľké množstvo zdrojov a ľudí čo náš tím zatiaľ nemá, ale robot je zostrojený s výhľadom na možné vylepšenia do budúcnosti, aby sa prípadne aj takejto súťaže dokázal zúčastniť.

2.4 Porovnanie výpočtu pozície zo stereo videnia, odometrie a GPS

Aby sa robot navigoval v priestore potrebuje vedieť čo najpresnejšie kde sa nachádza a akým smerom ide. Každý druh zapísaných dát má určitú odchýlku. Kamera ZED mini vie dáta udávať s presnosťou na milimeter zatiaľ čo dáta z modulu GPS majú presnosť niekoľko metrov. Aby robot vedel kde sa nachádza, musí na základe týchto dát určiť ktorým smerom sa pohol a o akú vzdialenosť. Pred narhovaním robota som vykonal analýzu schopností polohovacích zariadení a algoritmov, aby bolo možné vybrať vhodné komponenty pre výsledný návrh.

2.4.1 Definícia problému

Aby boli údaje použiteľné na výpočet aktuálnej polohy robota musia byt preškálované na rovnakú jednotku aj mierku. Dáta z kamery ZED mini sú naškálované na metre, dáta z motorov sú v pulzoch a dáta z GPS sú zapísané, ako zemepisná šírka a zemepisná dĺžka. Všetky dáta musím previesť na milimetre a následne ich použiť pri vykresľovaní pozície jednotlivých modulov.

2.4.2 Postup

Dáta z kamery ZED mini prevediem bez problémov z metrov na milimetre. Údaje o motoroch sú náročnejšie, pretože musím zohľadniť ich rýchlosť v pulzoch za sekundu a prevody motora a mikrokrokovanie motora. Motor má 200 pulzov na jedno otočenie hriadeľa motora, na jednu otočku hriadeľa motora musí do ovládačov prísť 400 pulzov, pretože je nastavené mikrokrokovanie na 400, a vzhľadom na prevody prevodovky je potrebných 15 otočení hriadeľa motora na jedno otočenia kolesa takže 400*15, a to je 6000 pulzov na otočenie kolesa o jednu obrátku. Obvod kolesa sa počíta pomocou 2*r*pi pričom r je 15,24 a obvod kolesa je teda 95,707 centimetra, a to je 957,07 milimetra na jedno otočenie kolesa. Súradnice GPS musím najprv naškálovať, aby bol v kilometroch jeden stupeň zemepisnej šírky rovnako dlhý ako jeden stupeň dĺžky a potom ich musím previesť na milimetre pomocou vzorca.

2.4.3 Výsledky

Výsledné dáta zobrazujem pomocou programu napísanom v Pythone na grafickú plochu. Zelenou farbou je zakreslená dráha kamery ZED mini, modrou čiarou sú zakreslené motory a červenou farbou sú zakreslené dáta z GPS. Z dát jasne vidieť, že dáta z GPS sa nedajú použiť na určenie dráhy robota, pretože majú príliš veľkú odchýlku, zatiaľ čo z dát je vidieť, že presnosť kamery ZED mini a presnosť motorov má medzi sebou len drobnú odchýlku, ale sú relatívne presné a takmer rovnaké (obr. 2.1). Odchýľka na teste zobrazenom vľavo je spôsobená kopcovitým terénom, v ktorom kolesá prešli dlhšiu dráhu (šikmá prepona) ako je posun zaznamenaný pomocou kamery ZED, keďže v obrázku sú zobrazené len súradnice x, y, hoci ZED pochopiteľne zaznamenal aj zmenu nadmorskej výšky. Aj z tohto príkladu vidno, že využitie otáčkových senzorov aj v prí-

Obr. 2.1: Porovnanie záznamov zaznamenaných senzormi.

pade, že sú presné, vedie k chybám v navigácií v zvlnenom teréne - pokiaľ sa neberie do úvahy aj naklonenie robota zmerateľné pomocou akcelerometra. Na obrázku vpravo zasa vidno, že kamera a motory sú umiestnené na inom mieste na robotovi (posun je skoro pol metra), čo znamená, že sa pri zatáčaní pohybujú po kružniciach s inými polomermi. V tomto experimente sa nám teda potvrdilo, že najlepšie polohovanie môžeme dosiahnuť pomocou výpočtu polohy zo stereovidenia kombinovaného s akcelerometrom a gyroskopom - za podmienky, že tento signál bude dlhodobo spoľahlivý. Na to bude treba vykonať viacero testov s už hotovým robotom.

Kapitola 3

Návrh robota Maťo

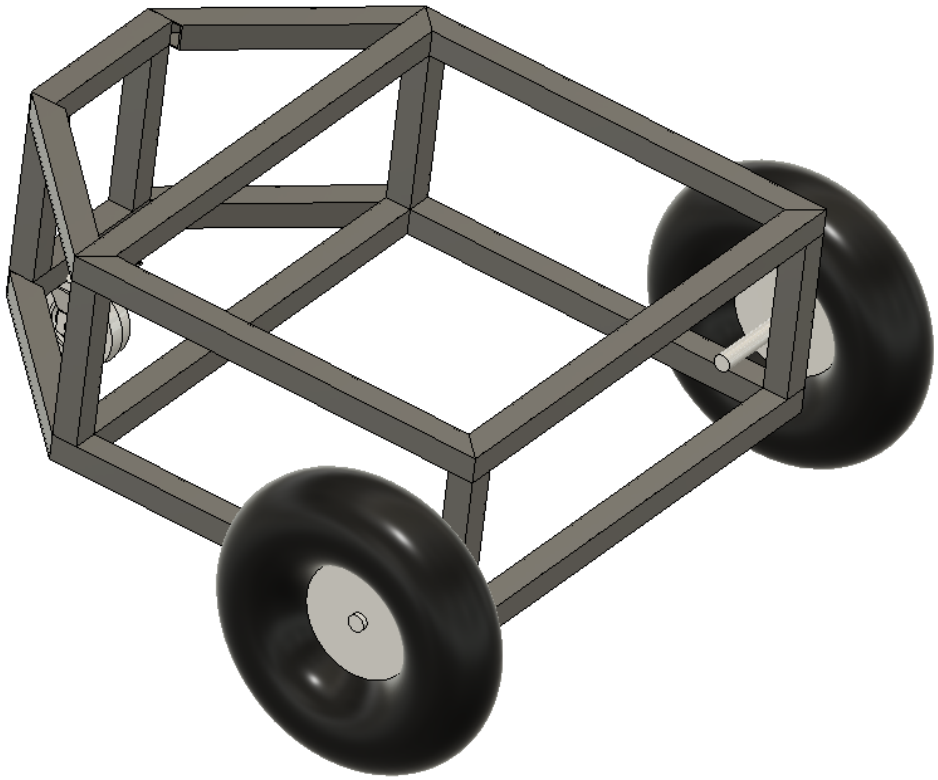
V tejto časti práce sa budem venovať návrhu robota Maťo.

3.1 Bezpečnosť

Robot Maťo musí obsahovať prvky bezpečnosti ,aby sa pri súťaží alebo iných aktivitách nikomu nič nestalo. V prednej časti má robot umiestnené ultrazvuky, ktoré detegujú prekážky na kratšie vzdialenosti vysielaním ultrazvukových signálov, ktoré sa odrazia od prekážky a vrátia sa o nejaký čas neskôr a tento čas vypovedá o vzdialenosti robota od prekážky. Ďalším dôležitým prvkom bezpečnosti je núdzové tlačidlo na zastavenie, ktoré je pripojene do Arduino a pri jeho stlačení sa robot zastaví v čo možno najkratšom čase. Stlačenie tlačidla robota zastaví, ale po opätovnom uvoľnení tlačidla sa robot môže pohybovať ďalej bez straty jeho pozície.

3.2 Konštrukcia robota

Konštrukcia robota vždy zavisí od účelu využitia robota. Keďže účelom robota Maťo bude zúčastniť sa robotickej sútaže roboorienteering, tak aj jeho konštrukcia tomu musí zodpovedať. Robota som skonštruoval tak, aby bol ľahký vďaka použitiu hliníkových profilov s vysokou pevnosťou a nízkou váhou, ktoré som zvaril pomocou zváracej metódy TIG do jednej pevnej konštrukcie, na ktorú budu uchytené zvyšné komponenty. Tuhosť konštrukcie je prispôsobená terénu, pretože robot Maťo bude jazdiť po ľahšom teréne čo zahŕňa napríklad trávnatú plochu. Ťažisko robota je sústredené do prednej

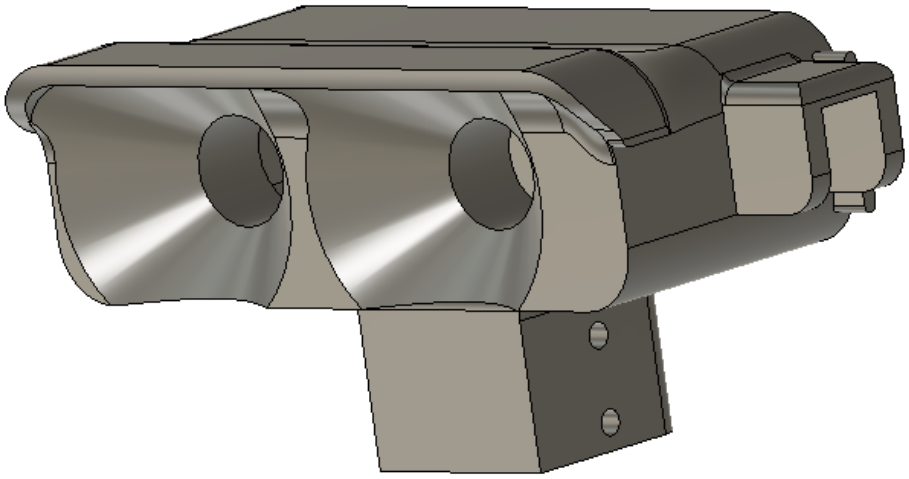


Obr. 3.1: 3D model vytvorený vo Fusion 360 znázorňuje konštrukciu a umiestnenie kolies robota.

časti, aby mali hnacie kolesá čo najväčší záber. Robot obsahuje 2 hnacie kolesá s priemerom 30,5 centimetra a jedno oporné vše smerové koliesko s priemerom 8 centimetrov. Robota som vymodeloval v 3d modelovacom programe Fusion 360. Je navrhnutý tak, aby som naňho jednoducho vedel upevniť plexisklo a tým ochrániť komponenty pred vodou, ktorá sa na súťaži môže vyskytovať v podobe dažďa alebo v podobe drobných kvapiek zachytených na tráve. Konštrukcia je navrhnutá na jednoduché vylepšovanie ak by bolo potrebné pripevniť robotické rameno (obr. 3.1).

3.3 Hardvér použitý na robotovi Maťo

Pohon robota Maťo budú tvoriť dva krokové motory Nema23, ktoré majú 200 krokov na otáčku a vyžadujú prúd 3A. K nim sú pripojene ovládače krokových motorov MB450A, ktoré spracovávajú signál z Arduina. Krokové motory sú zapojené do prevodovky s



Obr. 3.2: 3D model vytvorený vo Fusion 360 ktorý chráni kameru pred vodou.

pomerom 1:15, aby mal robot menšiu rýchlosť a zároveň väčšiu silu pri jazde do kopca alebo cez prekážky. Otáčkové senzory pri krokových motoroch nie sú potrebné, pretože motory udržujú pozíciu presne takú, akú mu program pošle a otočia sa presne o toľko krokov, koľko im program pošle. Robot využíva ultrazvukové senzory v prednej a v zadnej časti na detegovnie prekážok pri jazde vpred alebo vzad. Ďalší ultrazvukový senzor je využitý na sledovanie, či je náklad naložený alebo nie. Kamera ZED mini je schovaná v ochrannom obale, ktorý som navrhol špeciálne na toto použitie, vďaka čomu sa do kamery nedostane voda ani pri silnom daždi a zachováva plný zorný uhol pre obe kamery. Tento obal bol navrhnutý a vymodelovaný v programe Fusion 360 a následne bol vytlačený na 3d tlačiarni (obr. 3.2).

3.4 Špecifikácia frameworku pre nového robota

Hlavným cieľom je jednoduchosť a znovu použiteľnosť frameworku aj pri ďalších robotických projektoch. Framework zabezpečuje spoluprácu modulov a prenosu dát medzi nimi. Framework komunikuje prostredníctvom TCP/IP protokolu, moduly komunikujú so zariadeniami aj cez sériovú linku. Framework vie kedykoľvek pridať nový modul

alebo ho odstrániť. Framework registruje každý modul a vie kedykoľvek poskytnúť ich zoznam. Každý modul môže bežať na rôznej platforme a budú spolu komunikovať prostredníctvom frameworku, ako keby boli na tom istom zariadení.

3.5 Navrh API frameworku

Všetky dáta pochádzajú z kanálov modulov. Každý modul má minimálne jeden kanál cez ktorý posiela dáta. Tieto dáta sa dostávajú postupne do frameworku volaním funkcie `mato_post_data()` ktorá ich odovzdá frameworku a hneď sa vráti naspäť do modulu. Všetky dáta sú posielané v bufferoch konkrétnej veľkosti, ktoré sú alokované vo frameworku a modul si ich musí vypýtať pred každým odosielaním dát. Framework spravuje tieto buffre a keď už nie sú dáta potrebné, tak ich sám deallocuje.

3.5.1 Prenos dát

Prenos dát na vyžiadanie

Modul zavolá framework a vypýta si od neho posledné aktuálne dáta z daného modulu a daného kanálu. Framework pošle poslednú platnú verziu dát, ktoré uchováva vo forme dát alebo smerníku.

`mato_get_data()`

Ak modul požiada druhý modul o dáta pomocou `mato_get_data()` tak framework odošle žiadateľovi posledné validné dáta, ktoré v sebe má uložené. Ak framework neuchováva žiadne uložené dáta, ktoré by mohol poslať, tak odošle NULL.

`mato_borrow_data()`

Ak modul požiada druhý modul o dáta pomocou `mato_borrow_data()` tak framework odošle žiadateľovi pointer na posledné validné dáta, ktoré v sebe má uložené. Ak framework neuchováva žiadne uložené dáta, ktoré by mohol poslať, tak odošle NULL.

Sledovanie všetkých dát konkrétneho modulu na danom kanále

Každe dáta, ktoré vytvorí modul na kanáli sú automaticky odoslané všetkým modulom, ktoré sú prihlásené na odoberanie dát a framework ich všetkým rozpošle. Framework v sebe udržuje všetky dáta ktoré sú aktuálne alebo ešte neprečítané.

3.5.2 Tri spôsoby posielania dát

Všetky typy prijímania dát musia pracovať rýchlo, pretože každý prijímateľ musí počkať pokiaľ jeho predchodca číta dáta. Všetky dáta na ktoré má niekto referenciu pomocou pointeru si framework uchováva a počíta koľko modulov má dané dáta požičané. Akonáhle je počet referencii nulový, tak framework vie, že už niekto nepoužíva dáta a vymaže ich. Každe dáta držané vo frameworku začínajú s počtom referencii 1, aby framework nevymazal dáta skôr, ako sa rozpošlú a vrátia od všetkých prijímateľov dát. Každý modul, ktorý posiela dáta musí byť nastavený tak, aby neposielal dáta rýchlejšie, ako je priemerná doba počas ktorej všetky callbacky prijímateľov spracujú dáta od odosielateľa.

Priame dáta

Priame dáta sú odoslané bufferom do frameworku a ten ich rozposiela všetkým modulom, ktoré ich chcú prečítať. Dáta sú určené len na čítanie a akonáhle si ich raz prečitajú a skončí ich callback, tak k nim už nemajú pristúp. Dáta sa udržiavajú len pokiaľ neskončia callbacky všetkých modulov a potom dáta zaniknú.

Kópia dát

Každý modul, ktorý sleduje dáta prichádzajúce z iného modulu v takomto type framework vytvorí kópiu tých dát a odošle ju danému modulu. Po spracovaní všetkých dát sa modul musí postarať o deallocovanie pamäte buffera.

Pointer na dáta

Vždy keď modul prijíma dáta od iného modulu v type pointera na dáta tak dostane pointer na dáta od frameworku, ktoré sú určene len na čítanie a tieto dáta mu ostanú aj po tom, ako skonči jeho callback. Tento pointer v sebe uchováva až dokiaľ ho neuvoľni

a nepošle ho naspať do frameworku. Ak viacero modulov odoberá dáta na tom istom kanáli takýmto spôosobom, všetky moduly zdieľajú tú istú kópiu dát. Framework dáta automaticky uvoľní z pamäti potom, keď už ich už žiaden modul nepoužíva, čiže všetky moduly poitner vrátili frameworku volaním `mato_release_data()`.

Framework poskytuje funkciu na diagnostiku, ktorá poskytuje informácie o počte alokovaných bufferov na dáta pre daný kanál a počet všetkých referencií, ktoré referujú na daný dátový buffer. Všetky posielané buffre obsahujú len dáta, ich veľkosť sa posiela v samostatnej sprave.

3.5.3 Modul

Aktívne moduly

Aktívne moduly majú svoje vlastné vlákno. Väčšinou kopírujú prijaté dáta alebo pointer na dáta do ich lokálnych štruktúr.

Pasívne moduly

Pasívne moduly nemajú svoje vlákno a sú spravidla veľmi rýchle a vykonané priamo v callbacku.

3.5.4 Návrh modulu

Všetky moduly budu musieť spĺňať zakladnú štruktúru.

Základné údaje modulu

Každý modul musí mať 2 základne údaje, a to je reťazec znakov name, ktorý obsahuje meno daného modulu a musí byť unikátne, a reťazec znakov type, ktorý hovorí o type daného modulu. Framework si ich uloží a priradí im spoločný ID. Toto ID je uložené v prevodnej tabuľke.

Inicializácia modulu

Modul obsahuje `XXX_init()` pomocou ktorého sa musí zaregistrovať vo frameworku a musí vytvoriť všetky potrebné údaje, ktoré sú spoločné pre všetky inštancie daného modulu a daného typu.

Inicializácia inštancie modulu

Modul obsahuje `XXX_create_instance(module_id) - instance_data`, ktorý zavolá framework a ten modulu vráti dátovú štruktúru obsahujúcu dáta danej inštancie.

Štart modulu

Všetky moduly obsahujú funkciu `XXX_start(instance_data)`, ktorá je spustená naraz pre všetky moduly frameworkom a ak je nejaký modul pridaný po štarte, tak ho framework pomocou tejto metódy odštartuje.

Vymazanie modulu

Modul musí obsahovať funkciu `XXX_delete_instance(instance_data)`, ktorá deblokuje všetky dáta uložene v pamäti.

3.5.5 Návrh frameworku

`mato_register_new_type_of_module(type, module_specification)`každý nový modul zavolá túto triedu zo svojej funkcie init(), pomocou nej sa do tabuľky sa pridá špecifikácia daného typu modulu.

`mato_create_new_module_instance(type, name)` funkcia vytvorí všetky potrebné dátové štruktúri pre inštanciu a priradí jej id.

`mato_delete_module_instance(module_id)` funkcia vymaže všetky dáta a referencie na inštanciu a všetky jej kanály.

`mato_start()` odštartuje všetky inštancie spustením funkcie `mato_start_module` ktorá pre každé id modulu zavolá funkciu, ktorá je súčasťou každeho modulu.

`mato_start_module(id_module)` spustí funkciu, ktorá odštartuje modul s daným id.

`mato_get_module_id(name)` funkcia vráti id modulu, ktoré používa zadané id.

`mato_get_data(id_module, channel)` -> vráti posledné platné dáta modulu na danom kanáli.

`mato_subscribe(subscriber_module_id, subscribed_module_id, channel, callback, int subscription_type)` funkcia vytvorí na pozícii so zadaným id modulu a jeho kanálom novú štruktúru obsahujúcu údaje o odoberani dát ktorú vráti.

`mato_unsubscribe(id_subscription)` funkcia vymaže dáta pomocou ktorých bol modul prihlásený na odber dát.

`mato_get_data_buffer(size)` framework vráti modulu buffer na nové dáta s požadovanou veľkosťou.

`mato_post_data(id_of_posting_module, channel, data_length, data)` funkcia pošle pomocou pipe nové dáta.

`mato_release_data(id_module, id_channel, data)` funkcia prehľadá všetky dáta, ktoré framework uchováva pre daný modul na danom kanále a zníži referenciu o používaní týchto dát. Ak dáta už nikto nepoužíva tak ich vymaže.

`mato_get_module_instance_data(id_module)` funkcia vráti dáta o danej inštancii.

`mato_get_list_of_all_modules()` funkcia vráti zoznam modulov uložený v poli typu GArray.

`mato_get_list_of_modules(type)` funkcia vráti zoznam modulov daného uložený v poli typu GArray.

`mato_send_global_message(sender_module_id, message_id, data_length, data)` funkcia pošle každému modulu správu.

`mato_data_buffer_usage(id_module, id_channel, *number_of_allocated_buffers, *total_sum_of_ref_count)` funkcia slúži na diagnostiku a vracia údaje o počte allocovaných buffrov na dáta a počet koľko modulov používa tie dáta.

3.5.6 Návrh distribuovania dát vo frameworku

Úlohou frameworku je zabezpečiť komunikáciu pre modely nielen na jednom zariadení, ale aj prostredníctvom viacerých zariadení zapojených v sieti. Žiadny modul, ktorý komunikuje cez takúto sieť nebude vedieť, že druhý modul je na inom zariadení vďaka využívaniu frameworku. Uzly musia byť zapojené tak, aby vypojenie niektorého uzla neznemožnilo komunikáciu ostatných uzlov. Na začiatku sa každý uzol spojí so všetkými ostatnými. Po pridaní nového modulu sa aktualizujú zoznamy vo všetkých uzloch, aby všetky uzly vedeli ktoré moduly sú pripojené do siete. Hoci framework v čase odovzdania práce funguje len v rámci jedného výpočtového uzla, jeho distribuovaná verzia je rozpracovaná, navrhnutá a je predpoklad, že bude čoskoro úspešne implementovaná.

`mato_get_data()`

Každý modul, ktorý požaduje dáta od modulu, ktorý nie je na lokálnom zariadení zavolá `mato_get_data()` ako by boli na tom istom zariadení a framework sám zistí, kde sa požadovaný modul nachádza. Ak sa nachádza na zariadení, tak mu pošle dáta a ak je na inom zariadení, tak z funkcie `mato_get_data()` zavolá ďalšiu funkciu, ktorá pošle požiadavku na framework v inom uzle, ktorý ju spracuje a pošle dáta cez pakety naspät. Lokálny framework následne pošle dáta modulu, ktorý ich chcel vidieť. Tieto dáta sa vždy pošlú ako kópia dát z jedného frameworku na druhý.

Subscribe

Prihlásenie sa na odber dát cez sieť bude realizovaný pomocou funkcie `mato_subscribe` ktorá zistí či je modul od ktorého sú požadované dáta na lokálnom zariadení a ak nie, tak sa framework prihlási na odber požadovaných dát z druhého frameworku. Všetky dáta, ktoré vzniknú na druhom frameworku sa automaticky posielajú na lokálny framework, ktorý ich rozpošle všetkým modulom, ktoré požadujú dané dáta.

3.6 Dátové štruktúry frameworku

3.6.1 Dátové štruktúry

subscription

subscription je štrúktura, ktorá udržuje informácie o odoberateľoch dát. Skladá sa z unikátneho subscription id, ďalej uchováva informáciu o tom akého typu je daný odoberateľ dát. Má uloženú aj funkciu, ktorá bude zavolaná a na koniec aj id samotného odoberateľa.

`channel_data`

`channel_data` v sebe drží údaje o module a kanále, z ktorého pochádzajú dáta a má uloženú aj dĺžku dát a smerník na uložené dáta a ukladá si aj počet referencií na dané dáta.

modul_names	
modul_id	Name
0	Name1
1	Name2

modul_types	
modul_id	Type
0	Type1
1	Type2

Obr. 3.3: Dátová štruktúra znázornuje vzťah medzi module id a údajmi modulu.

module info

Táto štruktúra slúži ako pomocná štruktúra pri vypisovaní dát o module a skladá sa z module id, názovu a typu daného modulu.

`module_specification`

Táto štruktúra sa skladá zo smerníkov na funkcie, ktoré vedia inštanciu vytvoriť, vymazať, odštartovať a poslať globálnu správu všetkým zapojeným modulom. Posledný údaj tejto dátovej štruktúry je, koľko má daná inštancia kanálov.

`module_instance_data`

Táto štruktúra sa nachádza priamo v moduloch a obsahuje id, ďalej id modulu a id modulu, od ktorého chce prijímať dáta.

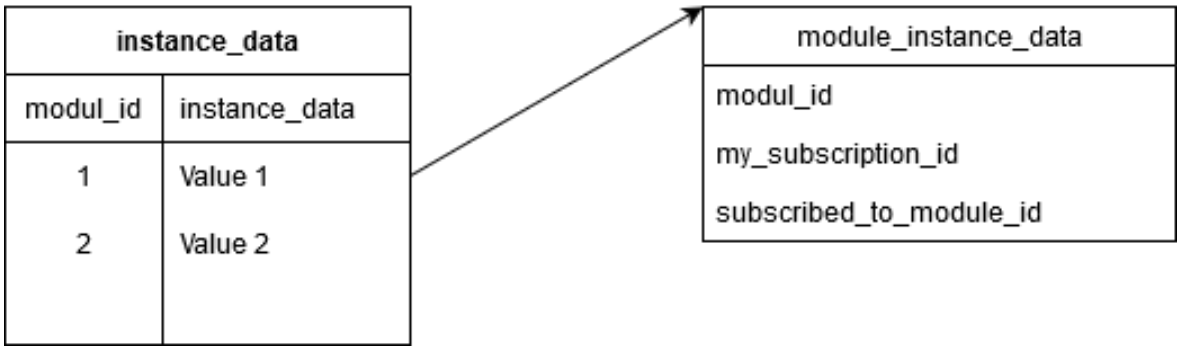
3.6.2 Spôsob ukladania dát

`module_names`

`module_names` je pole typu GArray, ktoré má na indexe `module_id` uložené meno modulu (obr. 3.3).

`module_types`

`module_types` je pole typu GArray, ktoré má na indexe `module_id` ulozený typ modulu (obr. 3.3).



Obr. 3.4: Dátová štruktúra znázornuje príklad ako môžu byť uložené dáta inštancie.

`instance_data`

`instance_data` je pole typu GArray, ktoré obsahuje dátové štruktúry `module_instance_data` o inštancii podľa `module_id`(obr. 3.4).

`modul_specifications`

`modul_specifications` je hešovacia tabuľka, v ktorej sú uložené špecifikácie modulov a ako kľúč sa používa reťazcová hodnota typu modulu.

`buffers`

Premenná `buffers` je vytvorená pomocou pola typu GArray a ako index sa používa `module_id` a každé políčko obsahuje ďalšie pole GArray, ktorého index tvoria kanali daného modulu a na tom mieste sa nachádza obojsmerný spájaný zoznam, ktorý pozostáva zo štruktúr `channel_data`.

`subscriptions`

V `subscriptions` je uložené pole, v ktorom sa pomocou `module_id` ako indexu nachádza ďalšie pole, ktorého indexi tvoria `channel_id` a v každom políčku s indexom `channel_id` sa nachádza pole, ktoré obsahuje všetkých odoberateľov dát od modulu ktorého id je `module_id`.

Kapitola 4

Implementácia a testovanie

4.1 Testy

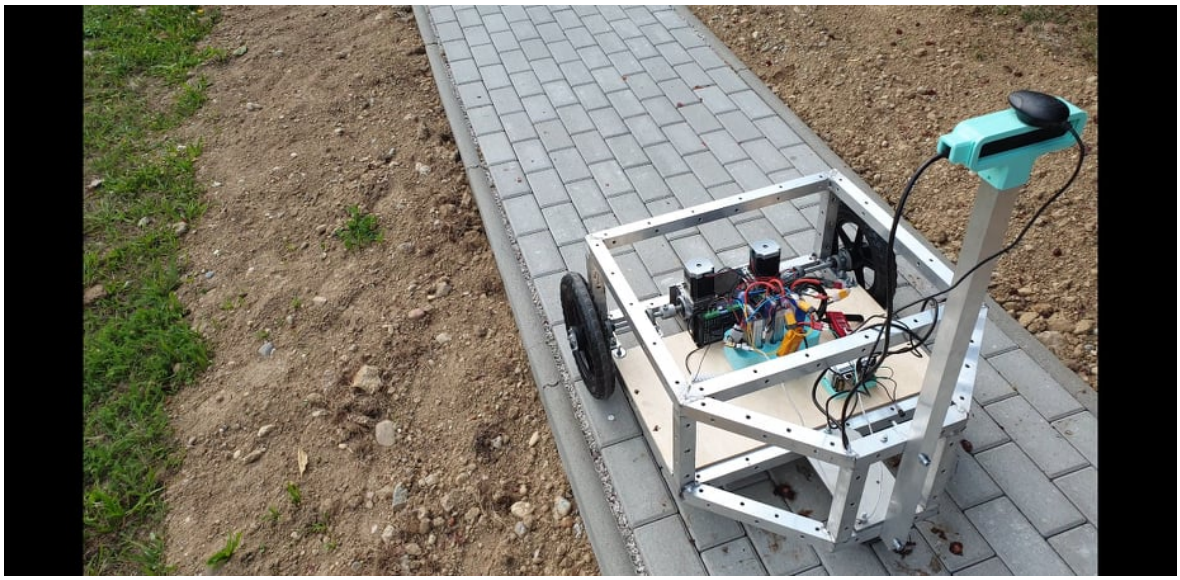
Funkčnosť frameworku testujem využitím testov, ktoré sú navrhnuté od jednoduchších po zložitejšie.

Test dvoch inštancií modulu

V tomto teste testujem registráciu modulu a vytvorenie dvoch inštancií toho istého modulu. Ďalej testujem, či sú správne zapísané v zozname modulov pomocou funkcie `mato_get_list_of_modules()` a `mato_get_list_of_all_modules()`. Pomocou cyklu testujem odosielanie správ v podobe indexov cyklu pomocou `mato_post_data()` a ich následné spracovanie frameworkom a odoslanie dát všetkým odoberatelom dát od konkrétneho modulu.

Test dvoch modulov

Druhý test obsahuje dva moduly a každý modul vytvorí dve nové inštancie. Opäť testujem, či sú správne zapísané v zozname modulov pomocou `mato_get_list_of_modules()` `mato_get_list_of_all_modules()`. Testujem aj prijímanie dát, ktoré si medzi sebou moduly navzájom posielajú. Vytvorené inštancie sa na seba navzájom prihlásia na odber dát a v cykle posielaju dáta v podobe číselných hodnôt. Každú správu, ktorú dostane modul typu A spracuje a pripočíta k nej hodnotu 100, každú správu ktorú spracuje modul typu B k prijatej hodnote pripočíta hodnotu 1000. Každá inštancia odošle každú spracovanú hodnotu `mato_post_data()` do frameworku a ten ju rozpošle



Obr. 4.1: Robot zachytený počas testovacej jazdy

všetkým inštanciám, ktoré sú prihlasené na odber tých dát. Výnimku tvorí len jeden modul typu B, ktorý už ďalej neposiela dáta frameworku, aby nevznikol cyklus.

4.2 Výsledky

Konštrukcia robota Maťo (obr. 4.1) zodpovedá návrhu a má drobné nedostatky v miestach zvarov, ktoré ale neovplyvňujú funkčnosť robota. Tvar robota a jeho celkové prevedenie nepotrebovalo žiadne softvérové ladenie na to, aby išiel rovno vďaka presne namontovaným oskám kolies. Výkon motorov zodpovedá očakávaniam a sú primerane silné vzhľadom na ich účel. Na strmých trávnatých terénoch nemá dezén kolies dostatočnú trakciu a prešmikujú kolesa, čo spôsobuje že sa robot nevie dostať cez strmé trávnaté kopce. Vybrácie, ktoré sa prenášajú rámom robota na kameru ZED mini nemá veľký vplyv a po celú dobu testovania sa ani raz nestalo že by kamera stratila svoju pozíciu v priestore vďaka priveľkým otrasom. Podarilo sa mi vytvoriť funkčný framework, ktorý ma potenciál v budúcnosti nahradiť riadiacu architektúru v ostatných robotických projektoch v robotickej skupine KAI. Pomocou frameworku sa robot Maťo úspešne zaregistroval na súťaž Robotour a dokáže sa sám pohybovať, reagovať na prekážky zastavením, regulovať rýchlosť a je pripravený na integráciu ďalších modulov, ktoré sú použité na iných robotoch. Kamera ZED mini bola overená a je vhodná na používanie v robotických projektoch.

Kapitola 5

Záver

V tejto práci som vysvetlil čo sú mobilné roboty a na čo sa používajú. Vysvetlil som, akým spôsobom prebiehajú robotické súťaže a ich pravidlá a na ktorej súťaži sa robot Maťo zúčastní. Ako fungujú jednotlivé časti robotov ako je pohon, senzory, hardvérové a softvérové prvky. Vysvetlil som, ktoré komponenty použijem pri stavbe nového robota Maťo a podrobne som ukázal, ako som ho zostrojil od návrhu v 3d modelovacom programe až po hotového robota, ktorého som prihlásil na robotickú súťaž Robotour. Ďalej som vysvetlil návrh nového frameworku, ktorý by mal časom nahradiť doterajšie frameworky v ostatných robotoch. Zaznamenané údaje zo senzorov z jázd som porovnal a ukázal nepresnosti senzorov. API nového frameworku bolo zdokumentované pomocou systému Doxygen a bola k nemu vytvorená dokumentácia, ktorá v budúcnosti pomôže vývojarom. Projekt sa nachádza na github a je voľne dostupný v podobe open source.
`https://github.com/Robotics-DAI-FMFI-UK/mato-common`
`https://github.com/Robotics-DAI-FMFI-UK/mato-generic`

5.1 Budúce vylepšenia

Robot Maťo funguje z velkej časti lepšie ako jeho predchodcovia, ale stále je na ňom veľa vecí, ktoré sa dajú zlepšovať.

5.1.1 Aktuálne nedostatky

Vďaka frekvencii pulzov, ktoré sú posielané do motorov, sa v ráme robota generuje kmitanie vo vysokej frekvencii, ktoré tvorí rušenie pre ultrazvukové senzory. Tie na

základe meraní musia byť umiestnené minimálne 3 centimetre od rámu robota, aby fungovali správne.

Literatúra

[1] H. Choset a kol. *Principles of Robot Motion. Theory, Algorithms, and Implementations.* MIT Press, 2005.

[2] National Marine Electronics Association. *NMEA 0183 Interface Standard.* dostupné on-line: `https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard` (citované: 1. júna 2020).

[3] J. Case and E. Boldt. *Pulse Width Modulation with analogWrite.* Robotic Controls, 2018. dostupné on-line: `http://robotic-controls.com/learn/arduino/pulse-width-modulation-analogwrite` (citované: 1. júna 2020).

[4] J. Dúc. *Robotour with Laser Range Sensor, diploma thesis.* FMFI UK Bratislava, 2017.

[5] M. Fikar. *Local Map for a Robot for the RoboTour Contest, diploma thesis.* FMFI UK Bratislava, 2019.

[6] ELEC Freaks. *Ultrasonic Ranging Module HC - SR04.* dostupné on-line: `https://www.mouser.com/datasheet/2/813/HCSR04-1022824.pdf` (citované: 1. júna 2020).

[7] Korneliusz Jarzebski. *Tilt compensated HMC5883L + MPU6050 (GY-86 / GY-87).* HMC5883L Triple Axis Digital Compass Arduino Library, 2014. dostupné on-line: `https://github.com/jarzebski/Arduino-HMC5883L/blob/master/HMC5883L_compensation_MPU6050/HMC5883L_compensation_MPU6050.ino` (citované: 1. júna 2020).

[8] Raspberry Pi Trading Ltd. Raspberry pi 4 computer model b, 2020. https://static.raspberrypi.org/files/product-briefs/200521+Raspberry+Pi+4+Product+Brief.pdf.

[9] D. Matejka. *Lokalizačný systém pre mobilného robota, diplomová práca.* FMFI UK Bratislava, 2018.

[10] A. Matejov. *Efficient Convolutional Neural Networks Recognizing Driveable Trails, diploma thesis.* FMFI UK Bratislava, 2020.

[11] M. Moravčík. *Autonómny mobilný robot pre súťaž Robotour, diplomová práca.* FMFI UK Bratislava, 2015.

[12] M. Nadhajský. *Robotour, diplomová práca.* FMFI UK Bratislava, 2011.

[13] The GNOME Project. Glib v2.64.4, 2014. https://developer.gnome.org/glib/2.64/.

[14] Open Robotics. Robotic operating system, 2017. https://www.ros.org/.

[15] ECLIPSERA s.r.o. Arduino nanostrong v3.0, 2018. https://arduino-shop.cz/docs/produkty/0/643/1512125230.pdf.

[16] M. Tučáni. *Mobilný inteligentný robot s ramenom a stereovidením, diplomová práca.* FMFI UK Bratislava, 2015.

[17] Univerzita Komenského v Bratislave. Vnútorný predpis č. 12/2013, smernica rektora Univerzity Komenského v Bratislave o základných náležitostiach záverečných prác, rigoróznych prác a habilitačných prác, kontrole ich originality, uchovávaní a sprístupňovaní na Univerzite Komenského v Bratislave, 2013. https://uniba.sk/fileadmin/ruk/legislativa/2013/Vp_2013_12.pdf.

[18] R. Zeman. *Integrovaný navigačný systém pre mobilného robota, bakalárska práca.* FMFI UK Bratislava, 2017.