

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A
INFORMATIKY



Robocup at Home Education

BAKALÁRSKA PRÁCA

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.**



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

Pod'akovanie: 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.....	13
1 Teoretické východiská	15
1.1 Slovník pojmov	15
1.2 Robot Jupiter	16
1.2.1 Počítač.....	16
1.2.2 Dotykový displej	17
1.2.3 LiDAR	17
1.2.4 ORBBEC ASTRA 3D kamera.....	19
1.2.5 Robotické rameno	19
1.2.6 Kobuki základňa	20
1.3 ROS – Robot Operating System.....	21
1.3.1 Ciele ROS-u.....	22
1.3.2 Základné koncepty ROSu.....	23
1.3.3 Distribúcie ROSu.....	24
1.3.4 ROS Kinetic	25
1.3.5 ROS2	25
1.3.6 TurtleBot Gazebo	26
1.3.7 RViz.....	26
1.4 Ďalšie nástroje a knižnice	27
1.4.1 Python	27
1.4.2 OpenCV	28
1.4.3 Numpy.....	28
1.4.4 Pygame.....	29
1.4.5 SpeechRecognition.....	29
1.4.6 gTTS.....	29
1.4.7 YOLOv3	30
1.4.8 Darknet.....	30
1.4.9 Raspberry Pi 4	30
1.4.10 TensorFlow.....	31
1.4.11 Google Coral USB Accelerator.....	31
1.5 Robocup@Home Education.....	32
1.5.1 Pravidlá Robocup@Home Education Challenge	32
1.5.2 Implikácie vyplývajúce z pravidiel	34

2	Návrh riešenia	35
2.1	Ovládanie hlasom	35
2.1.1	Architektúra programu	36
2.1.2	Komunikácia komponentov	36
2.2	Detekcia prekážok pomocou LiDARu	37
2.2.1	Architektúra programu	37
2.2.2	Komunikácia komponentov	37
2.3	Nájdenie pohára v obraze a presunutie ramenom	38
2.3.1	Architektúra programu	38
2.3.2	Komunikácia komponentov	38
2.4	Rozpoznanie osoby podľa oblečenia	40
2.4.1	Architektúra programu	40
2.4.2	Komunikácia komponentov	40
3	Implementácia	42
3.1	Ovládanie hlasom	42
3.1.1	Node na rozpoznanie reči	42
3.1.2	Node na spracovanie príkazov	43
3.1.3	Komponent spätnej väzby	43
3.1.4	Potenciálne vylepšenia a využitie v budúcnosti	44
3.2	Detekcia prekážok pomocou LiDAR-u	45
3.2.1	Node na snímanie okolia	45
3.2.2	Node na pohyb robota	45
3.2.3	Potenciálne vylepšenia a využitie v budúcnosti	47
3.3	Nájdenie pohára v obraze a presunutie ramenom	48
3.3.1	Node na snímanie obrazu	48
3.3.2	Komponent na nájdenie súradníc pohára	48
3.3.3	Node na odoslanie súradníc pohára	50
3.3.4	Node na vypočítanie pozície ramena	50
3.3.5	Node na pohyb ramena	50
3.3.6	Potenciálne vylepšenia a využitie v budúcnosti	51
3.4	Rozpoznanie osoby podľa oblečenia	53
3.4.1	Node na snímanie obrazu	53
3.4.2	Klient na odoslanie obrázku	53
3.4.3	Server na spracovanie obrázku	54
3.4.4	Komponent na klasifikáciu obrázku	54
3.4.5	Node na odoslanie výsledku klasifikácie	55

3.4.6	Node na rozpoznanie osoby.....	55
3.4.7	Komponent spätnej väzby.....	57
3.4.8	Potenciálne vylepšenia a využitie v budúcnosti.....	57
4	Tutoriál.....	58
5	Záver	61
6	Literatúra	63

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, navrhne 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 posielat' 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

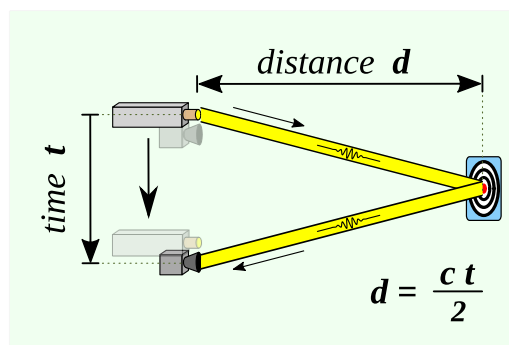
Robot Jupiter je takisto vybavený dotykovým displejom s Full HD rozlíšením, obnovovacou frekvenciou 56-76MHz a HDMI vstupom.

1.2.3 LiDAR

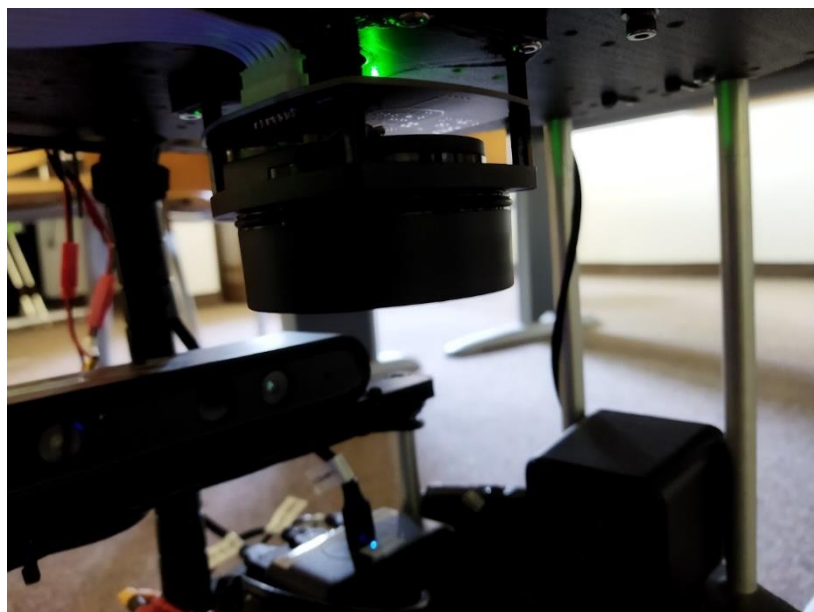
Jedným zo senzorov, ktorými robot disponuje je 360 stupňový skener, ktorý využíva metódu LiDAR – „light detection and ranging“ na snímanie okolia a meranie vzdialeností. Skener vyžaruje svetlo vo forme lúča, ktoré dopadá na povrch objektu a odráža sa späť do prijímača. Vzdialenosť povrchu objektu sa 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 tváří, 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

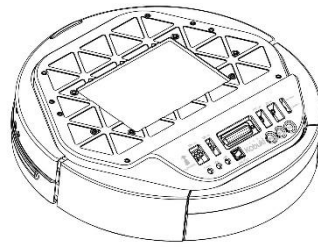
s rozsahom otáčania 0 až 300 stupňov. Úplne vystreté rameno má dĺžku 33 centimetrov a je umiestnené v strede robota.

1.2.6 Kobuki základňa

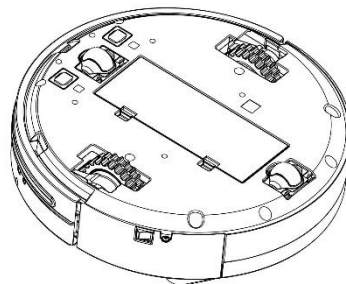
Kobuki je robotický podvozok, na ktorom je robot Jupiter postavený. Je to korytnačková základňa s motorom, vysokovýkonnými batériami a množstvom rozhraní, cez ktoré sa dajú následne pripojiť ďalšie periférne zariadenia alebo počítače, laptopy a iné. Na prístup ku všetkým funkcionalitám Kobuki základne 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.

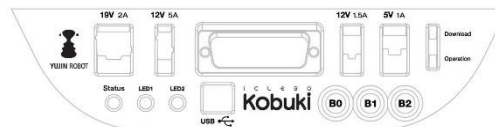
Hardvér Kobuki obsahuje pripojenie k počítaču cez USB alebo RX/TX piny na sériovom porte. Ďalej obsahuje detekciu preťaženia motora, gyroskop, ktorý je továrensky kalibrovaný, nárazníky naľavo, v strede a napravo. Takisto sa na základni Kobuki nachádzajú senzory detekciu schodov a senzor prepadnutia 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 synchronne 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 navrhnuť 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 obalovaná, 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 dlhodobějších 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 oddelovanie 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 `rostopic` 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 topicu 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 desiatu 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 prispôbiť 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 premostňuje 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 súborami.

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 úrovni hlasitosti a implementáciu zvukových efektov na zlepšenie herného zážitku [5].

1.4.5 SpeechRecognition

SpeechRecognition 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 tréning 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 ethernetový 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 začiatočnických 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 prispôbené 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.

2 Návrh riešenia

Ako sme uviedli v úvode, jedným z hlavných cieľov tejto práce je pripraviť čo najviac materiálu, ktorý môže poslúžiť stredoškolským tímom, ktoré by mali záujem o účasť v kategórii RoboCup@Home Education, ako zdroj informácií, inšpirácií a možných prípadov použitia.

V tejto kapitole si preto bližšie popíšeme a navrhujeme jednotlivé ukážky práce s robotom Jupiter, ktoré sú súčasťou balíka „matus_showcase“. V ukážkach je snaha demonštrovať jednotlivé implikácie popísané v časti 1.4.2. Ukážky, ktorých fungovanie tu popíšeme sú Ovládanie hlasom, Detekcia prekážok pomocou LiDARu, Nájdenie a presunutie pohára a Rozpoznanie osoby podľa oblečenia.

Pri každej ukážke rozoberieme cieľ ukážky, architektúru a komunikáciu medzi jednotlivými komponentami. Komponenty samotné a algoritmy, ktoré sú v nich použité si opíšeme v kapitole 3 – Implementácia.

2.1 Ovládanie hlasom

Cieľom tejto ukážky je program, ktorý dostane vstup z mikrofónu, tento vstup sa spracuje ako príkaz pre robota a robot následne príkaz vykoná. Robot očakáva rečový vstup v anglickom jazyku. Pokiaľ sa vstup nedá rozpoznať alebo ak daný príkaz neexistuje v pripravených príkazoch robota, robot zahlási chybu. To, aký príkaz ide vykonať alebo aká chyba nastala robot zahlási pomocou reproduktorov.

Robot momentálne pozná príkaz „recognize“, ktorý spustí program Rozpoznanie osoby podľa oblečenia, príkaz „find object“, ktorý spustí Nájdenie a presunutie pohára a príkaz „move forward“, ktorý spustí ukážku Detekcia prekážok pomocou LiDARu.

2.1.1 Architektúra programu

Tento program pozostáva z dvoch ROS nodeov a jedného komponentu spätnej väzby. Jeden node zabezpečuje rozpoznávanie reči a odosielanie rozpoznaného výsledku cez ROS a druhý node má na starosti vykonávanie jednotlivých príkazov na základe rozpoznaného výsledku. Komponent spätnej väzby slúži na samotné prehratie zvuku pre zodpovedajúci príkaz, ktorý sa vykonáva alebo zvuku konkrétnej chybovej hlášky.

2.1.2 Komunikácia komponentov

Na obrázku č.7 je znázornená komunikácia medzi ROS nodeami. Obrázok bol vytvorený pomocou nástroja rqt_graph spomenutého v 1.2.3, ktorý dokáže vykresliť jednotlivé nodey a topicy ROSu.

Topic „voiceCommand“ je znázornený ako obdĺžnik, na ktorý sú pomocou šípok napojené dva naše nodey. Smer šípky určuje smer toku dát a teda šípka smerujúca smerom do topicu určuje, že node, z ktorého vychádza („speech_recognizer“) je publisher a naopak šípka vychádzajúca z topicu určuje, že node, do ktorého šípka smeruje („voice_command_listener“) je subscriber.



Obrázok 7 Diagram komunikácie ovládania hlasom

2.2 Detekcia prekážok pomocou LiDARu

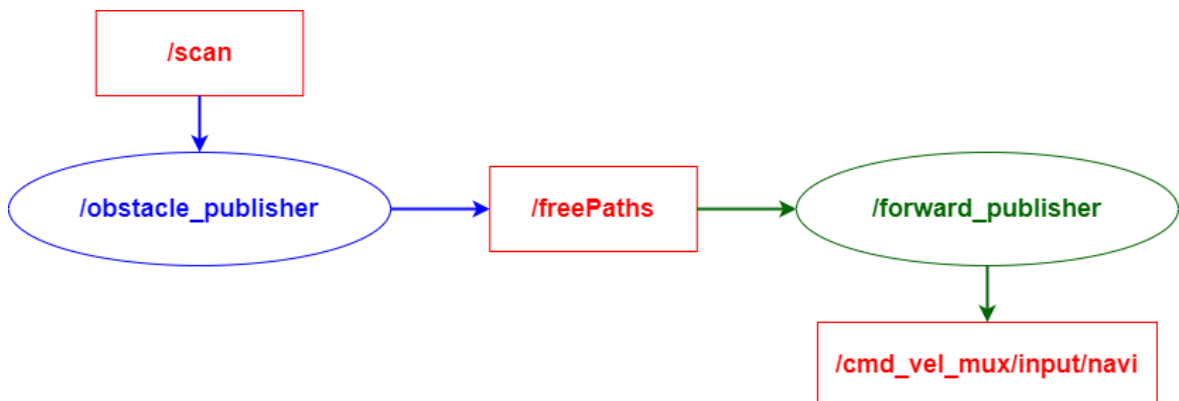
Táto ukážka má za cieľ predviesť využitie LiDARu na snímanie okolia a zachytenie prekážok v robotovej blízkosti a pohyb robota po miestnosti bez toho, aby narazil do niektorej z nasnímaných prekážok.

2.2.1 Architektúra programu

Program sa skladá z dvoch ROS nodeov. V prvom node robot pomocou LiDARu sníma okolie a údaje spracúva a zisťuje prítomnosť prekážok v každom smere. Voľné smery alebo smery s prekážkou robot odosiela pomocou nami vytvorenej správy ROS. Druhý node má na starosti zabezpečiť pohyb robota s vyhýbaním sa prekážkam, ktoré LiDAR zachytí.

2.2.2 Komunikácia komponentov

Komunikáciu medzi nodeami opäť znázorníme pomocou nástroja rqt_graph. Ako vidno na obrázku č.8 node „obstacle_publisher“ dostáva nasnímané údaje LiDARom z topicu „scan“. Následne ich spracuje a odošle na topic „freePaths“. Z tohto topicu zas údaje o prekážkach získava node „forward_publisher“ a následne po spracovaní posiela dáta s pohybom robota na topic „cmd_vel_mux/input/navi“.



Obrázok 8 Diagram komunikácie detekcie prekážok a pohybu robota

2.3 Nájdenie pohára v obraze a presunutie ramenom

V tejto ukážke má robot za úlohu nasnímať obraz, ktorý vidí cez spodnú kameru. Robot fotografiu následne spracuje pomocou neurónovej siete a zistí, či sa v nej nachádza pohár. Ak sa pohár v obraze nachádza, odošlú sa súradnice ohraničujúceho boxu (bounding box) s pohárom vnútri. Súradnice ohraničujúceho boxu sa spracujú a podľa nich vypočítame, ako sa má pohnúť rameno robota. Rameno dostane súradnice pohybu a pohne sa na miesto, na ktorom sa pohár nachádza, pohár vezme a premiestni na robota, vedľa základne ramena. Táto ukážka má predviesť a využiť počítačové videnie robota a manipuláciu objektami pomocou robotického ramena.

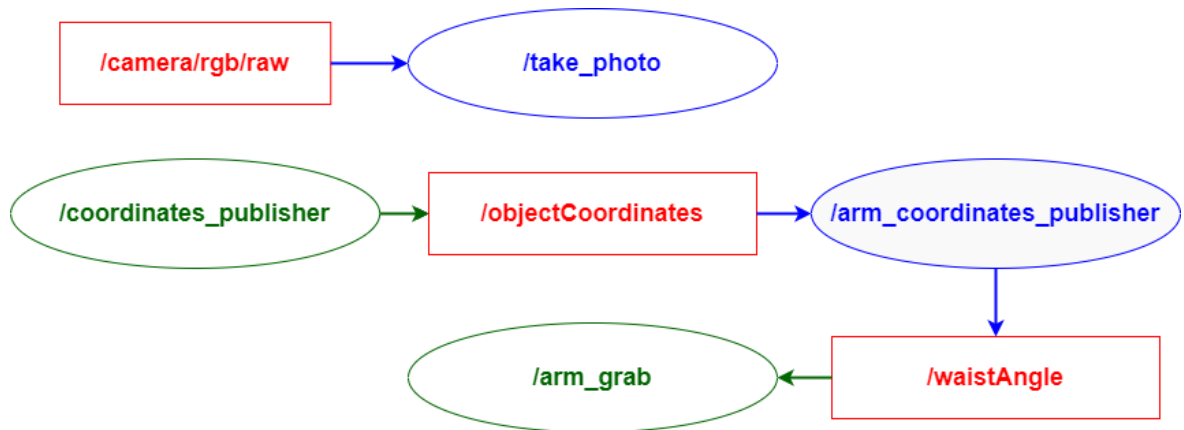
2.3.1 Architektúra programu

Tento program sa skladá z viacerých komponentov, pričom každý z nich má za úlohu niečo iné. Ako prvý je potrebný node na snímanie obrazu, ktorý robot práve vidí cez kameru. Ďalším komponentom je modul, ktorý pomocou neurónovej siete Darknet a modelu YOLOv3 spracuje odfotený obrázok a vráti súradnice ohraničujúceho boxu, v ktorom sa pohár nachádza. Tieto súradnice sa následne pošlú cez vytvorený node na odosielanie súradníc. Súradnice sa posielajú cez nami vytvorenú správu. Tieto súradnice prečíta ďalší node, určený na spracovanie súradníc ohraničujúceho boxu a vypočítanie súradníc pozície robotického ramena. Pozícia ramena je odoslaná na ďalší nami vytvorený topic, z ktorého node číta dáta na pohyb ramena. Tento node následne podľa súradníc vezme pohár, položí ho vedľa základne ramena na robota a vráti sa do počiatočnej pozície.

2.3.2 Komunikácia komponentov

Komunikáciu medzi jednotlivými komponentami si opäť zobrazíme pomocou `rqt_graph` na obrázku č.9. V prvej časti programu vidíme, že node „take_photo“ odoberá topic „camera_top/rgb/raw“. Po vypočítaní súradníc prichádza na rad node „coordinates_publisher“, ktorý posiela súradnice na topic „objectCoordinates“. Tento topic

odoberá node „arm_coordinates_publisher“. Topic „waistAngle“ dostáva dáta z nodeu „arm_coordinates_publisher“ a node „arm_grab“ tieto dáta odoberá.



Obrázok 9 Diagram komunikácie nájdenia a presunutia pohára

2.4 Rozpoznanie osoby podľa oblečenia

V poslednej naprogramovanej ukážke máme príklad toho, ako robot dokáže rozpoznať človeka a zapamätať si ho podľa určitej špecifickej časti oblečenia. Využívame robotické videnie, rečové schopnosti robota, TCP/IP komunikáciu s Raspberry Pi typu server-klient a Google Coral USB urýchľovač neurónových sietí.

2.4.1 Architektúra programu

Tento program sa skladá z viacerých, vzájomne poprepájaných komponentov. Prvým komponentom je node na snímanie obrazu. Obraz pomocou komponentu klienta na odoslanie obrazu odošleme na Raspberry Pi, kde ho prijímame komponentom servera na spracovanie obrazu. Na Raspberry Pi máme komponent na klasifikáciu obrázku, ktorá vráti názov oblečenia. Kvôli nekompatibilite USB urýchľovača Google Coral s dnes už zastaranou verziou Ubuntu 16.06, ktorá je nainštalovaná na robotovi Jupiter, sme sa rozhodli umiestniť komponent na spracovanie obrazu na Raspberry Pi. Keď už máme informáciu o type oblečenia na obrázku, tak ju použitím nodeu na odosielanie informácií o oblečení odošleme na topic. Z tohto topicu dáta vezmeme a node na rozpoznanie osoby danú osobu buď rozpozna alebo si vypýta meno. Robot komunikuje s používateľom cez komponent spätnej väzby.

2.4.2 Komunikácia komponentov

Komunikáciu medzi jednotlivými komponentami si znázorníme pomocou `rqt_graph` nástroja a sekvenčného diagramu.

Na obrázku č.10 vidíme, že najskôr node „take_photo“ odoberá obrazové dáta z topicu „camera_top/rgb/raw“.

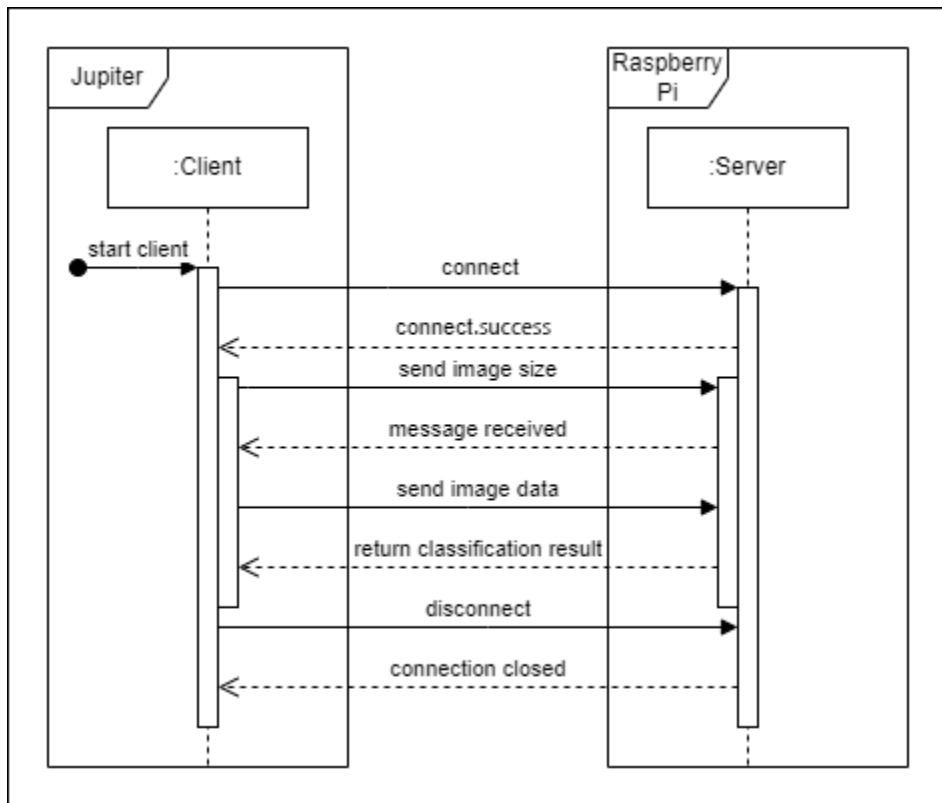
Na obrázku č.11 je znázornená komunikácia medzi klientom, ktorý je spustený na robotovi Jupiter a serverom, ktorý beží na Raspberry Pi. Klient si najprv vypýta spojenie a následne odošle informáciu o veľkosti obrázku a obrázok samotný. Server obrázok

spracuje a zavolá komponent na spracovanie obrázku. Výsledok klasifikácie obrázu sa odošle ako odpoveď klientovi a spojenie sa ukončí.

Na obrázku č.12 je zobrazená komunikácia medzi nodeom „clothing_publisher“, ktorý odosiela typ oblečenia na topic „clothes“ a node-om „people_recognizer“, ktorý tieto dáta odoberá a pracuje s nimi ďalej.



Obrázok 10 Diagram komunikácie snímania obrázu



Obrázok 11 Diagram komunikácie server-klient



Obrázok 12 Diagram komunikácie rozpoznávania osoby

3 Implementácia

V tejto kapitole si bližšie popíšeme jednotlivé komponenty ukážok z kapitoly 2 – Návrh riešenia. Takisto spomenieme potenciálne vylepšenia a ich využitie v budúcnosti.

3.1 Ovládanie hlasom

Tak ako uvádzame v časti 2.1.1, tento program sa skladá z troch komponentov, ktoré nižšie podrobnejšie opíšeme.

3.1.1 Node na rozpoznanie reči

Node na rozpoznanie reči je kľúčovým komponentom architektúry programu na ovládanie hlasom. Tento node je implementovaný ako pythonovský skript a využíva ROS na komunikáciu. Node počúva príkazy pomocou mikrofónu a pythonovskej knižnice `speech_recognition`. Táto knižnica vytvorí spojenie s Google Speech Recognition službou, ktorá prepíše zachytené audio do textu.

Počas inicializácie triedy `SpeechRecognizer` sa inicializuje aj ROS publisher, ktorý bude posielať dáta na „`voiceCommand`“ topic. Publisher posielala rozpoznaný text na topic ako reťazec pomocou ROS String message.

Ak je reč úspešne rozpoznaná a prevedená na text, tak sa výsledok rozpoznávania odošle na náš „`voiceCommand`“ topic. V prípade nejakej chyby alebo nerozpoznaného textu sa na náš topic odošle správa s textom „!Error“, aby ostatné nodey vedeli, že nastala chyba pri rozpoznávaní reči.

3.1.2 Node na spracovanie príkazov

Ďalšou veľmi dôležitou súčasťou ovládania hlasom je node na spracovanie príkazov. Tak ako aj predchádzajúci node je tento node implementovaný ako pythonovský skript, ktorý využíva ROS na komunikáciu. Je zodpovedný za počúvanie príkazov a vykonávanie príslušných akcií na ich základe.

Pri inicializácii „VoiceCommandListener“ triedy sa vytvorí spojenie na sieť ROS nodeov tým, že inicializujeme „voice_command_listener“ subscribera. Tento subscriber je odoberá/počúva dáta z topicu „voiceCommand“. Prijaté dáta z topicu sa spracúvajú vo funkcii „voice_command_callback“.

Funkcia na spracovanie dát je spustená zakaždým, keď subscriber dostane nejaké dáta. Tieto dáta sa zaznamenajú a program skontroluje, či sa daný príkaz nenachádza v slovníku preddefinovaných príkazov. Ak sa takýto kľúč v slovníku nachádza, spustí sa akcia zodpovedajúca danému príkazu. Ak robot dostane príkaz „!Error“, tak sa spustí príslušná „handle_error“ funkcia. Slovník príkazov sa dá jednoducho rozšíriť o ďalšie nové príkazy pridaním nového kľúča(hlasového príkazu) a vytvorením príslušnej funkcie, ktorá vykoná potrebné príkazy.

Tento node poskytuje aj spätnú väzbu vo forme audia. Na toto sa využíva komponent spätnej väzby, ktorý konvertuje text k danému príkazu do reči. Audio výstup je následne nahlas prehraný, vďaka čomu je rozhranie prirodzené a interaktívne.

Trieda „VoiceCommandListener“ takisto interaguje s ostatnými komponentami systému. Pomocou funkcie „os.system“ vykonáva systémové príkazy na volanie externých skriptov, ktoré zodpovedajú nášmu hlasovému príkazu.

3.1.3 Komponent spätnej väzby

Komponent spätnej väzby nie je nutne takou istou dôležitou súčasťou ovládania hlasom, akou je node na rozpoznávanie reči a spracovanie príkazov, avšak pre používateľa vytvára z interakcie prirodzený a interaktívny zážitok. Komponent je implementovaný v Pythone

a využíva „gTTS“ knižnicu a „Pygame“ knižnicu na konverziu textu na reč a prehrávanie audia používateľovi.

Trieda „Talker“ sa inicializuje spolu s parametrom, ktorým je text určený na vyslovenie. V inicializácii sa takisto vytvorí aj slovník „clothes_response“, ktorý má preddefinované odpovede ku konkrétnym častiam oblečenia. Tento slovník sa využíva v ukážke „Rozpoznanie osoby podľa oblečenia“, ktorú popíšeme neskôr. Po inicializácii sa vytvorí audio súbor na základe poskytnutého textu pomocou modulu „gTTS“, ktorý konvertuje text do reči v anglickom jazyku. Audio súbor je dočasne uložený, pokiaľ sa neprehrá.

V metóde „talk“ používa náš komponent na prehratie dočasného audio súboru s rečou modul „pygame.mixer“. Po prehraní audia sa „pygame.mixer“ zastaví a audio súbor sa vymaže.

3.1.4 Potenciálne vylepšenia a využitie v budúcnosti

Tento program je možné v budúcnosti vylepšiť viacerými spôsobmi. V prvom rade sa dajú veľmi jednoducho pridať nové príkazy, ktoré robot pozná. Do pripraveného slovníka stačí vložiť hlasový príkaz a metódu, ktorá vykoná potrebný úkon.

Ďalej by bolo možné spojzdníť kontinuálne rozpoznávanie hlasových príkazov, ktoré môže byť výhodné, ak používateľ nechce stále spúšťať program na rozpoznávanie reči a robota ovládať iba pomocou hlasu. Je to síce náročnejšie na zdroje, nakoľko treba dávať pozor na to, aby robot nepočul vlastné odpovede, ale na druhej strane je to používateľsky príjemnejšie.

Takisto by bolo zaujímavé pridať podporu ovládania robota vo viacerých jazykoch. Ďalším možným vylepšením by mohli byť personalizované odpovede, nastavenie hlasu robota alebo prízvuku podľa nastavení používateľa.

Veľkým prínosom by bolo, keby robot rozumel prirodzenému jazyku a vedel interpretovať zložitejšie príkazy a komunikovať s používateľom flexibilnejšie.

3.2 Detekcia prekážok pomocou LiDAR-u

V časti 2.2.1 sme spomínali, že tento program sa skladá z dvoch komponentov, ktoré predstavujú dva nodey v systéme ROS.

3.2.1 Node na snímanie okolia

Podstatnou súčasťou architektúry na detekciu prekážok je node na snímanie okolia. Vytvorený je v Pythone a využíva knižnicu rospy na komunikáciu s ROSom. Node odoberá dáta nasnímané laserovým skenom pomocou senzoru LiDAR z topicu „scan“.

Počas inicializácie node nastaví publisher na „freePaths“ topic s použitím nami vytvoreným typom správy „DetectedObstacles“. Tento publisher je zodpovedný za odosielanie informácií o zachytených prekážkach.

V metóde „scan_callback“ sa dáta z topicu „scan“ spracujú podľa vypočítaných uhlov sa uložia nasnímané vzdialenosti v zozname. Cez tento zoznam sa potom iteruje v metóde „check_distances“, kde sa skontroluje či je nejaký objekt dostatočne blízko robota, aby bol označený za prekážku. Prekážky sa zaznačia pre každý z ôsmich smerov – vpredu, vpredu vpravo, vpravo, atď. Slovník, v ktorom sú prekážky zaznačené, nám slúži na to, aby sme vedeli naplniť správu „DetectedObstacle“ údajmi o prekážkach a voľných smeroch pohybu. Táto správa sa potom publikuje na topic „freePaths“.

3.2.2 Node na pohyb robota

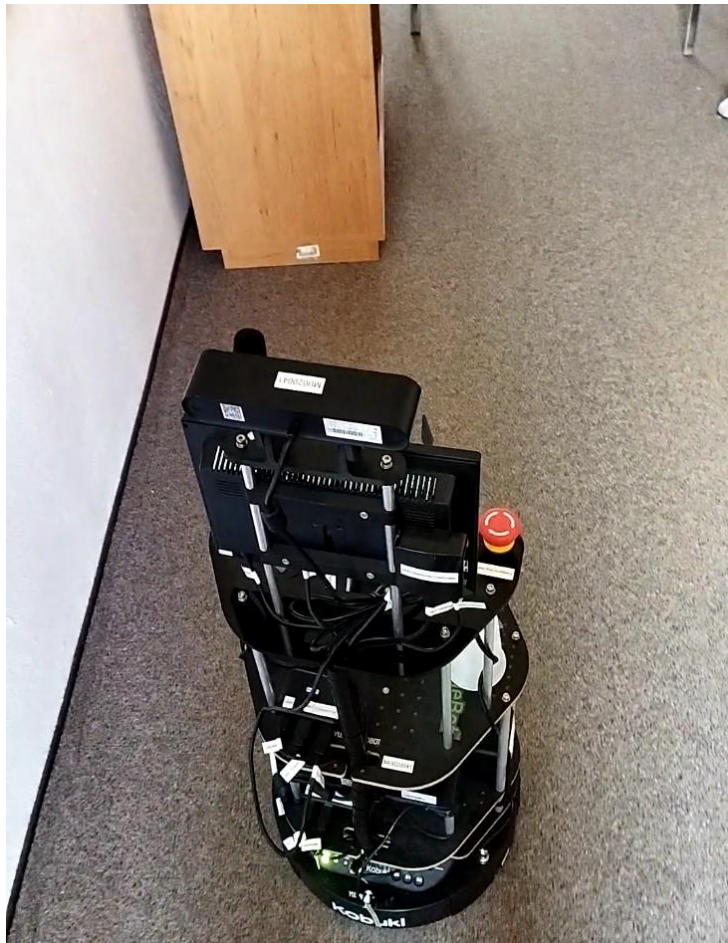
Tento node je opäť implementovaný v Pythone a s použitím ROSu a je zodpovedný za pohyb robota odosielaním príkazov na ovládanie rýchlosti.

Počas inicializácie vytvorí node jedného subscribera napojeného na náš „freePaths“ topic, aby vedel získať údaje o prekážkach a smere v ktorom sa nachádzajú. Takisto sa vytvorí aj publisher napojený na „cmd_vel_mux/input/navi“ topic, ktorý očakáva správu typu „Twist“ a spracúva príkazy na rýchlosť pohybu robota. Táto správa určuje lineárny a uhlový pohyb robota.

Funkcia „free_path_callback“ prečíta dáta z „freePath“ topic-u a podľa nich aktualizuje slovník v ktorom sú uložené jednotlivé smery a to či sa v danom smere nachádza prekážka.

Ďalej sú v triede MoveForward definované funkcie na pohyb robota. V každej z funkcií nastavíme rýchlosť pohybu lineárne a uhlovo. Pohyb máme definovaný pre smer vpred, vľavo, vpravo, vzad a takisto úplné zastavenie pohybu robota.

Logika vyberania ďalšieho smeru pohybu je jednoduchá a riešená v metóde „choose_path“. Ak sa prekážka nachádza priamo pred robotom, robot sa otočí do jedného zo smerov v ktorom sa momentálne žiadna prekážka nenachádza a po otočení sa do daného smeru sa opäť začne hýbať vpred.



Obrázok 13 Robot natočený od steny pretože ju LiDAR zachytil ako prekážku

3.2.3 Potenciálne vylepšenia a využitie v budúcnosti

Zatiaľ čo súčasná implementácia programu detekcie prekážok pomocou LiDAR poskytla uspokojivé výsledky, existuje niekoľko potenciálnych oblastí pre budúce zlepšenia a vylepšenia. Cieľom týchto vylepšení je zlepšiť funkčnosť, výkon a robustnosť systému.

Prvým vylepšením je integrácia viacerých senzorov a ich spojením na lepšie a kvalitnejšie snímanie svojho okolia. LiDAR totiž zachytí iba prekážky v určitej výške a preto sa môže stať, že niektoré prekážky vôbec nezachytí. Ďalším senzorom, ktorý by sme mohli použiť je 3D kamera s hĺbkovým vnímaním, ktorá by potenciálne videla aj prekážky, ktoré LiDAR nezachytil. Možno by sa oplatilo použiť aj obe kamery na robotovi spolu s LiDARom.

Ďalším možným vylepšením je začleniť nejaký algoritmus na plánovanie cesty. Vďaka tomu by sme vedeli optimalizovať cestu do nejakého cieľa, nájdením najrýchlejšej cesty s najmenším množstvom prekážok. Toto by značne zefektívnilo pohyb v premenlivom prostredí.

3.3 Nájdenie pohára v obraze a presunutie ramenom

Architektúru programu sme si popísali v časti 2.3.1. Nižšie popíšeme päť komponentov, z ktorých sa ukážka skladá.

3.3.1 Node na snímanie obrazu

Tento node je naprogramovaný v Pythone a takisto, ako ostatné využíva rospy knižnicu. Jeho hlavnou úlohou je urobiť snímku obrazu z kamerového topicu a uložiť ju do zvoleného priečinka ako súbor.

Počas inicializácie nodeu vytvoríme CvBridge objekt, ktorý umožňuje konverziu ROS obrazových správ na OpenCV obrázky a naopak. Takisto vytvoríme subscribera napojeného na kamerový topic, ktorý bude dostávať dáta o obrázkoch.

V „img_callback“ funkcii, ktorá sa zavolá vždy, keď príde nová správa na kamerový topic, sa správa v ROS formáte konvertuje do OpenCV formátu použitím CvBridge. Výsledný obrázok uložíme do premennej. Obrázok vo formáte PNG sa do nami vybraného priečinka uloží až po zavolaní funkcie „save_picture“, ktorá kontroluje aj to, či sa nám vôbec nejaké dáta z kamery podarilo dostať.

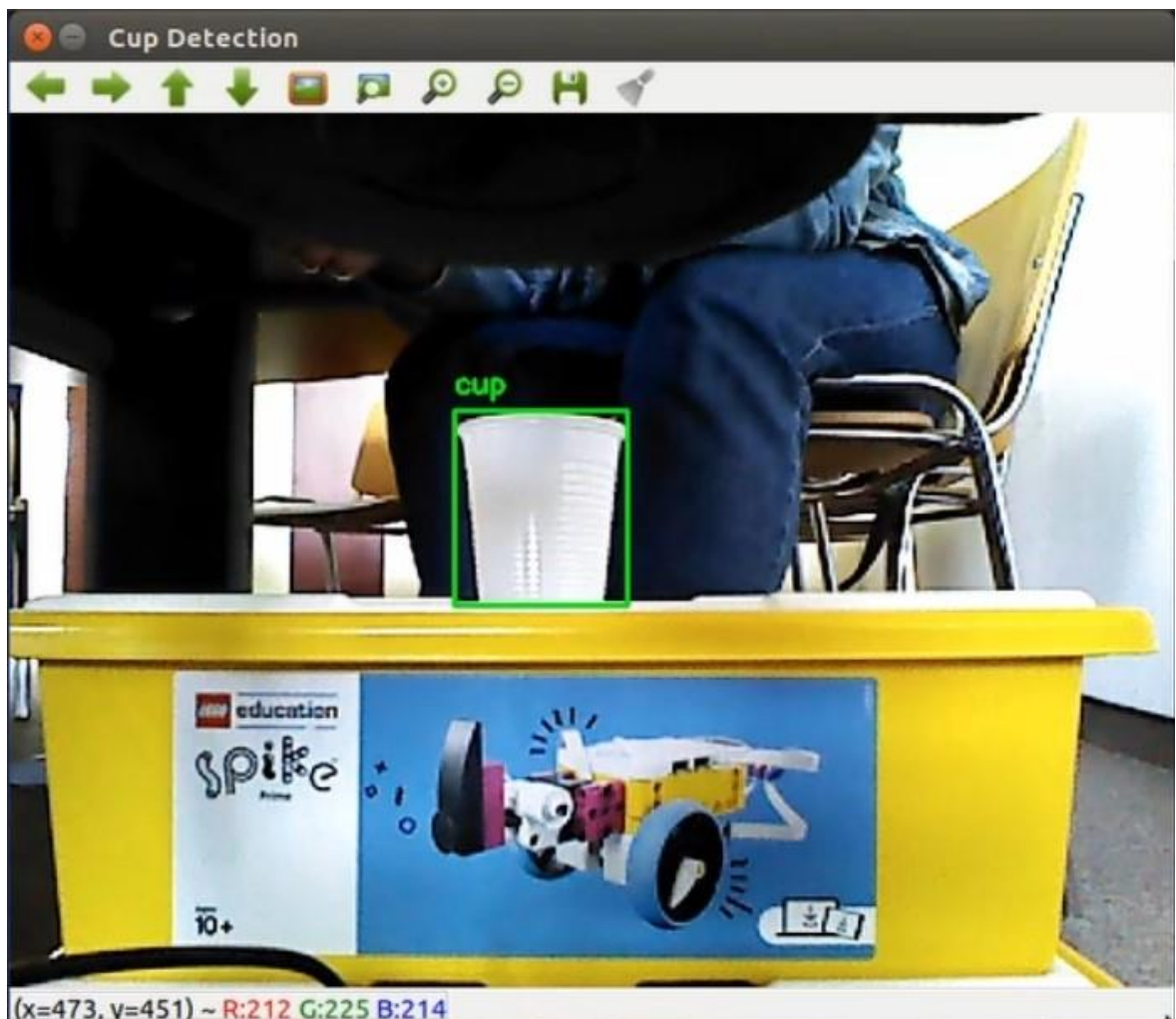
3.3.2 Komponent na nájdenie súradníc pohára

Komponent na nájdenie súradníc pohára je skript napísaný v Pythone. Tento skript využíva Darknet rozhranie na detekciu objektov použitím modelu YOLOv3. Tento skript vezme obrázok odfotený predchádzajúcim nodeom, nájde súradnice ohraničujúceho boxu, v ktorom sa nachádza pohár.

Na začiatku skriptu si nastavíme cesty k Darknet priečinku, obrázku, konfiguračnému súboru YOLOv3 modelu, váham YOLOv3 modelu a dátovému súboru coco a inicializujeme si samotnú neurónovú sieť Darknet.

Pomocou OpenCV si obrázok načítame a prevedieme ho do formátu kompatibilného s neurónovou sieťou Darknet pomocou metód implementovaných v Darknet rozhraní. Následne spustíme detekciu objektov v obrázku.

Výsledok detekcie prefiltrujeme, aby obsahoval iba súradnice objektov, označených ako pohár. Pre vizualizáciu a kontrolu do obrázka vykreslíme ohraničujúce boxy okolo pohárov spolu s ich označením. Toto samozrejme nemusíme zobraziť, nakoľko nás zaujímajú najmä súradnice týchto boxov, ale uľahčí to vizuálnu kontrolu správnosti detekcie objektov. Súradnice boxov uložíme a odošleme ich ako argument ďalšiemu nodeu, ktorý ich spracuje.



Obrázok 14 Pohár s nakresleným ohraničujúcim boxom okolo neho

3.3.3 Node na odoslanie súradníc pohára

Úlohou tohto nodeu je odosielať súradnice ohraničujúceho boxu pohára na „object_coordinates“ topic. Je písaný v Pythone a využíva funkcionality ROS-u.

Pri inicializácii triedy CoordinatesPublisher potrebujeme zadať aj argument so súradnicami ohraničujúceho boxu. Tieto súradnice priradíme ako dáta do nami vytvorenej správy na súradnice. Správu napokon pošleme na topic pre ďalší node.

3.3.4 Node na vypočítanie pozície ramena

Node na vypočítanie pozície ramena spracúva súradnice ohraničujúceho boxu a podľa nich prepočítava, ako sa majú jednotlivé kĺby ramena otočiť, tak aby sa rameno dostalo nad pozíciu pohára a chápadlo mohlo pohár uchopiť. Pri inicializácii sa vytvorí subscriber, ktorý číta súradnice ohraničujúceho boxu z topicu „ObjectCoordinates“.

Ako prvé potrebujeme vypočítať otočenie „pásu“ robota vzhľadom k zornému uhlu kamery a offsetu kamery od ramena. Ďalej potrebujeme zistiť vzdialenosť pohára od robotického ramena a na to budeme potrebovať rozmery pohára (výšku a šírku) ako aj rozmery ohraničujúceho boxu. Opäť musíme zohľadniť posunutie kamery od ramena. Nakoniec použitím inverznej kinematiky a goniometrie môžeme vypočítať súradnice každého kĺbu ramena. Súradnice kĺbov potrebujeme nakoniec previesť ešte na radiány.

Tieto súradnice sa nakoniec pošlú pomocou publishera na topic „waistAngle“ v správe typu JointCoordinates, ktorá má záznam typu float64 pre každý kĺb robotického ramena.

3.3.5 Node na pohyb ramena

V neposlednom rade máme node určený na pohyb ramena. Pri inicializácii si vytvoríme publishera pre každý z piatich ovládačov pre kĺby robota. Prvý ovláda otáčanie „pásu“, druhý ovláda „rameno“, tretí ovláda „laket“, štvrtý ovláda „zápästie“ a piaty ovláda otváranie a zatváranie „ruky“ robota.

Robotické rameno začína v počiatočnej pozícii – vystreté nahor. Následne prečítame súradnicu otočenia „pásu“ a nastavíme ju do správy, ktorá bude odoslaná na topic ovládača „pásu“. Rameno sa pohne nad pohár, vezme ho a nasleduje sekvencia správ pre rameno, ktorá zabezpečí to, aby bol pohár položený na robota vedľa základne ramena. Rameno sa vráti do počiatočnej pozície.



Obrázok 15 Robot presúva pohár vedľa ramena

3.3.6 Potenciálne vylepšenia a využitie v budúcnosti

Tento program sa dá ešte vylepšiť a jeho využitie sa dá rozšíriť. Prvým z možných vylepšení je rozpoznávanie viacerých objektov, ako len objektu pohára a možnosť vybrať si, s ktorým objektom bude robot pomocou ramena manipulovať.

Vylepšiť by sa tento program dal aj vytvorením URDF súboru k robotickému ramenu robota Jupiter. Program MoveIt, ktorý je integrovaný v ROSe, by nám potom umožnil komplexnejšie využitie robotického ramena, ako je napríklad plánovanie pohybu,

uchopenie, inverzná kinematika alebo vyhýbanie sa prekážkam. Toto všetko sa dá implementovať bez potreby riešiť nízkoúrovňové algoritmy od úplného začiatku.

V neposlednom rade by sme mohli implementovať sledovanie pohybu objektu a monitorovať jeho pozíciu aj počas behu programu a robot by vedel ramenom sledovať momentálnu pozíciu pohára, a nielen pozíciu, ktorá bola vypočítaná zo snímky z kamery.

3.4 Rozpoznanie osoby podľa oblečenia

Z opisu architektúry tejto ukážky v časti 2.4.1 vidíme, že program je zložený zo siedmich rôznych komponentov, ktoré si teraz podrobnejšie priblížime.

3.4.1 Node na snímanie obrazu

Node na snímanie obrazu používame ten istý, ako v časti 3.3.1, ktorý sme využívali pri nájdení pohára v obraze. Jediný rozdiel pri tomto programe je, že na snímanie obrazu používame spodnú kameru robota a teda musíme node zavolať s príslušným argumentom.

3.4.2 Klient na odoslanie obrázku

Pomocou tohto klienta, ktorý je naprogramovaný v Pythone a využíva model klient-server, sme schopní odoslať odfotený obrázok na Raspberry PI, kde nastáva jeho ďalšie spracovanie.

Na to, aby sme mohli vytvoriť TCP/IP pripojenie, musíme najprv na Raspberry Pi nastaviť statickú IP adresu.

Najprv si vytvoríme TCP/IP pripojenie na server, ktorý sa nachádza na Raspberry PI. Prvú odošleme veľkosť obrázka ako štvorbajtové celé číslo a potom postupne odošleme samotné obrazové dáta. Klient potom čaká na odpoveď od servera, aký typ oblečenia má osoba na obrázku oblečený.

Túto odpoveď nakoniec pošleme ako argument pri inicializácii nodeu „ClothingPublisher“.

3.4.3 Server na spracovanie obrázku

Server na prijatie a spracovanie obrázku sa nachádza na Raspberry PI, ktoré je k robotovi pripojené pomocou LAN kábla.

Skript je napísaný v Pythone a využíva model klient-server. Kód vytvorí TCP/IP spojenie s komponentom na odosielanie obrázka. Server sa pripojí na konkrétnu IP adresu a port a čaká na prichádzajúce spojenia. Po úspešnom vytvorení pripojenia dostane server správu s veľkosťou prichádzajúceho obrázku, obrázok samotný a uloží ho do vybraného priečinka.

Následne zavoláme komponent, ktorý vykoná klasifikáciu obrázka a vráti nám názov oblečenia na obrázku. Názov oblečenia odošleme naspäť klientovi ako odpoveď.

3.4.4 Komponent na klasifikáciu obrázku

Tento skript, písaný v Pythone, zabezpečuje klasifikáciu obrázka použitím natrénovaného modelu a Edge TPU (Tensor Processing Unit) akcelerátora.

V skripte na začiatku definujeme cesty modelu, súboru so štítkami a obrázku, ktorý chceme klasifikovať. Potom inicializujeme interpretér použitím Edge TPU modelu a alokujeme tensoru.

Vstupný obrázok konvertujeme do RGB formátu pomocou PIL knižnice a zmeníme jeho veľkosť na veľkosť očakávanú modelom. Ak je to potrebné, obrázok ešte predspracujeme. Inferenciu na obrázku vykonáme päťkrát, aby sme dostali stabilné výsledky. Skript nám vráti názov daného oblečenia na obrázku.



Obrázok 16 Raspberry Pi 4 so zapojeným Google Coral USB akcelerátorom cez USB3.0

3.4.5 Node na odoslanie výsledku klasifikácie

Tento kód reprezentuje node v ROS rozhraní, ktorý sa správa ako publisher odosielajúci typ oblečenia, ktorý sa nachádza na obrázku pomocou správy typu „ClothesMessage“.

Pri inicializácii nodeu potrebujeme zadať typ oblečenia ako parameter, ktorý sa potom odošle na topic. Podľa typu oblečenia v správe nastavíme pravdivú boolovskú hodnotu k prislúchajúcemu riadku s typom oblečenia. Ostatným typom oblečenia nastavíme nepravdivú boolovskú hodnotu.

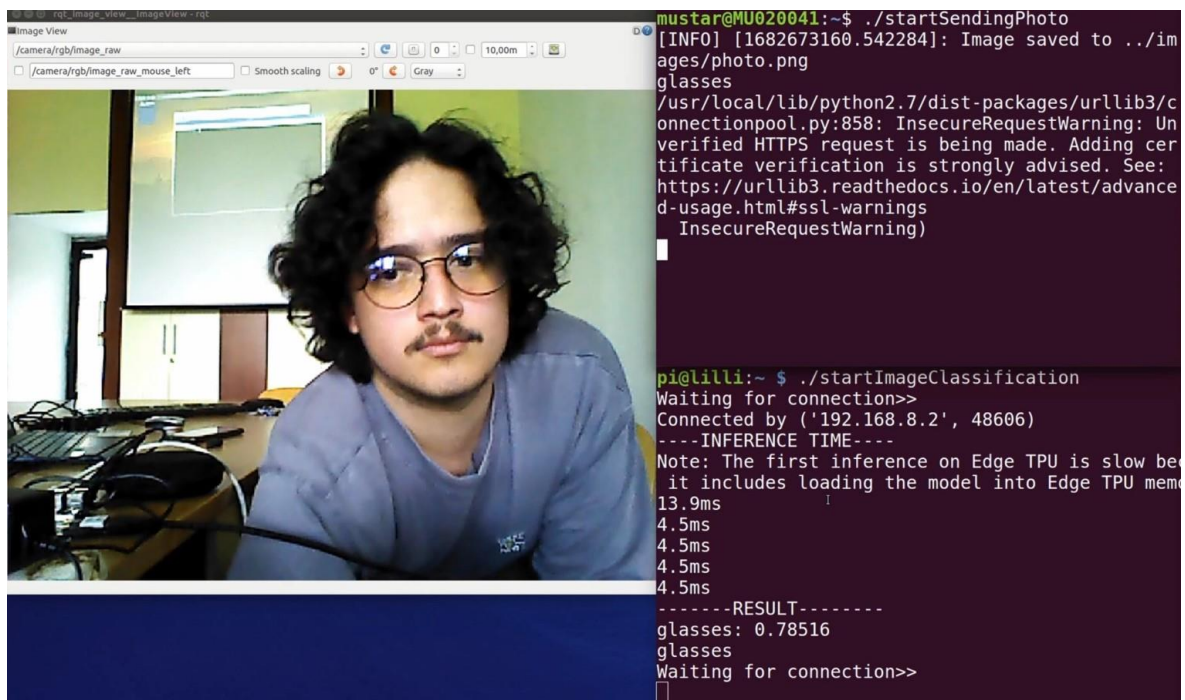
Správu o type oblečenia odošleme na náš topic iba raz, node sa potom zastaví.

3.4.6 Node na rozpoznanie osoby

Opäť máme node, ktorý je naprogramovaný v Pythone a využíva rozhranie ROS. Na začiatku si node inicializujeme a vytvoríme subscribera, ktorý dostáva dáta o oblečení, ktoré má na sebe osoba na obrázku.

Do slovníka, ktorý je na začiatku prázdny, si ukladáme typ oblečenia ako kľúč a meno osoby ako hodnotu. Pomocou metódy „recognize_person“, ktorá sa vykoná vždy, keď na topic príde nová správa, vieme osobu rozpoznať podľa názvu oblečenia. Ak daný kus oblečenia ešte nie je uložený v slovníku, vypýtame si meno osoby a túto dvojicu údajov do slovníka vložíme.

Následne robot povie aké oblečenie má osoba na sebe a pozdraví danú osobu menom. Toto spravíme pomocou komponentu spätnej väzby.



Obrázok 17 Node rozpoznal, že na obrázku sa nachádzajú okuliare, vid' okno terminálu vpravo dolu

3.4.7 Komponent spätnej väzby

Komponent spätnej väzby sa nemenil a ostáva rovnaký ako v sekcii 2.1.4. V tejto ukážke ho využívame na oznámenie typu rozpoznaného oblečenia na obrázku, ako aj na pozdravenie osoby podľa mena, ktoré je uložené v slovníku „people“.

3.4.8 Potenciálne vylepšenia a využitie v budúcnosti

Hoci táto ukážka predstavuje základný spôsob rozpoznávania osôb, nájdeme vylepšenia, ktoré by sa dali v budúcnosti implementovať do nášho programu a tým ho zefektívniť, spresniť a rozšíriť.

Prvým z vylepšení je natrénovať kvalitnejšiu neurónovú sieť na rozpoznávanie ľudí. Nový natrénovaný model by mohol byť schopný rozoznávať viacero typov oblečenia, prípadne aj kombinácie oblečenia, hlas osoby alebo tváre ľudí.

Jednou z nevýhod nášho programu je, že nebeží iba na robotovi Jupiter, ale časť programu beží na Raspberry Pi. Tento spôsob sme zvolili kvôli neschopnosti pripojiť Google Coral USB akcelerátor na robot Jupiter. Na to, aby sme vedeli vytvoriť pripojenie z robota Jupiter na Raspberry Pi potrebujeme mať aj funkčný ethernetový switch. Ako aj Raspberry Pi, tak aj ethernetový switch potrebuje zdroj elektriny a preto musíme zabezpečiť napájanie buď zo zásuvky, čo značne obmedzuje mobilitu robota alebo napájanie z batérií, ktoré však treba pravidelne nabíjať. Toto všetko sú obmedzenia, ktoré by zanikli napríklad použitím grafickej karty priamo na Jupiterovi, ktorú by bolo možné v budúcnosti na robot Jupiter pridať a neurónová sieť by potom mohla bežať rýchlejšie a iba v rámci robota Jupiter.

4 Tutoriál

Hlavným cieľom tejto práce bolo sprístupniť robota Jupiter záujemcom zo stredných stredných škôl. Preto sme vytvorili GitHub repozitár s jednotlivými ukážkami na tejto URL adrese:

https://github.com/Robotics-DAI-FMFI-UK/jupiter/tree/main/matus/matus_showcase

Keďže ide o spoločný repozitár všetkých študentov, ktorí s robotom už pracovali, tak sa v ňom nachádzajú aj ukážky programov iných študentov. Ukážky z tejto bakalárskej práce sa však nachádzajú v rovnomennom podpričinku repozitára názvu nášho balíčka - „matus_showcase“.

K naprogramovaným ukážkam sme vytvorili aj tutoriál, ktorý je spravený formou GitHub Wiki k nášmu repozitáru a nachádza sa na URL adrese:

<https://github.com/Robotics-DAI-FMFI-UK/jupiter/wiki>

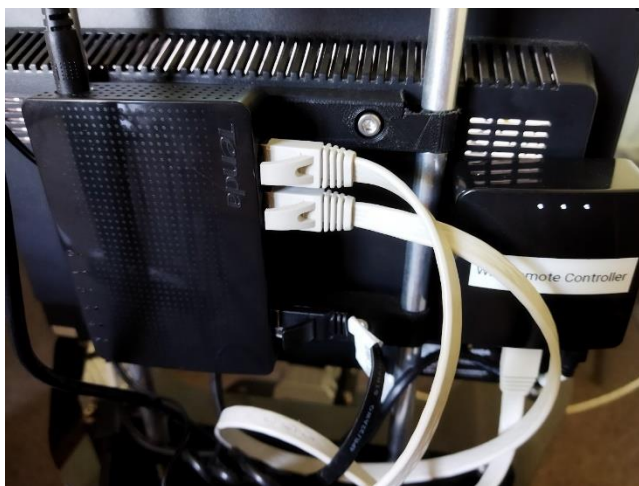
Tutoriál je rozdelený do niekoľkých logických spolu súvisiacich sekcií.

V prvej sekcii „Požiadavky“ popisujeme, čo všetko treba splniť na to, aby sme naše ukážky mohli spustiť na robotovi Jupiter. Predpokladáme, že robot Jupiter je v továrenskome stave a teda, je na ňom nainštalovaný iba ROS Kinetic. Postupne opisujeme ako nainštalovať jednotlivé knižnice a programy a takisto ako nakonfigurovať Raspberry Pi 4, s ktorým taktiež pracujeme.

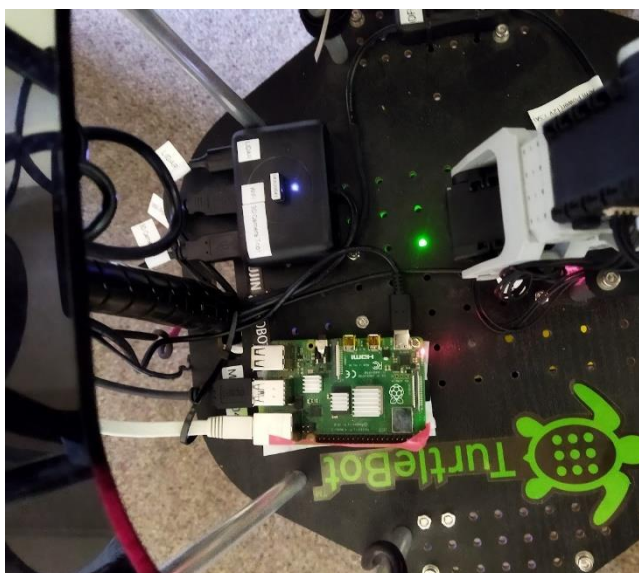
V druhej sekcii tutoriálu „ROS“ predstavíme systém ROS, na ktorom robot Jupiter funguje, vysvetlíme základné koncepty tohto systému a ukážeme si ako vytvoriť prvý ROS balík a node.

Vo zvyšných štyroch sekciách sa už venujeme konkrétnym ukážkam, vysvetľujeme ako fungujú a ukazujeme na nich princípy ROSu. V každej z týchto štyroch sekcií si najprv popíšeme, čo má ukážka robiť, potom si povieme ako ukážku spustiť a ďalej opisujeme jednotlivé časti kódu. Na konci ešte máme aj diagram komunikácie jednotlivých komponentov, ktorý sme vytvorili pomocou funkcionality rqt_graph. Pri každom programe je aj linka na video danej ukážky.

Na záver máme ešte zopár relevantných obrázkov k našej práci.



Obrázok 18 Ethernetový switch na pripojenie Raspberry Pi 4 k Jupiterovi



Obrázok 19 Raspberry Pi 4 pripevnené k robotovi Jupiter



Obrázok 20 Batéria, ktorá napája Raspberry Pi 4 a ethernetový switch



Obrázok 21 Google Coral USB akcelerátor pripevnený na robotovi Jupiter

5 Záver

Cieľom tejto bakalárskej práce bolo pripraviť tutoriál pre študentov stredných škôl, ktorí by mali záujem zapojiť sa do medzinárodnej súťaže Robocup@Home v kategórii Robocup@Home Education. Zároveň sme chceli ukázať, predviesť a pripraviť niekoľko netriviálnych a komplexných ukážok práce s robotom, ktoré sú relevantné pre výzvy v tejto súťažnej kategórii.

Takisto bolo treba preskúmať a predstaviť systém ROS, ktorý zabezpečuje komunikáciu a fungovanie jednotlivých komponentov a je populárnym systémom na vývoj v oblasti robotiky. Ciele sa nám naplniť podarilo a vzniklo zopár programov, ktoré predvádzajú schopnosti robota, akými sú počítačové videnie, snímanie okolia pomocou senzorov, interakcia s používateľom použitím mikrofónu a reproduktorov a manipulácia objektov pomocou robotického ramena.

Každá z ukážok má veľký potenciál na implementáciu rôznych vylepšení v budúcnosti. Ukážky by sa tak dali algoritmicky zefektívniť, zrýchliť a spresniť, rozšíriť o ďalšie ciele, vylepšiť a integrovať do väčšieho a komplexnejšieho systému. Ako základ a úvod do práce s robotom Jupiter poskytujú ukážky dostatok informácií a príkladov využitia na plynulé začatie práce a vývoja s robotom a zapojenie sa do súťaže v danej kategórii.

Cieľovou skupinou tejto bakalárskej práce boli žiaci stredných škôl. Myslím si však zo skúsenosti jednak osobnej, ako aj zo skúsenosti z účasti na súťaži Robocup Junior Slovensko 2023 v Prešove, že účasť na Robocup@Home Education je vedomostne vhodná skôr pre študentov prvých ročníkov vysokých škôl s technickým zameraním. Koncepty, ktoré systém ROS využíva a jednotlivé ukážky tohto systému sú podľa môjho názoru dosť náročné na pochopenie pre žiakov stredných škôl, ktorí vzhľadom k ich veku nemajú zatiaľ potrebný vedomostný základ na prácu s nimi. Preto si myslím, že náš tutoriál využijú skôr študenti vysokých škôl a vyučujúci v oblasti robotiky a umelej inteligencie na Katedre aplikovanej informatiky.

Robot Jupiter nebol v úvode našej práce s ním v stopercentne funkčnom stave, jeho oprava nám spolu s vedúcim tejto práce trvala určitú dobu, počas ktorej sme boli nútení kontaktovať výrobcov robota v Singapore. Napriek tomu si myslím, že sme v práci

preukázali robotov potenciál, základné ukážky práce s ním a tutoriál, ktorý bude nemalým prínosom pre študentov, ktorí sa chcú zapojiť do súťaže Robocup@Home Education ako aj pri výučbe v oblasti robotiky a umelej inteligencie na Katedre aplikovanej informatiky alebo pre záujemcov o vývoj a prácu s robotom Jupiter.

6 Literatúra

- [1] Van Rossum, G. (1995). Python tutorial. Technical Report CS-R9527, Centrum voor Wiskunde en Informatica (CWI), Amsterdam.
- [2] Abadi, M., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI).
- [3] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.
- [4] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
- [5] Pygame Community. (n.d.). Pygame Documentation. Dostupné na: <https://www.pygame.org/docs/> Navštívené dňa: [23.05.2023].
- [6] SpeechRecognition. (n.d.). SpeechRecognition 3.8.1. Dostupné na: <https://pypi.org/project/SpeechRecognition/> Navštívené dňa: [23.05.2023].
- [7] gTTS (Google Text-to-Speech). (n.d.). Dostupné na: <https://gtts.readthedocs.io/en/latest/> Navštívené dňa: [23.05.2023].
- [8] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.
- [10] AlexeyAB's GitHub repository. Dostupné na: <https://github.com/AlexeyAB/darknet/> Navštívené dňa: [23.05.2023]
- [11] Raspberry Pi Foundation. (n.d.). Raspberry Pi 4 Model B. Dostupné na: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/> Navštívené dňa: [23.05.2023].
- [12] TensorFlow. (n.d.). Dostupné na: <https://www.tensorflow.org/> Navštívené dňa: [23.05.2023].
- [13] Coral USB Accelerator. (n.d.). Dostupné na: <https://coral.ai/products/accelerator> Navštívené dňa: [23.05.2023].

- [14] iCLEBO. "iCLEBO Kobuki Documentation." Dostupné na: <https://iclebo-kobuki.readthedocs.io>. Navštívené dňa: [23.05.2023].
- [15] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). "ROS: an open-source Robot Operating System." In ICRA Workshop on Open Source Software (Vol. 3, No. 3.2, pp. 5-15).
- [16] ROS Wiki. Dostupné na: <https://wiki.ros.org> . Navštívené dňa: [23.05.2023].
- [17] By RCraig09 - Own work, CC BY-SA 4.0, Dostupné na: https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/20200501_Time_of_flight.svg/1920px-20200501_Time_of_flight.svg.png . Navštívené dňa: [01.06.2023]

Príloha A: obsah elektronickej prílohy

Prvá elektronickej príloha priložená k bakalárskej práci obsahuje zdrojový kód programu. Zdrojový kód možno takisto nájsť na URL adrese: https://github.com/Robotics-DAI-FMFI-UK/jupiter/tree/main/matus/matus_showcase

Príloha B: Tutoriál k robotovi Jupiter

Druhá elektronická príloha priložená k bakalárskej práci obsahuje tutoriál k robotovi Jupiter a ukázkam z tejto práce. Tutoriál sa taktiež nachádza na URL adrese: <https://github.com/Robotics-DAI-FMFI-UK/jupiter/wiki>