UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A
INFORMATIKY

Robocup at Home Education

BAKALÁRSKA PRÁCA

2023 **Matúš Granec**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A
INFORMATIKY

Robocup at Home Education

BAKALÁRSKA PRÁCA

Študijný program: **Aplikovaná informatika**

Študijný odbor: **Informatika**

Školiace pracovisko: **Katedra aplikovanej informatiky**

Vedúci práce: **Mgr. Pavel Petrovič, PhD.**

Bratislava 2023 **Matúš Granec**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matúš Granec
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: Robocup at Home Education
Robocup at Home Education

Anotácia: Mobilný robot Jupiter je postavený na platforme TurtleBot 2. Je vybavený lidarom, dvomi 3D kamerami, ramenom s 5 stupňami voľnosti, výkonným počítačom s OS Linux, mikrofónom a dotykovým displejom. Je určený pre stredoškolskú kategóriu Robocup at Home, kde sa roboty pohybujú v interiéri a pomáhajú človeku, ktorý na pomoc môže byť odkázaný. Okrem toho je robot vhodný aj na iné výučbové a výskumné aplikácie na Katedre aplikovanej informatiky.

Cieľ: Cieľom bakalárskej práce je pripraviť robota na použitie v tejto súťaži, čiže pripraviť tutoriál zložený zo sady dobre zdokumentovaných príkladov, ktoré sú relevantné pre výzvy v Robocup at home Education a ktoré pomôžu slovenským tímom, aby sa zúčastnili s robotom v tejto kategórii. Okrem toho bude tutoriál dobrou pomôckou pri vyučovaní predmetov v oblasti umelej inteligencie na Katedre aplikovanej informatiky. Súčasťou práce je niekoľko ukážkových netriviálnych príkladov využitia vytvorenej knižnice.

Literatúra: R.R.Murphy: Introduction to AI Robotics, MIT Press, 2000.
H.Choset et.al: Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005.

Kľúčové slová: mobilný robot, robocup at home education, navigácia, interakcia človek-robot

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

Dátum zadania: 01.09.2022

Dátum schválenia: 17.10.2022

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Rád by som vyjadril veľkú vďaku môjmu školiteľovi a vedúcemu práce Mgr. Pavlovi Petrovičovi, PhD. za vedenie práce, hodiny strávené pri oprave robota, množstvo cenných rád a trpezlivosť pri odpovedaní na moje otázky. Takisto by som sa rád poďakoval svojej rodine za všetku podporu, ktorú som z ich strany dostával počas písania práce.

Čestné vyhlásenie:

Čestne vyhlasujem, že som bakalársku prácu Robocup at Home Education vypracoval samostatne s použitím uvedenej literatúry, zdrojov dostupných na internete a využitím teoretických a praktických vedomostí.

V Bratislave dňa 19.5.2023

_____

Matúš Granec

Abstrakt

V tejto bakalárskej práci, ktorá sa zameriava na tému "Robocup at Home Education" sme pripravili mobilného robota Jupiter na účasť v súťaži Robocup at Home. Jupiter, postavený na platforme TurtleBot 2, je vybavený LiDAR-om, dvoma 3D kamerami, ramenom s piatimi stupňami voľnosti, výkonným počítačom s operačným systémom Linux, mikrofónom a dotykovou obrazovkou. Robot je určený pre stredoškolskú kategóriu súťaže Robocup at Home, v ktorej roboty pracujú v interiéri a pomáhajú jednotlivcom v rôznych situáciách. Okrem toho je robot vhodný pre rôzne vzdelávacie účely a výskumné aplikácie v rámci Katedry aplikovanej informatiky.

Robota sme pripravili na súťaž vytvorením tutoriálu pozostávajúceho z dobre zdokumentovaných príkladov, ktoré sú relevantné pre výzvy v Robocup at Home Education. Vytvorili sme tutoriál, ktorý pomôže slovenským tímom zapojiť sa do tejto kategórie s ich vlastnými Jupiter robotmi. Tento tutoriál slúži aj ako cenný zdroj pre výučbu predmetov súvisiacich s umelou inteligenciou v rámci Katedry aplikovanej informatiky. Práca obsahuje niekoľko netriviálnych príkladov použitia, ktoré demonštrujú možnosti vyvinutej knižnice.

Kľúčové slová:

Robocup at Home, TurtleBot 2, mobilný robot, tutoriál, robotické rameno, počítačové videnie, umelá inteligencia, Robot Operating System

Abstract

In this bachelor's thesis, which focuses on the topic of "Robocup at Home Education", we prepared the mobile robot Jupiter for participation in the Robocup at Home competition. Jupiter, built on the TurtleBot 2 platform, is equipped with a LiDAR, two 3D cameras, a 5-DOF arm, a powerful Linux-based computer, a microphone, and a touch screen. The robot is designed for the high school category of Robocup at Home, where robots operate in indoor environments and assist individuals in need. Additionally, the robot is suitable for various educational purposes and research applications within the Department of Applied Informatics.

We prepared the robot for competition by creating a tutorial consisting of well-documented examples that are relevant to the challenges in Robocup at Home Education. We created a tutorial, which will assist Slovakian teams in participating in this category with their own Jupiter robots. Furthermore, the tutorial serves as a valuable resource for teaching subjects related to artificial intelligence within the Department of Applied Informatics. The thesis includes several non-trivial exemplary use cases that demonstrate the capabilities of the developed library.

Keywords:

Robocup at Home, TurtleBot 2, mobile robot, tutorial, robotic arm, computer vision, artificial intelligence, Robot Operating System

Obsah

Úvod

Robocup@Home je renomovaná medzinárodná súťaž, ktorá sa zameriava na vývoj autonómnych robotov schopných asistencie ľuďom v domácom prostredí a organizuje sa od roku 2006 ako súčasť celej iniciatívy RoboCup, ktorej sa zúčastňujú predovšetkým univerzitné tímy. V rámci súťaží RoboCup existujú aj juniorské kategórie zastrešené iniciatívou RoboCup Junior. V posledných rokoch vznikla aj kategória RoboCup@Home Education určená najmä účastníkom do 19 rokov.

Edukačný aspekt tejto súťaže je nemenej dôležitý, pretože umožňuje žiakom a študentom skúmať oblasť robotiky, umelej inteligencie a interakcie medzi človekom a robotom. Pripraviť robota Jupiter na použitie v tejto súťaži v kategórii Robocup@Home Education je jedným z cieľov tejto práce. Ďalším cieľom tejto bakalárskej práce je vytvoriť a poskytnúť obsiahly tutoriál pozostávajúci z dobre zdokumentovaných príkladov použitia robota relevantných pre možné výzvy v kategórii Robocup@Home Education. Tutoriál má slúžiť ako cenný zdroj informácií pre slovenské tímy, ktoré sa plánujú zúčastniť súťaže s robotom Jupiter.

Vzhľadom k tomu, že ešte nedávno som aj ja bol žiakom strednej školy, resp. študentom vysokej školy v nižšom ročníku, viem, aký je to pocit byť oboznámený s obrovským množstvom informácií o niečom, čomu vôbec nerozumiem. Programovanie je téma na jednej strane častokrát zaujímavá a lákavá, no na druhej strane aj veľmi zložitá a časovo náročná na jej pochopenie. Aspekt náročnosti programovania môže odradiť potenciálne šikovného programátora. Preto má tutoriál slúžiť aj ako pomôcka na ušetrenie času, ktorý by programátor strávil hľadaním tutoriálov na internete a tým uľahčiť mu začiatky práce s robotom Jupiter.

Práca sa nebude zameriavať iba na teoretické aspekty kategórie Robocup@Home Education, ale poskytne aj niekoľko netriviálnych a praktických ukážok práce s robotom Jupiter. Tieto príklady predvedú schopnosti robota, akými sú interakcia s používateľom, počítačové videnie, rozpoznanie objektov, snímanie svojho okolia a pohyb po miestnosti. Prezentovaním týchto príkladov sa v práci snažíme motivovať žiakov stredných škôl, aby sa zúčastňovali vyššie uvedenej súťaže, ale aj študentov a výskumníkov, ktorí sa zaujímajú o prieskum v oblasti robotiky a umelej inteligencie.

V prvej kapitole si priblížime robot Jupiter a jeho jednotlivé časti, medzinárodnú iniciatívu Robocup@Home, rovnomennú súťaž, súťažnú kategóriu Robocup@Home Education, systém ROS, na ktorom robot funguje a v neposlednom rade aj ďalšie technológie, ktoré sme v práci s robotom využili.

Druhá kapitola bude slúžiť na predstavenie návrhu jednotlivých netriviálnych ukážok práce s robotom Jupiter. Popíšeme si cieľ každej ukážky, navrhneme architektúru komponentov a komunikáciu medzi nimi.

V tretej kapitole sa budeme venovať samotnej implementácii komponentov z ukážok z kapitoly Návrh systému a bližšie si rozoberieme jednotlivé komponenty programov a pozrieme sa aj na potenciálne vylepšenia ukážok v budúcnosti.

1 Teoretické východiská

V tejto kapitole si popíšeme robota Jupiter, s ktorým pracujeme a jeho komponenty, ktoré budeme využívať a systém ROS, na ktorom robot funguje. Takisto si povieme o edukačnej iniciatíve Robocup@Home Education, do ktorej sa dá s robotom Jupiter zapojiť.

1.1 Slovník pojmov

V slovníku uvedieme niekoľko cudzích pojmov a skratiek, ktoré budeme v práci používať.

ROS – Robot Operating System – je open-source platforma pre vývoj a riadenie robotov, poskytujúca nástroje a komunikačné mechanizmy pre jednotný a flexibilný vývoj robotických aplikácií.

Topic – téma – je kanál, prostredníctvom ktorého môžu rôzne časti systému ROS posielať správy medzi sebou.

Node – uzol – je samostatný výpočtový proces, ktorý vykonáva určité úlohy a komunikuje s inými nodeami prostredníctvom správ a topicov pre výmenu dát.

Subscriber – odoberateľ – je komponent, ktorý prijíma správy zo špecifického topicu a spracováva ich.

Publisher – vydavateľ – je komponent, ktorý posiela správy na špecifický topic, aby ich mohli prijímať a spracovávať ostatné komponenty.

Service – služba – je mechanizmus, ktorý umožňuje dvojsmernú komunikáciu medzi dvoma nodeami, kde jeden node poskytuje určitú službu a druhý ju môže volať a získať výsledok.

Message – správa – je definovaná štruktúra dát, ktorá sa používa na prenos informácií medzi nodeami cez topic.

Framework – rámec – je sada nástrojov, knižníc a pravidiel, ktoré poskytujú štruktúru a prostredie pre vývoj softvéru.

1.2 Robot Jupiter

V tejto časti podrobnejšie popíšeme, z akých komponentov sa robot Jupiter skladá, aby sme získali lepšiu predstavu, s čím pracujeme a čo všetko v našej práci využijeme.

Obrázok 1 robot Jupiter

1.2.1 Počítač

Robot Jupiter je vybavený výkonným počítačom s operačným systémom Linux. Distribúciou operačného systému je Ubuntu 16.04 (v novších verziách robotov je to Ubuntu 18.04). Počítač má procesor Intel Core i5-8259U, integrovanú grafiku od Intel-u. Úložný priestor predstavuje disk SSD s kapacitou 120 GB a operačná pamäť má kapacitu 8 GB.

1.2.2 Dotykový displej

Robot Jupiter je takisto vybavený dotykovým displejom s Full HD rozlíšením, obnovovacou frekvenciou 56-76MHz a HDMI vstupom.

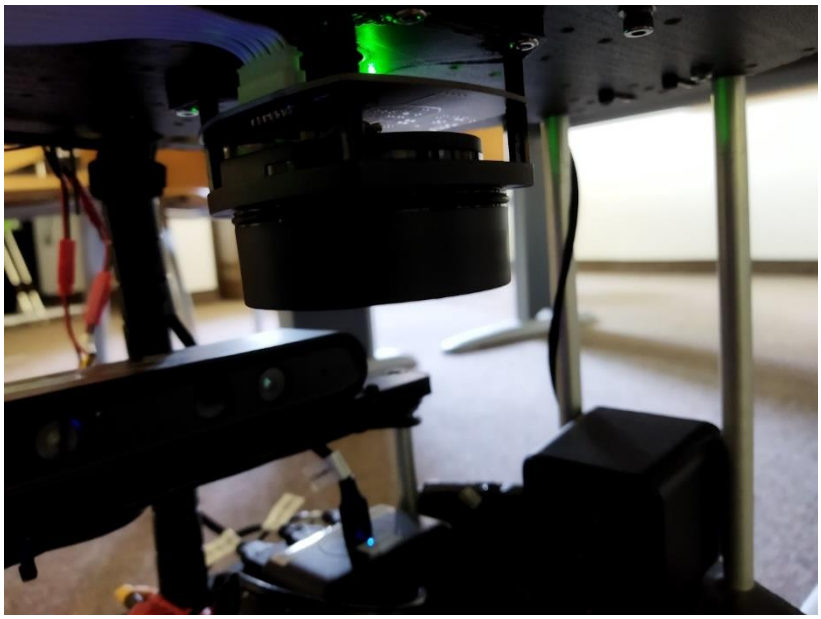
1.2.3 LiDAR

Jedným zo senzorov, ktorými robot disponuje je 360 stupňový skener, ktorý využíva metódu LiDAR – „light detection and ranging“ na snímanie okolia a meranie vzdialeností. Skener vyžaruje svetlo vo forme lúča, ktoré dopadá na povrch objektu a odráža sa späť do prijímača. Vzdialenosť povrchu objektu sa následne vypočíta pomocou času od vyslania lúča po čas jeho návratu a rýchlosti lúča.

Základný vzorec na výpočet vzdialenosti objektu od skenera využívajúceho metódu LiDAR:

$$d = \frac{c \cdot t}{2}$$

Obrázok 2 Základný výpočet vzdialenosti, Obrázok prevzatý z [17]



Obrázok 3 LiDAR na robotovi Jupiter

1.2.4 ORBBEC ASTRA 3D kamera

Ide o RGBD kameru s aktívnym stereo videním založeným na zabudovanom IR projektore. Táto kamera umožňuje využiť algoritmy počítačového videnia s viacerými funkciami, akými sú rozpoznávanie tvárí, rozpoznávanie gest, sledovanie ľudského tela, trojdimenzionálne meranie, vnímanie prostredia a trojdimenzionálna rekonštrukcia máp. Robot Jupiter obsahuje dve takéto kamery, z ktorých jedna je umiestnená nad displejom a druhá pod LiDARom.

Tabuľka 1 Technické parametre kamery

Product Name	**Astra**
Range	0.6m – 8m
FOV	60°H x 49.5°V x 73°D
RGB Image Resolution	640 x 480 @30fps
Depth Image Resolution	640 x 480 @30fps
Size	165mm x 30mm x 40mm
Temperature	0 – 40°C
Power Supply	USB 2.0
Power Consumption	< 2.4 W
Operating Systems	Android / Linux / Windows7/8/10
SDK	Astra SDK or OpenNI 2 or 3rd Party SDK
Accuracy	+/- 1 – 3mm @1 m
Microphones	2 Built-in

1.2.5 Robotické rameno

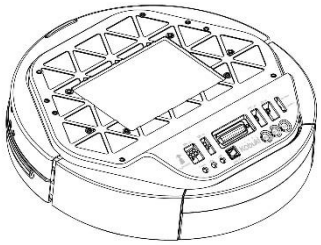
Robot Jupiter je vybavený robotickým ramenom s piatimi stupňami voľnosti, konkrétne so štyrmi bodmi otáčania v štyroch kĺboch a jedným chápadlom na uchopovanie a manipuláciu objektov. Na otáčanie kĺbov slúžia 4 servomotory Dynamixel AX-18A s rozsahom otáčania 0 až 300 stupňov. Úplne vystreté rameno má dĺžku 33 centimetrov a je umiestnené v strede robota.

1.2.6 Kobuki základňa

Kobuki je robotický podvozok, na ktorom je robot Jupiter postavený. Je to korytnačková základňa s motorom, vysokovýkonnými batériami a množstvom rozhraní, cez ktoré sa dajú následne pripojiť ďalšie periférne zariadenia alebo počítače, laptopy a iné. Na prístup ku všetkým funkcionalitám Kobuki základne je potrebné mať pripojenú externú výpočtovú jednotku ako počítač, laptop, tablet alebo vstavanú dosku so softvérom na komunikáciu s Kobuki.

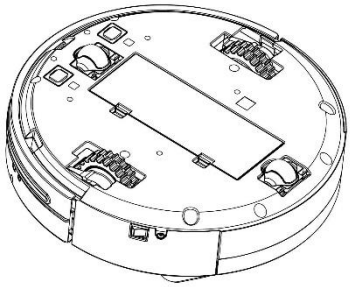
Kobuki základňa dosahuje maximálnu rýchlosť 70 cm/s, rýchlosť otáčania 180 stupňov/s. Maximálna nosnosť je 5 kg na podlahe a 4 kg na koberci. Očakávaný čas používania je pri menšej verzii s menšou batériou 3 hodiny a 7 hodín pri verzii s väčšou batériou. Očakávaný čas nabíjania menšej batérie je 1.5 hodiny a väčšej 2.6 hodiny.

Hardvér Kobuki obsahuje pripojenie k počítaču cez USB alebo RX/TX piny na sériovom porte. Ďalej obsahuje detekciu preťaženia motora, gyroskop, ktorý je továrensky kalibrovaný, nárazníky naľavo, v strede a napravo. Takisto sa na základni Kobuki nachádzajú senzory detekciu schodov a senzor prepadnutia kolesa. Súčasťou základne sú aj dve dvojfarebné programovateľné LEDky a niekoľko programovateľných sekvencií pípania. Samozrejmosťou na základni Kobuki je aj lítium-iónová batéria s nominálnym napätím 14.8 V, veľkosťou 2200 mAh alebo 4400 mAh a nabíjací adaptér so vstupom 100-240 V AC, frekvenciou 50/60 Hz, maximálnym prúdom 1.5 A a výstupom 19 V DC a prúdom 3.16 A [14].



Obrázok 4 Kobuki základňa pohľad zvrchu

Obrázok 5 Kobuki základňa pohľad zospodu



Obrázok 6 Kobuki základňa - ovládací panel

1.3 ROS – Robot Operating System

Robot Operating System je meta-operačný systém pre roboty. Obsahuje množstvo softvérových knižníc a nástrojov, ktoré pomáhajú budovať robotické aplikácie. ROS je open source softvér. Poskytuje služby, ktoré zahŕňajú abstrakciu hardvéru, nízkoúrovňové ovládanie zariadení, implementáciu bežne používaných funkcionalít, posielanie správ medzi procesmi a manažovanie balíčkov.

Beh ROS-u predstavuje graf peer-to-peer procesov siete, ktoré sú voľne poprepájané pomocou komunikačnej infraštruktúry ROSu [15]. ROS implementuje niekoľko rôznych štýlov komunikácie, ako napríklad synchrónne vzdialené volanie procedúr pomocou serviceov, asynchrónny tok dát pomocou topicov a ukladanie dát na serveri s parametrami.

ROS momentálne funguje iba na unix platformách. Testovaný je hlavne na Ubuntu a Mac OS X systémoch. Komunita však vytvára podporu aj pre Fedora, Gentoo, Arch Linux

a ostatné Linux platformy. Pre niektoré verzie ROSu existuje aj port na Microsoft Windows, avšak jeho podpora je obmedzená.

1.3.1 Ciele ROS-u

Cieľom ROS-u nie je byť frameworkom s čo najväčším množstvom vlastností. ROS má za hlavný cieľ podporovať opätovné použitie vytvoreného kódu na vývoj a výskum v oblasti robotiky. ROS je distribuovaný framework procesov v podobe nodeov, ktoré umožňujú to, aby bolo možné spustiteľné programy samostatne navrhnúť a počas behu programu ich voľne prepájať. Tieto procesy je možné zhromaždiť do balíčkov, ktoré sa dajú následne jednoducho distribuovať.

Okrem tohto hlavného cieľu má ROS aj niekoľko iných cieľov. ROS sa snaží byť čo „najtenší“ – hlavná časť kódu v časti main() nie je zbytočne obaľovaná, aby bol kód písaný pre ROS použiteľný aj v iných robotických frameworkoch. ROS je jednoducho integrovateľný s ostatnými robotickými frameworkami ako sú OpenRAVE, Orocos alebo Player.

ROS podporuje nezávislosť programovacích jazykov. Dá sa jednoducho implementovať v ktoromkoľvek modernom programovacom jazyku. ROS bol implementovaný v Pythone, C++ a Lispe a existujú aj experimentálne knižnice pre Javu a Lua.

V ROSe je zabudovaný jednotkový a integračný testovací framework, ktorý sa volá rostest a preto testovanie v ROS-e nepredstavuje veľký problém.

ROS je škálovateľný a preto je vhodný na veľké systémy a zložité vývojárske procesy.

1.3.2 Základné koncepty ROSu

Catkin je systém na budovanie balíčkov ROSu pre aplikácie a dá sa prirovnať napríklad k CMake alebo GNU Make. Nakoľko aplikácie môžu mať rôzne Python alebo C++ súbory, ROS vývojári sa rozhodli pre systém Catkin, ktorý pomerne dobre dokáže budovať balíčky v rôznych jazykoch.

Aplikácie sa nachádzajú v priečinku catkin_ws t.j. Catkin Workspace. V tomto priečinku je kód vyvíjaný, budovaný a testovaný. Na počítači môžeme mať viacero „pracovných priestorov“ a nemusia sa volať catkin_ws. Vytvoriť si nový pracovný priestor je vhodné pre každý nový samostatný a nezávislý projekt.

Kód samotný píšeme do jednotlivých ROS balíčkov, ktoré si vytvárame v našom pracovnom priestore. Balíček je najmenšia samostatná časť aplikácie, ktorá sa dá vybudovať, nainštalovať a spustiť. V balíčku sa nachádza zdrojový kód, skripty, CMakeLists súbory, launch súbory, message súbory, service súbory a iné. Každý balíček by sa mal všeobecne zameriavať na jednu časť aplikácie (autonómny pohyb, interakcia robota s človekom, atď.).

Jednou z najsilnejších vlastností ROSu je jeho modularita. Jednotlivé moduly bežia ako samostatné procesy. V systéme ROS sa proces nazýva ROS node. Každá aplikácia je vybudovaná z množstva nodeov. Komunikujú medzi sebou navzájom napríklad pomocou publisherov a subscriberov, alebo poskytovaním a využívaním synchrónnych služieb a štartovaním dlhodobejších akcií v architektúre klient/server.

Topic predstavuje v ROSe komunikačný kanál, umožňujúci výmenu informácií a dát medzi jednotlivými nodeami bez toho, aby o sebe dané nodey niečo museli vedieť. Každý node môže informácie na nejaký topic publikovať alebo informácie a dáta z neho odoberať. Toto oddeľovanie komunikácie umožňuje vyvíjať a testovať každý node jednotlivo a teda budovať modulárne a škálovateľné ROS systémy. Všetky topicy a nodey sa dajú vypísať pomocou nástroja rosgraph alebo graficky vizualizovať pomocou jeho grafickej verzie rqt_graph pre jednoduchšie ladenie a hľadanie chýb [16].

Message v systéme ROS predstavuje dátovú štruktúru, ktorá sa používa na posielanie informácií a dát cez topicy medzi jednotlivými nodeami. Message je definovaný typom messageu, ktorý je špecifikovaný v .msg súbore. Správy môžu obsahovať rôzne dátové typy ako integer, string, boolean atď. ako aj polia [16].

Publisher v ROSe predstavuje typ nodeu, ktorý posiela dáta cez message na určitý topic. Publisher vytvorí objekt message, naplní ho potrebnými dátami a následne ho publikuje na daný topic. V správe sa bežne posielajú dáta zo senzorov, pokyny na ovládanie robota alebo informácie o stave systému [16].

Subscriber je typ nodeu v ROSe, ktorý spracúva dáta z určitého topicu. ROS middleware sa stará o prijímanie a posielanie messageov všetkým subscriberom, ktorí odoberajú daný topic. Subscriber následne daný message už spracuje podľa potrieb [16].

Service je typ komunikácie medzi nodeami, ktorá umožňuje nejakému nodeu vyžiadať určitú akciu od iného nodeu. Service je definovaný dvojicou messageov v jednom .srv súbore – žiadosťou a odpoveďou. Žiadajúci node pošle žiadosť service nodeu, ktorý žiadosť spracuje a pošle message s odpoveďou späť žiadateľovi. Service sa tak ako publisher a subscriber využíva na ovládanie správania robota alebo vykonávanie špecifických akcií. Rozdielom je však to, že zatiaľ čo komunikácia cez publishera, subscribera a topicy je asynchrónna, čiže nečaká na spracovanie messageov, komunikácia cez service je synchrónna a žiadateľ čaká na odpoveď predtým, než niečo vykoná [16].

Action server takisto ako service predstavuje komunikáciu medzi nodeami, ktorá má umožniť nodeu požiadať o dlho bežiacu akciu od iného nodeu. Akcia je definovaná dvojicou messageov, ktoré sú špecifikované v jednom .action súbore: požiadavka a spätná väzba. Využíva sa napríklad na komplexné úlohy, akými sú pohyb robotického ramena na určité miesto alebo vizuálne hľadanie objektov. Na rozdiel od service komunikácie, action server posiela spätnú väzbu žiadajúcemu node-u aj počas toho, ako akcia prebieha. Vďaka tomu môže žiadajúci node monitorovať progres akcie a robiť rozhodnutia na základe tejto spätnej väzby [16].

1.3.3 Distribúcie ROSu

Distribúcia je set ROS balíčkov v nejakej verzii (Melodic, Noetic, Kinetic, Lunar) a pridružená infraštruktúra ROSu. Distribúcie ROSu sú spravidla priradené jednotlivým distribúciám Linuxu (Ubuntu). Účelom jednotlivých distribúcií je poskytnúť používateľom relatívne stabilný a bezchybný zdrojový kód. Preto, pri portovaní aplikácie na novú distribúciu sú potrebné zmeny a úpravy, bez ktorých aplikácia v novej verzii nebude automaticky fungovať. Keďže potreby

jednotlivých robotov sú rôzne, v budúcnosti sa očakáva, že časti komunity si budú vytvárať vlastné distribúcie, ktoré budú vyhovovať ich využitiu.

1.3.4 ROS Kinetic

ROS Kinetic je distribúcia ROSu, ktorá je nainštalovaná na robotovi Jupiter. Táto distribúcia vyšla dňa 23. mája 2016 a je desiata v poradí. Koniec podpory pre túto distribúciu bol naplánovaný na apríl 2021. Kinetic je primárne zameraný na distribúciu Linuxu Ubuntu verzie 16.04 (Xenial), ale do istej miery podporuje aj Mac OS X, Android a Windows.

Novšia verzia softvérovej infraštruktúry robota Jupiter využíva ROS Melodic asociovaný s Ubuntu 18, ale vzhľadom na to, že dodanom robotovi bola vyladená verzia Kinetic, nepristúpili sme k upgrade s predpokladom, že následne rovno preskočíme na v súčasnosti odporúčanú verziu Noetic, ale najskôr chceme preskúmať možnosti robota v pôvodnej verzii.

1.3.5 ROS2

ROS2 je nová generácia pôvodného systému ROS, ktorá má oproti pôvodnému ROS-u niekoľko vylepšení v oblasti komunikácie, bezpečnosti, flexibility použitia rôznych programovacích jazykov a požiadaviek v reálnom čase. Je vytvorená s cieľom posunúť ROS z výlučne akademického prostredia aj smerom k aplikáciám vo svete priemyslu.

Komunikačné protokoly v pôvodnom ROSe boli postavené na TCP alebo UDP protokoloch. ROS2 používa DDS protokol, ktorý je viac flexibilný.

ROS nebol navrhnutý s ohľadom na požiadavky systémov v reálnom čase, zatiaľ čo ROS2 obsahuje podporu systémov s požiadavkami na beh v reálnom čase, ako napríklad lepšiu záruku doručenia správ.

ROS2 podporuje na rozdiel od starého ROS-u viac programovacích jazykov, ako napríklad Python, C++, Java a ďalšie.

ROS2 obsahuje zabudované bezpečnostné vlastnosti, ako šifrovanie a overovanie pre lepšiu ochranu pred bezpečnostnými hrozbami.

Štandardizácia rozhraní medzi komponentami má viac formalizovaný proces a tým sa uľahčuje použitie komponentov tretích strán.

ROS2 však napriek všetkým jeho výhodám oproti pôvodnému systému ROS nepoužívame, nakoľko robot Jupiter má ešte starý operačný systém, v ktorom je podporovaný iba pôvodný ROS. Väčšina súčasných akademických systémov ešte len pomaly začína prechádzať na ROS2 a preto v ňom ešte nie je dostatočná podpora a komunita.

1.3.6 TurtleBot Gazebo

Gazebo popri CoppeliaSim je jeden z dvoch najznámejších robotických simulátorov. Oba sú dobre integrované so systémom ROS. TurtleBot Gazebo je 3D prostredie pre robotickú platformu TurtleBot, ktoré je určené na simuláciu funkcií robota. TurtleBot je open-source, nízko-nákladový mobilný a prispôsobiteľný robot postavený nad podvozkom Kobuki určený na výskum, vzdelávanie a voľnočasové projekty.

V 3D prostredí Gazebo sa robot dokáže pohybovať, interagovať s objektami a vnímať svoje okolie. Vývojári v tomto prostredí môžu testovať svoj kód a algoritmy predtým alebo bez toho, aby ich nasadili na skutočného robota. TurtleBot Gazebo je veľmi dobrý nástroj na vývoj a testovanie robotického softvéru a stal sa podstatnou súčasťou systému ROS.

1.3.7 RViz

RViz je nástroj slúžiaci na 3D vizualizáciu a je zároveň základnou súčasťou systému ROS. Pomocou RViz sa dajú vytvárať a zobrazovať modely robotických komponentov v 3D prostredí. Tento nástroj poskytuje aj grafické používateľské rozhranie na manipuláciu a interakciu s týmito objektami. So systémom ROS pracuje tento nástroj bezproblémovo a integrácia robotických modelov a dát zo senzorov do aplikácií je veľmi jednoduchá.

Medzi vlastnosti vizualizácie patria nástroje na zobrazenie mračien bodov, laserových skenov a obrázkov z kamery, ako aj podpora na simuláciu kinematiky a dynamiky. Používatelia

si takisto môžu prispôsobiť vzhľad svojho robotického modelu. Zmeniť si vedia farbu, veľkosť alebo priehľadnosť jednotlivých komponentov.

1.4 Ďalšie nástroje a knižnice

V tejto časti práce si popíšeme technológie, nástroje a knižnice, s ktorými sme pracovali a využili sme ich v tejto bakalárskej práci. Budeme popisovať knižnice OpenCV, gTTS, SpeechRecognition, Pygame, PIL, Numpy, TensorFlow, model neurónovej siete YOLOv3, neurónovú sieť Darknet, USB akcelerátor Google Coral a počítač Raspberry Pi 4.

1.4.1 Python

Python je jednoduchý, no výkonný programovací jazyk, ktorý premosťuje priepasť medzi programovaním v jazyku C a shell a preto je ideálny na „programovanie na jedno použitie“ a rýchle prototypovanie [1]. V Pythone je kladený dôraz na čitateľnosť kódu čistotou svojej syntaxe a na definovanie blokov kódu sa v ňom používa odsadenie.

V oblasti strojového učenia sa rámce (framework) Pythonu ako TensorFlow a scikit-learn stali populárnou voľbou pre vytváranie a tréning modelov [2].

Python sa vo veľkej miere používa na stredných školách a vzhľadom na ciele tejto práce je preto prirodzenou voľbou.

1.4.2 OpenCV

OpenCV (Open Source Computer Vision Library) je rozsiahlo používaná open source knižnica na počítačové videnie a strojové učenie. OpenCV bol pôvodne vyvinutý spoločnosťou Intel a neskôr podporovaný Willowom Garageom a Itseezom. Poskytuje rozsiahlu zbierku nástrojov, algoritmov a funkcií pre rôzne úlohy týkajúce sa počítačového videnia [3].

Medzi nástroje na spracovanie obrazu alebo videa patria napríklad detekcia objektov, rozpoznávanie, sledovanie objektov a segmentácia. Zdrojový kód OpenCV je napísaný primárne v programovacom jazyku C++, vďaka čomu sú algoritmy dostatočne rýchle a OpenCV poskytuje API aj pre programovacie jazyky Python, MATLAB, Java a ďalšie.

1.4.3 Numpy

NumPy (Numerical Python) je jednou zo základných a rozsiahlo používaných open-source knižníc pre numerické výpočty v programovacom jazyku Python [4]. NumPy je nevyhnutným stavebným kameňom pre vedecké výpočty a analýzu údajov, ponúka efektívne dátové štruktúry, výkonné matematické funkcie a nástroje na prácu s poľami. Jeho hlavnou súčasťou je objekt ndarray (N-rozmerné pole), ktorý umožňuje efektívne ukladanie a manipuláciu s rozsiahlymi, homogénnymi dátovými súbormi.

Efektívne operácie s poľami a matematické funkcie, ktoré ponúka NumPy, výrazne prispievajú k výkonu, škálovateľnosti a výpočtovej efektivite algoritmov strojového učenia. Poskytuje potrebné nástroje na predspracovanie údajov, inžinierstvo funkcií a ohodnotenie modelov, čo umožňuje odborníkom zostaviť a nasadiť efektívne systémy strojového učenia [4].

1.4.4 Pygame

Pygame je populárna knižnica pre Python, ktorá poskytuje robustný rámec pre tvorbu hier a multimediálnych aplikácií. Obsahuje širokú paletu funkcií, ktoré umožňujú manipuláciu s grafikou, zvukom a používateľskými vstupmi.

Pygame ponúka možnosti práce so zvukom a hudbou, čo umožňuje vývojárom začleniť zvukové efekty a hudbu na pozadí do svojich hier. Modul mixéra poskytuje funkcie na načítanie a prehrávanie rôznych zvukových formátov, ovládanie úrovní hlasitosti a implementáciu zvukových efektov na zlepšenie herného zážitku [5].

1.4.5 SpeechRecognition

SpeechRecogntion knižnica v Pythone je výkonný nástroj, ktorý poskytuje integráciu rozpoznávania reči do python projektov a interpretovať hovorené slovo do textovej formy.

SpeechRecognition podporuje viacero nástrojov na rozpoznávanie reči, vrátane Google Speech Recognition, CMU Sphinx a Microsoft Azure Speech. Tieto prostriedky (engine) využívajú algoritmy a techniky strojového učenia na konverziu hovoreného jazyka do textu. Využitím týchto prostriedkov môžu vývojári prepisovať hovorené slovo, vykonávať hlasové príkazy a umožniť hlasom ovládané interakcie vo svojich aplikáciách [6].

1.4.6 gTTS

Knižnica gTTS (Google Text-to-Speech) je Pythonovská knižnica, ktorá umožňuje vývojárom jednoducho konvertovať text na reč pomocou rozhrania Google Text-to-Speech API. Poskytuje pohodlný spôsob generovania hovoreného zvuku z textového obsahu [7].

1.4.7 YOLOv3

You only look once (YOLO) je najmodernejší systém detekcie objektov v reálnom čase. [8]

YOLOv3 stavia na svojich predchodcoch YOLO (You Only Look Once) a YOLOv2 a prináša niekoľko vylepšení na zlepšenie schopností detekcie objektov. Volí jednorazový prístup detekcie, kde je celý obraz spracovaný konvolučnou neurónovou sieťou (CNN), aby sa simultánne predpovedali ohraničujúce výrezy a pravdepodobnosti tried pre viacero objektov [8].

1.4.8 Darknet

Darknet je open source neurónová sieť napísaná v C a CUDA. Je rýchly, ľahko sa inštaluje a podporuje výpočty CPU a GPU. [8] Slúži ako základný komponent pre rodinu algoritmov na detekciu objektov YOLO (You Only Look Once), vrátane YOLOv3 [8]. Darknet poskytuje platformu na trénovanie a nasadenie neurónových sietí pre rôzne úlohy počítačového videnia. Vďaka svojej efektívnosti, rýchlosti a pomerne nenáročným hardvérovým požiadavkám je Darknet dobrou voľbou pre aplikácie, ktoré vyžadujú detekciu a rozpoznávanie objektov v reálnom čase.

1.4.9 Raspberry Pi 4

Raspberry Pi 4 je počítač s jednou doskou vyvinutý nadáciou Raspberry Pi Foundation. Ide o štvrtú generáciu zo série Raspberry Pi, ktorá ponúka vylepšený výkon, rozšírenú konektivitu a možnosti [11].

Raspberry Pi 4 obsahuje výkonný štvorjadrový procesor ARM Cortex-A72 s frekvenciou až 1.5 GHz a pamäť RAM o veľkostiach 2GB, 4GB a 8GB. Štvrtá generácia Raspberry Pi podporuje 4K video výstup na pripojenie dvoch monitorov naraz. Raspberry Pi 4 takisto obsahuje USB 3.0 port na rýchlejší presun dát a USB2.0 port na pripojenie periférnych zariadení a gigabitový eternetový port na vysokorýchlostné sieťové pripojenie. Ďalej má zabudované bezdrôtové pripojenie Wi-Fi a Bluetooth na pripojenie bezdrôtových periférií.

1.4.10 TensorFlow

TensorFlow je open source systém na strojové učenie, vyvinutý spoločnosťou Google Brain. Poskytuje komplexný ekosystém nástrojov, knižníc a prostriedkov na vytváranie a nasadzovanie modelov strojového učenia [12].

Jednou z kľúčových vlastností TensorFlow je grafová abstrakcia výpočtov, ktorá používateľom umožňuje definovať a vykonávať zložité matematické operácie vo forme výpočtového grafu. Tento prístup umožňuje efektívne spúšťanie na rôznych hardvérových platformách vrátane CPU, GPU a dokonca aj na špecializovaných akcelerátorov, ako sú TPU (Tensor Processing Unit).

1.4.11 Google Coral USB Accelerator

Google Coral USB Accelerator je hardvérové zariadenie, určené na zrýchlenie úloh strojového učenia. Je špeciálne optimalizovaný pre efektívne a rýchle spúšťanie modelov neurónových sietí [13].

Coral USB Accelerator obsahuje Edge TPU (Tensor Processing Unit), čo je na mieru navrhnutý ASIC (Application-Specific Integrated Circuit), vyvinutý spoločnosťou Google. Edge TPU poskytuje vysokovýkonnú inferenciu strojového učenia s nízkou spotrebou energie, vďaka čomu je vhodný pre rôzne výpočtové zariadenia vrátane Raspberry Pi.

Hlavná myšlienka činnosti tohto zariadenia spočíva v tom, že namiesto toho, aby výpočet prebiehal na klasickom von-Neumannovskom CPU, alebo na množine takýchto procesorov zoradených do GPU, výpočtová architektúra je vysoko paralelizovaná, obsahuje množstvo veľmi jednoduchých, efektívne prepojených malých samostatných výpočtových jednotiek, ktoré dokážu vykonávať desaťtisíce až stotisíce operácií súčasne – čo presne zodpovedá potrebám hlbokých neurónových sietí.

K počítaču sa pripojí cez USB 3.0 port a poskytuje podporu pre úlohy, akými sú rozpoznávanie objektov, počítačové videnie alebo rozpoznávanie reči. Dá sa použiť s viacerými populárnymi rámcami ako TensorFlow a PyTorch a podporuje programovacie jazyky Python

a C++. Výhodou Google Coral je, že je to malé prenosné „pripoj a hraj“ zariadenie, ktoré si nevyžaduje výrazné zmeny hardvéru na to, aby sme ho mohli začať používať.

1.5 Robocup@Home Education

Robocup At Home Education je edukačná iniciatíva na podporu vývoja servisných robotov, zameraných na umelú inteligenciu a na zvýšenie záujmu o zapojenie sa do tejto iniciatívy.

V rámci iniciatívy bežia štyri programy, a to **RoboCup@Home Education Challenge, Open Source Educational Robot Platforms, OpenCourseWare a Outreach Programs.**

Robocup@Home Education Challenge je edukačná súťažná platforma, ktorá slúži na podporu začiatočníckych tímov v súťažiach Robocup@Home Challenges. Súťaže fungujú na princípe workshopu spojeného so samotnou súťažou, kde majú účastníci za úlohu vypracovať a naprogramovať náročnejšie zadania, zamerané na vývoj robota a umelú inteligenciu v rámci istého časového limitu. Súťaže sú usporadúvané v rámci Robocup komunity na medzinárodnej alebo miestnej úrovni.

1.5.1 Pravidlá Robocup@Home Education Challenge

Pravidlá pre Robocup@Home Education Challenge sú veľmi podobné pravidlám opísaným v oficiálnej knihe pravidiel pre RoboCup@Home [18] v snahe dodržať isté štandardy. Jednotlivé úlohy sú však prispôsobené pre tímy nováčikov za účelom vzdelávania v tejto oblasti.

Tímy môžu súťažiť s dvomi typmi robotov, a to Open Platform a Standard Platform. Tímy v kategórii Open Platform používajú vlastnoručne zloženého robota, napríklad robota Jupiter. Standard Platform tímy používajú štandardizovanú robotickú platformu robota Pepper od SoftBank Robotics. Standard Platform kategória je viac zameraná na softvérový dizajn.

V súťaži máme dve hlavné kategórie súťažiacich podľa veku a to Open kategória pre účastníkov v akomkoľvek veku a kategória Junior pre účastníkov mladších ako 19 rokov.

Na kvalifikáciu do medzinárodnej súťaže musia súťažné tímy poslať isté kvalifikačné materiály, akými sú dokument s popisom tímu a video o tíme. Tím by takisto mal mať preukázateľné technické zdatnosti alebo skúsenosti, mal by pozostávať hlavne z nováčikov a pokiaľ má tím vlastného robota, ten by mal byť v podobnej cenovej relácii ako roboty poskytované súťažou.

Jednotlivé úlohy sa rozdeľujú do troch kategórií a to na navigačné úlohy, úlohy zamerané na robotické videnie a rečové úlohy. Pri manipulačných úlohách pre Open Platform robotov bude objekt umiestnený v dosahu robota podľa jeho výšky.

Súčasťou súťaže je aj príprava tímového plagátu vo formáte A1, ktorý bližšie predstaví technický vývoj tímu. Na konci workshopu jednotlivé tímy ešte pred začiatkom súťaže odprezentujú svoje plagáty.

Body sú udeľované postupne za každý podcieľ, ktorý robot splní, na rozdiel od Robocup@Home, pri ktorom sa boduje finálny cieľ úlohy. Čiastkové ohodnotenie úloh má pomôcť tímom začiatočníkov, pre ktorých môže byť náročnejšie splniť komplexnejšie zadané úlohy.

Pravidlo preskočenia umožňuje tímom preskočiť zložitejšiu časť zadania, resp. podcieľ a riešiť nasledovný podcieľ. Tímy sa majú pokúsiť splniť zadanie aspoň čiastočne, aj keď iná časť ich programu nefunguje, napr. vyriešiť rozpoznávanie objektu napriek tomu, že nefunguje navigácia. Tímy nemôžu opakovať podcieľ, ktorý preskočili, ale musia riešiť nasledujúci podcieľ.

Pravidlo zjednodušenia má motivovať tímy, aby zložitejšie podciele nepreskakovali predchádzajúcim pravidlom, ale pokúsili sa vyriešiť ich zjednodušenú verziu za bodovú penalizáciu. Napríklad pri rozpoznávaní objektov bude robot rozpoznávať tímom vybraný objekt namiesto objektu, ktorý by určila porota. Tím musí pred prezentáciou riešenia oznámiť, že chce využiť toto pravidlo.

Prezentácia výsledkov a finálnych úloh prebieha nasledovne: Každý tím ma na prípravu, prezentáciu a demonštráciu vypracovaného zadania desať minút. Po prezentácii nasleduje päť minút, počas ktorých sa porota pýta tímu otázky. Tím musí počas rozhovoru dať veci v prostredí prezentácie do pôvodného stavu.

1.5.2 Implikácie vyplývajúce z pravidiel

Z pravidiel a typov úloh popísaných vyššie vieme vyvodiť nejaké implikácie a závery toho, čo by mal robot zvládať, aby tím mohol potenciálne uspieť v plnení úloh súťaže Robocup@Home Education Challenge. Typy úloh v súťaži sú zamerané na navigáciu robota, robotické videnie, rozpoznávanie objektov a rečové schopnosti robota.

Z toho vyplýva, že robot by mal zvládať autonómnu navigáciu po zmapovanej miestnosti s vyhýbaním sa prekážkam. Ďalej by mal byť schopný rozpoznávať každodenné objekty, nájsť ich v priestore a manipulovať nimi pomocou robotického ramena. Takisto by mal byť schopný pomocou mikrofónu a reproduktorov komunikovať s človekom.