TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Využití dostupných senzorů robota Nao pro detekci objektů a mapování okolí

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Miroslav Eichler**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Use available Nao robot sensors to detect objects and map surroundings

Bachelor thesis

Study programme: B2612 – Electrotechnology and informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Miroslav Eichler**
Supervisor: Ing. Miroslav Holada, Ph.D.

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2017/2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav Eichler**
Osobní číslo:	**M15000089**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Využití dostupných senzorů robota Nao pro detekci objektů a mapování okolí**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s robotem NAO na pracovišti školitele. Zaměřte se na senzorický subsystém.
2. Navrhněte software pro detekci objektů a mapování okolí robota NAO. V rešeršní části uveďte alespoň jeden zahraniční projekt, který se obdobnou problematikou zabývá.
3. Navržený software realizujte a odlaďte.
4. Software otestujte v reálných podmínkách a dosažené výsledky srovnejte s výsledky obdobných projektů.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30-40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. 1. vyd. Praha: BEN - technická literatura, 2005. ISBN 80-7300-141-1. Connectivity ? Aldebaran 2.1.4.13 documentation. SoftBank Robotics Documentation [online]. Dostupné z: http://doc.aldebaran.com/2-1/family/robots/connectivity_nao.html**

[2] **KISUNG, SEO. Using NAO: Introduction to interactive humanoid robots.**

Vedoucí bakalářské práce: **Ing. Miroslav Holada, Ph.D.**
Ústav informačních technologií a elektroniky

Konzultant bakalářské práce: **Ing. Karel Paleček, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **19. října 2017**

Termín odevzdání bakalářské práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2017

Prohlášení

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

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

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

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

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 11.5.2018

Podpis:

Abstrakt

Náplní této práce byl vývoj komplexního programu pro mapování okolí a navigaci, humanoidního robota NAO, k vyznačenému cíli. Ten je specifikován NAO markem a je možnost vyhýbat se překážkám pomocí senzorů a dále manipulovat s objekty. Program je použitelný, jako výukový prostředek pro práci s roboty NAO ve vyšším programovacím jazyce. Základní úlohu lze snadno upravit na novou vyjmutím částí kódu z programu. Vše je řešeno v programovém prostředí Python za podpory implementované knihovny NAOqi SDK od výrobce, jenž je pro účely navigace a manipulace s předmětem ideální.

Klíčová slova:

NAO, robot, navigace, detekování, ultrazvukové senzory, NAO mark

Abstract

The content of this work was the development of a complex program for environmental mapping and navigation of the NAO humanoid robot to the indicated target. This is specified by NAO mark and it is possible to avoid any obstacles with the use of sensors and to manipulate objects. The program is usable as a teaching tool for working with NAO robots in higher programming language. The basic task can be easily adjusted to new task by removing part of the program code. Everything is solved in a programming environment Python, with the support of the NAOqi SDK library from the producer, which is ideal for navigation and manipulation of the object.

Keywords:

NAO, robot, navigation, detection, ultrasonic sensors, NAO mark

Poděkování

Tímto bych rád poděkoval svému vedoucímu práce, panu Ing. Miroslavu Holadovi, Ph.D., za zapůjčení robotů NAO a za cenné informace a rady, které byly poskytnuty v rámci konzultací, a také za prostory, jenž byly k tomuto účelu poskytnuty. Dále bych rád poděkoval svému konzultantovi, panu Ing. Karlu Palečkovi, Ph.D., za konzultace a výpomoc při dnech otevřených dveří TUL, na nichž byli roboti prezentováni. Poděkování patří také Anně Molové, která poskytla pomoc při gramatické kontrole práce.

Obsah

Seznam obrázků

Seznam zkratek

2D	Dvoudimenzionální
3D	Třídimenzionální
fps	Frames Per Second (snímkovací frekvence)
FSR	Force Sensitive Resistors (rezistory citlivé na sílu)
IEEE	Institute of Electrical and Electronics Engineers (institut pro elektrotechnické a elektronické inženýrství)
IP	Internet Protocol (identifikační síťové rozhraní)
IR	Infra Red (infra červený)
LAN	Local Area Network (lokální síť)
Li-Ion	Lithium-iontový
LED	Light-Emitting Diode (světlo vyzařující dioda)
Mpx	Mega-pixel (počet obrazových buněk v milionech)
PLC	Programmable Logic Controller (programovatelný logický automat)
QR	Quick Response (kód rychlé reakce)
RAM	Random Access Memory (paměť s náhodným přístupem)
RGB	Red-Green-Blue (červeno-zeleno-modrá)
SDK	Software Development Kit (sada vývojových nástrojů)
SONAR	SOund Navigation And Ranging (zvuková navigace a zaměřování)
TUL	Technická univerzita v Liberci
USB	Universal Serial Bus (univerzální sériová sběrnice)
Wi-Fi	Wireless Fidelity („bezdrátová věrnost“)

Předmluva

Velký rozmach robotiky v posledních letech vyvolali především humanoidní roboti. Průmysloví roboti jsou dnes již standardem ve většině podnicích, kde je vyžadována rychlá, přesná a často i nebezpečná a náročná manuální práce. Zastoupeni jsou různými manipulátory, robotickými rameny, nebo robotickými pažemi s několika stupni volnosti.

Do popředí se dostávají humanoidní roboti, tvarem i chováním podobající se člověku, kteří nacházejí uplatnění především v domácnosti, kde mohou pomoct hendikepovaným, postarat se o domácnost, nebo být sexuálními společníky. V nedávné době to mohlo znít jako „hudba budoucnosti“, avšak v dnešní době začínají být humanoidní roboti stále více reální a pracuje se na jejich vývoji a propojení s dalšími službami a možnostmi usnadnění našeho života. Právě částečné usnadnění našeho života bude náplní této bakalářské práce.

Úvod

Bakalářská práce je věnována programování humanoidního robota NAO pro potřeby školy. Práce by dále mohla sloužit, jako pomoc budoucím uživatelům při vytváření vlastního programu pro realizaci libovolné úlohy typu „pick and place“ pro roboty NAO v programovacím jazyce Python.

Jelikož nebyla možnost se s tímto typem robota v minulosti setkat, bylo podstatnou částí tohoto projektu se s ním nejdříve seznámit a naučit se jej programovat. Dále bylo potřeba si prostudovat projekty, které v minulosti byly na těchto robotech realizovány, neboť na ně bylo v práci částečně navázáno.

Náplní této práce je využití dostupných senzorů, jimiž robot disponuje pro detekci objektů a mapování okolí. Návrh a odladění softwaru by měl být realizován v laboratoři a vyzkoušen na obecné úloze. Následně by měl být tento kód použitelný, po úpravě a vyladění, na konkrétní úlohu. Program pro obecnou úlohu by měl být realizován tak, aby bylo možné z něj poté čerpat a vyjmout části kódu za účelem tvorby konkrétní úlohy, kde bude využito co nejvíce funkcí a schopností robota.

Robot by měl pomocí kamery vyhledat červený míček a dostat se k němu, přičemž by se měl po cestě vyhýbat překážkám díky ultrazvukovým senzorům. Míček by měl být uchopen a pro kontrolu detekován. Poté by mělo být nalezeno místo, kam by měl robot dojít a vhodit míček do nádoby a následně detekovat, zdali se v ní míček opravdu nachází. Pro demonstraci širšího použití kódu a možnosti úpravy by měly vzniknout další konkrétní úlohy vycházející z obecné.

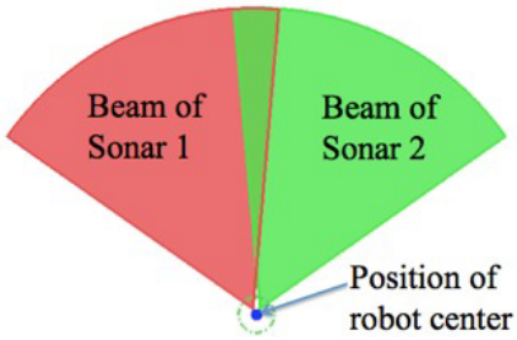
Obrázek 1: Roboti NAO, vlevo starší verze, vpravo novější

1 Stávající stav mapování okolí pomocí robotů NAO

Mapováním okolí se vyznačuje činnost, při které je využito kombinace různých senzorů, často odlišné specifikace, díky nímž je možné detekovat okolní prostředí robota a na základě vyhodnocených dat se rozhodnout, po jaké trajektorii se vydat k cíli. Na toto téma bylo napsáno již několik prací, z nichž některé budou v této kapitole „vyzdvihnuty“, aby měl čtenář povědomí o stávajícím stavu v tomto odvětví.

1.1 Navigování humanoidního robota v neznámém prostředí s detekcí překážek

Práce s originálním názvem „Humanoid Robot Navigation and Obstacle Avoidance in Unknown Environments“ pojednává o tvorbě propracovaného algoritmu GODZILA, určeného pro navigaci mezi překážkami v neznámém prostředí a zabránění kolize s nimi. Detekce překážek je založena na snímání okolí za pomoci dvou ultrazvukových senzorů (také sonarů), jenž jsou součástí robota.[2]



Obrázek 1.1: Prostor snímaný ultrazvukovými senzory [2]

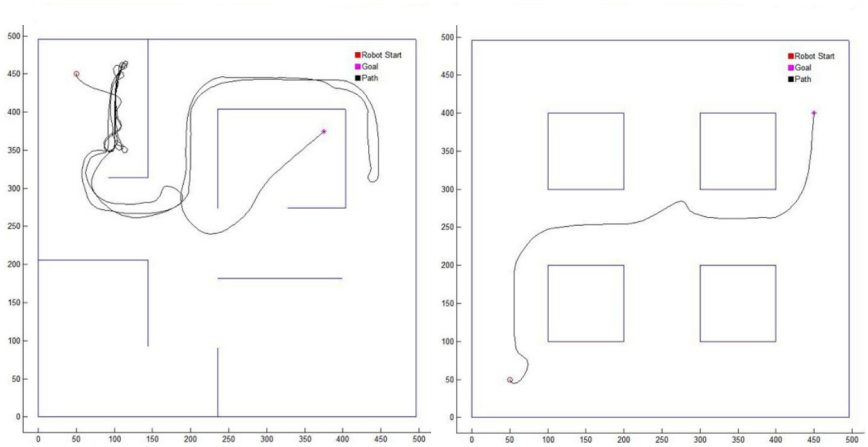
Navigace je zcela autonomní bez nutnosti využití překážkové mapy, v níž je nadefinován, nebo nasnímán prostor, ve kterém dochází k navádění robota. Tato mapa zvyšuje nároky na výpočetní výkon a je obvykle získána snímáním prostoru laserovou hlavou, která je nadstandardní výbavou robota.

V tomto případě je tedy možné navigovat k viditelnému či náhodnému cíli, jenž není na startu navigace viditelný. Navigace pak probíhá náhodnými pohyby, dokud

cíl není spatřen. Je zde kladen důraz na možnost získání kvalitních dat z ne příliš přesných, ultrazvukových snímačů, které mají široký sonarový svazek, kde není k dispozici žádná úhlová informace. Tato nedokonalost snímání sonary je částečně optimalizována měřením pomocí několika virtuálních sonarů. Díky tomu je získána vyšší přesnost. Spočívá to v částečném překrývání dvou paprsků, z nichž každý snímá prostor o úhlu 60°, jak je vidět na obr. 1.1 Je tak možné, díky postupnému pootáčení robota a snímáním okolního prostoru, nalézt předmět hranou paprsku. To má za následek přesnější detekci, díky které je možné hrubě určit, jak vzdálenost překážky, tak i úhel, pod kterým je snímána. Díky tomu nabývá navigační algoritmus na robustnosti.[2]

Hodnoty získané ze sonarů jsou váhově využity v daném algoritmu. Překážky detekované v malé vzdálenosti mají velkou váhu, naopak ty vzdálenější mají malou váhu s důrazem na změnu směru. Pomocí váhového využití hodnot o vzdálenosti je trasa k cíli optimalizována. Robot pak zmatečně nekličkuje mezi překážkami, nýbrž se jim plynule vyhýbá. Pokud je cíl viditelný, tak je nejprve zjištěna fiktivní cesta po přímce, která je dále váhově optimalizována podle údajů ze sonarů.[2]

Program byl testován v různém prostředí s odlišnou zástavbou překážek. Některá se podobala bludišti, jak je vidět na obr. 1.2 a i přes to se robot dokázal v prostředí orientovat a najít cílové místo, které na startu neviděl. Stejně tak obstojně dokázal navigovat k cíli v podobě červené značky, kterou viděl již od startu.[2]

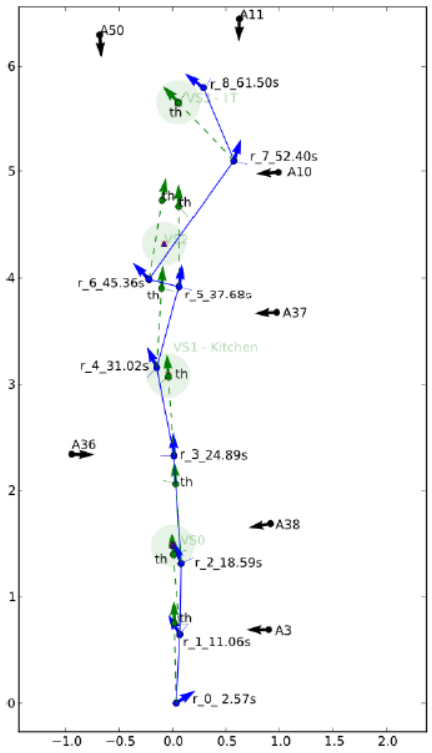


Obrázek 1.2: Prostředí, v němž probíhalo testování navigačního algoritmu GODZILA. Čárou je vyznačena trajektorie pohybu robota.[2]

Algoritmus GODZILA byl již dříve využit pro autonomní vozidla a stojí za ním několikaletý týmový vývoj. Je optimalizován tak, aby měl co nejmenší paměťové a výpočetní nároky. Tento propracovaný algoritmus bohužel není veřejně dostupný a nelze na něj nijak navázat v této práci.[2]

1.2 Interiérová navigace založená na detekování 2D kódů

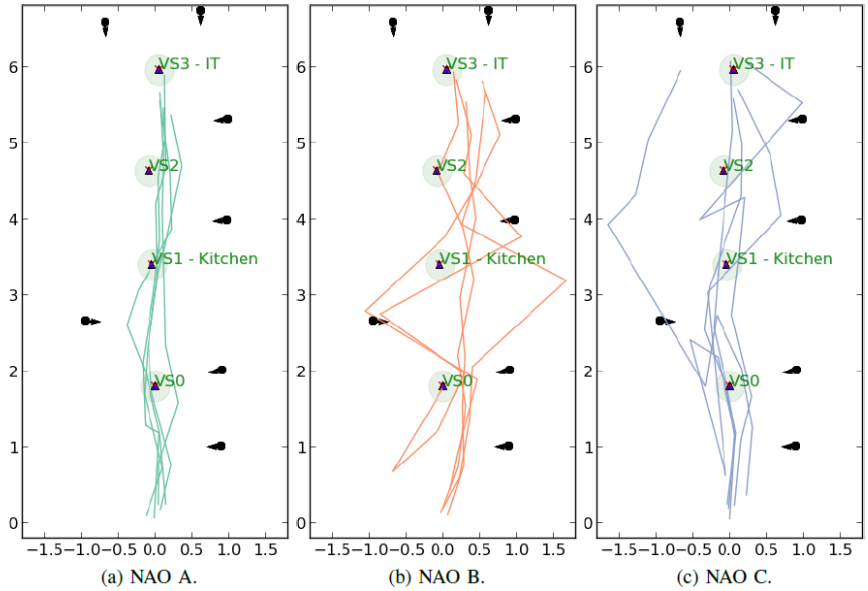
Práce s originálním názvem „Humanoid Robot Indoor Navigation Based on 2D Bar Codes: Application to the NAO Robot“ se věnuje interiérové navigaci založené na identifikaci 2D bar kódů umístěných na stěnách bez využití odometru. Robot NAO se orientuje v prostoru pouze pomocí kódů a sám si zvolí nejkratší možnou cestu, kterou se dostane k cíli. Práce je založena na dosažení cíle za co nejkratší čas, aby autoři překonali jinou práci, jež se v minulosti věnovala stejnému tématu. V této aplikaci robot nepotřebuje komunikovat s externím počítačem a všechny výpočty se dějí v procesoru robota. I přesto je systém schopen se rozhodnout za 1 vteřinu s tím, že program je optimalizován pro rychlé rozhodování, kam se vydat. Robot je sice schopen detekovat NAO marky, ale zde je z experimentálních důvodů využita implementace OpenCV pro detekci obrazu, čímž jsou rozpoznávány QR kódy o maticové struktuře. Po detekování kódu je odhadnuta poloha následujícího kódu, což je vidět na obr. 1.3 Autoři k tomu přistoupili z toho důvodu, že detekování QR kódů je mnohem přesnější a rychlejší, než v případě NAO marků.[3]



Obrázek 1.3: Trajektorie robota blížící se ideálně přímce. Šipky zobrazují odhadnutou polohu následující značky.[3]

Program byl vyzkoušen na třech shodných robotech, přičemž každý z nich se rozhodoval jinak a vydal se po jiné trajektorii. Je to zapříčiněno především absencí odometru, jenž by korigoval robota v globálním souřadném systému. Právě proto je

chůze nepřesná, neboť není využito zpětné vazby. Orientace probíhá pouze na základě kódů. Použití roboti se lišili stářím, což se výrazně podepsalo na době, za jakou urazili vyznačený koridor o délce 6 m. Nový robot, který měl nízkou úroveň opotřebení kloubů, zvládl projít koridor s výrazně kratším časem, než ostatní, starší roboti. Ve výsledku dosáhl nový robot (NAO A) odchylky od teoretické nejkratší cesty jen 3 % , kdežto u opotřebenějších robotů dosáhla chyba 33 a 36 % . Je to vidět na obrázku č. 1.4, kde je jasně patrný odlišný pohyb robotů NAO B a NAO C vůči NAO A držícího se poblíž roviny teoretické cesty. Aby robot dosáhl co nejrychlejšího času, tak se stále pohybuje kupředu a chyby vznikají tím, že se natáčí za značkami, aby je správně detekoval a právě proto kličkuje.[3]



Obrázek 1.4: Odlišné trajektorie pro různě opotřebené roboty [3]

K využití projektu v reálném životě autoři testovali roboty v kancelářích s velkým množstvím nábytku a nerovnoměrného osvětlení na dráze dlouhé 15 m. Zde měli roboti přejít z první místnosti, osvětlené umělým osvětlením, skrz dveře do druhé místnosti, kde bylo velké okno omezující možnosti snímání okolní scény. I přes to se s touto nástrahou roboti vypořádali a dosáhli cíle s malým zaváháním. Stalo se tak poté, co se jeden z nich dostal do kolize s nábytkem a upadl, načež vstal a pokračoval v navigaci. Zde se opět projevila výhoda nového robota, který zvládl celou trasu o poznání rychleji.[3]

2 Popis robota NAO

Jedná se o autonomního programovatelného robota vyvinutého francouzskou společností Aldebaran Robotics, kterou od roku 2015 spravuje japonská společnost SoftBank Robotics. Projekt byl spuštěn roku 2004 a v roce 2007 se objevili první roboti NAO na soutěži RoboCup, kde hráli fotbal. Na veřejnost se první verze dostala v roce 2008 pod názvem Nao Academics Edition.

Své využití robot nachází především pro vzdělávací a výzkumné účely s tím, že se využívá v akademických institucích po celém světě. K tomu byl také navrhnut, neboť účelem tohoto projektu bylo poskytnout jej pro potřeby výuky studentů a díky tomu na různých prezentačních akcích seznámit širokou veřejnost s robotikou.[6]

Obrázek 2.1: Robot NAO od firmy Aldebaran Robotics

Do roku 2015 bylo prodáno více než 5000 kusů. NAOvými následníky jsou Pepper a Romeo. Důvodem toho, že NAO není v dnešní době masově rozšířen a lze se s ním setkat spíše na univerzitách je cena, která převyšuje 100 tisíc korun. Dále jsou to omezené možnosti programování pro laiky, neboť v Choreographe, kde se dají programovat pouze základní funkce robota, toho nelze udělat mnoho. Pro maximalizování využitelnosti robota a jeho možností je třeba využít vyšších programovacích jazyků, jako jsou C++, nebo Python. Až se časem zvýší konkurence a poptávka, cena jistě klesne na přijatelnou úroveň a humanoidní roboti se dostanou více do popředí.[6]

2.1 Parametry robota

NAO není nikterak velký, na výšku měří 58 cm a váží 4,3 kg. Tělo robota je včetně převodů vyrobeno z plastu. Pohon kloubů zajišťují servomotory a celkem má 25 stupňů volnosti. Mozkem robota je procesor Intel Atom Z530 s taktem 1,6 GHz doplněným o 1GB operační paměť RAM. Interní paměť flash má velikost 2GB a lze ji doplnit paměťovou kartou o velikosti až 8 GB. Ovládání je zajištěno operačním systémem Linux založeným na Gentoo. Pohání jej Li-Ion baterie o kapacitě 48,6 Wh se kterou je schopen být v provozu na jedno nabití až 90 minut a aktivně fungovat zhruba hodinu. Programovat jej lze v jazycích C++, Python, Matlab, Java a .NET. Ovšem největší podporu ze strany výrobce, co se týče dokumentace, má v Pythonu. [6]

2.2 Komunikace s jiným zařízením

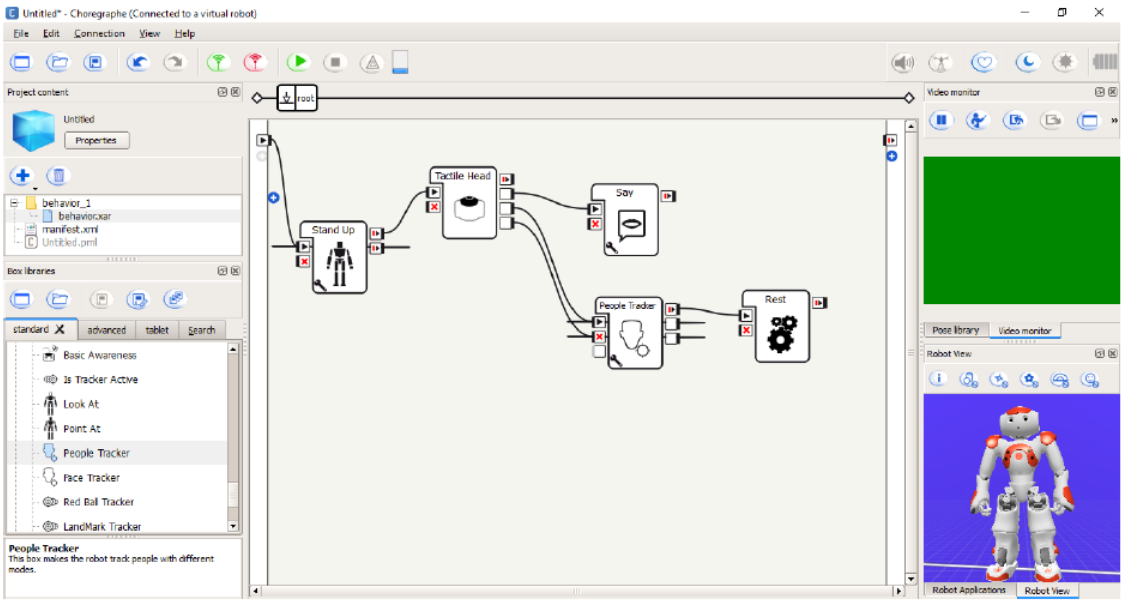
Pro možnosti komunikace s okolím je robot vybaven standardním ethernetovým rozhraním s konektorem RJ-45 umožňujícím rychlost přenosu až 1000 Mb/s a pro bezdrátovou komunikaci Wi-Fi modulem podporujícím IEEE 802.11 a/b/g/n, který je výhodnější při komunikaci s PC. Přes USB port lze připojit rozšiřující periferie sloužící pro zdokonalení některých funkcí, nebo pro lepší senzorické schopnosti pro sledování okolí. Příkladem může být Microsoft Kinect sloužící pro sledování pohybů člověka, nebo Asus 3D senzor pro mapování okolí.[1][6]

2.3 Možnosti programování robota

Programování robota není primárně určeno pro laiky. Jsou zde dvě zcela odlišné možnosti, jak k programování přistupovat. První možností je využití jednoduššího softwaru přímo od výrobce. Jedná se o Choreographe a je určen pro začátečníky, kteří se ještě ve velké míře nesetkali s programováním. Tou druhou možností je vyšší programovací jazyk, na což je potřeba mít s programováním již nějaké zkušenosti.[1]

2.3.1 Prostředí Choreographe

Jedná se o nejjednodušší způsob, jak lze NAO programovat. Tato výhoda s sebou ovšem přináší určité omezení při návrhu aplikací a nemožnost využití všech funkcí, jenž robot nabízí. Nicméně pro základní použití je software využitelný a dovoluje vytvářet program, ovládat robota, simulovat některé funkce na 3D modelu robota a kontrolovat základní parametry, jako je kvalita připojení, běh programu a stav baterie. K programování je využit grafický jazyk podobný ladder diagramu, ve kterém se programují PLC automaty.[1][8]



Obrázek 2.2: Prostředí Choreographe sloužící pro programování a ovládání robota

Tento jazyk je velice jednoduchý na pochopení a hodí se pro lidi bez zkušeností s programováním. Používají se předem definované bloky, které se spojují a vytváří se tak program. Choreograph je velice přehledný a nevýhody objevující se při programování v tomto prostředí se obrátí ve výhody při sledování běžícího programu, jenž je graficky znázorněn vybarvením bloků s aktuálně vykonávanou částí programu. Zároveň je možné sledovat i obrazový výstup z kamery v reálném čase. Další možností je grafická simulace programu, ke které lze využít virtuálního robota. Je tak možné nasimulovat vytvořený program bez přítomnosti robota. Program lze nahrát do robota a spouštět jej přímo ovládacími tlačítky na hlavě, nebo je možný spustit z počítače připojením přes Wi-Fi, nebo LAN kabel. Právě předpřipravené bloky jsou limitující pro stavbu programu. Lze zde jednoduše vytvořit např. chůzi spojením bloků „StandUp“, „WalkTo“ a „SitDown“. U toho může NAO ještě něco říct, pakliže je přidán blok „Say“. Větší možnosti nabízí programovatelný blok, v němž se dají psát příkazy v Pythonu, ale i zde jsou limitující možnosti. Pokud je tedy potřeba využít co nejvíce schopností robota ke sledování okolí, navigaci a manipulaci s předměty, tak toho v tomto prostředí není možné docílit.[8]

2.3.2 Vyšší programovací jazyk

Využitím jednoho z vyšších podporovaných jazyků, v tomto případě Pythonu, lze dosáhnout vyšší efektivity programování a využití všech funkcí. Lze tak odstranit nedostatky Choreographe popisované výše. Jedinou nevýhodou je nemožnost simulace a horší sledování běhu programu. Do Pythonu je potřeba doinstalovat výrobcem distribuovaný vývojový nástroj NAOqi SDK obsahující knihovny pro používání všech funkcí robota a pro přístup k senzorům. Lze se tak pomocí několika příkazů v hlavičce programu spojit s robotem a získat nad ním plnou kontrolu. Základem je vložit do programu správnou IP adresu a port robota, s nimiž následně pracují všechny knihovny. Po vyladění programu jsou dvě možnosti, jak program spouštět.[1]

Je zde možnost program nahrát do paměti robota a pak jej jen spouštět a ovládat za pomoci dostupných tlačítek. Nevýhodou tohoto řešení je, že parametry robota jsou již zastaralé, takže není dostupný dostatečný výpočetní výkon pro složitější operace, jako třeba práce s obrazem a rozpoznávání předmětů. K tomu je zapotřebí druhé možnosti spouštění programu, a to přímo z počítače.

Spuštěním programu z počítače dojde k navázání komunikace za pomoci stejné IP adresy a poté již běží program na PC s tím, že robot dostává pouze příkazy, co má dělat a o všechny výpočty se stará počítač. Robot vykonává pouze senzorické a aktuační funkce. Nezatěžuje se tak jeho procesor a baterie, která by při plném vytížení měla menší výdrž.

2.4 Senzorika robota

Senzorické schopnosti robota jsou velice komplexní, proto je velice výhodné jich využít pro orientaci v prostoru, detekci překážek a pro vyhledávání předmětů. Jednotlivé senzory nejsou dokonalé ve všech oblastech, ale jejich kombinací je možné dosáhnout velice dobrých orientačních schopností. Příkladem může být nemožnost určení vzdálenosti pod určitou úroveň pomocí ultrazvukových senzorů, což kompenzuje snímání NAO marků pomocí kamery a vyhodnocení jejich úhlové velikosti. Dalším příkladem může být špatná detekce překážky pod nohama, kterou robot nerozpozná kamerou. Robot by mohl zakopnout a upadnout, zde však zafungují nárazníky ve špičkách nohou, jenž vyhodnotí kopnutí do překážky a robot se zastaví a udrží stabilitu. To vše je třeba kombinovat pro dosažení optimálních senzorických a vyhodnocovacích schopností.

2.5 Ultrazvukový senzor

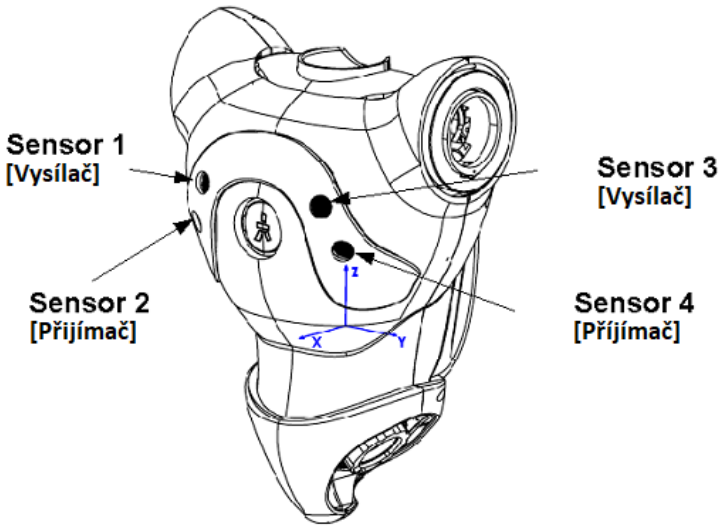
Robot Nao je vybaven dvěma ultrazvukovými senzory (sonary) na hrudi, sloužícími pro vysílání vysokofrekvenčních zvukových vln o frekvenci 40 kHz. Nahoře jsou vysílače a pod nimi jsou umístěny dva přijímače (viditelné na obr. 2.3) sloužící pro zachytávání echa získaného odrazem od překážky. Údaje o vzdálenosti jsou obnovovány každých 100 ms. Údaje jsou již předzpracovány a díky tomu je k dispozici

hodnota vzdálenosti od předmětu, nikoliv čas mezi vysláním a příjmem signálu. Robot může využít tyto senzory k měření vzdálenosti od překážky, nebo k orientaci v prostoru využitím obou sonarů umístěných na pravé a levé straně hrudi zároveň. Výhoda takto umístěných senzorů spočívá v tom, že vyhodnocením rozdílu vzdálenosti z obou snímačů je možné určit, zda se nachází překážka před robotem vpravo, či vlevo, popřípadě natočení k ploše, jako je třeba zeď, nebo nějaký větší předmět.[1]

Využitím těchto senzorů je možné získat velice přesné údaje o vzdálenosti v rozpětí od 0,25 m do 2,55 m s přesností 1 až 4 cm v závislosti na vzdálenosti. Pokud je vzdálenost menší než 0,25 m, už není možné určit, jak daleko se objekt nachází. Robot pouze ví, že před ním objekt je.

Robot má optimalizovaný algoritmus pro vyhodnocení vzdálenosti z devíti po sobě jdoucích vln, z čehož dělá průměr. Je možnost se dostat jednotlivě k získaným devíti vzorkům, ovšem už není možné určit, která z vln se odrazila od hledaného předmětu.

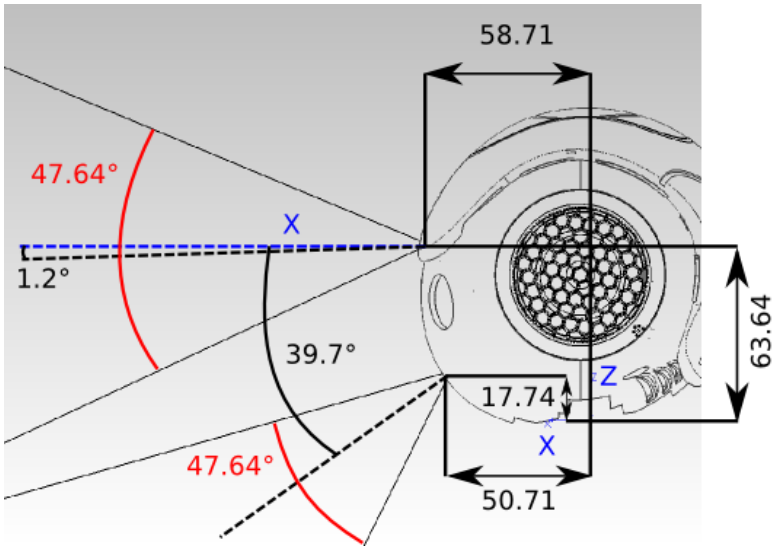
V praxi tedy už nastává problém u vzdáleností pod 0,4 m, protože většina odražených vln může přicházet ze země, která se nachází v podobné vzdálenosti od snímačů. Stejně tak je třeba dát si pozor, aby senzor nedetekoval echo od pohybujících se paží.[1]



Obrázek 2.3: Ultrazvukové senzory umístěné na hrudi [1]

2.6 Kamery

Na hlavě jsou umístěny dvě kamery, které se navzájem doplňují, aby pokryly velký úhel ve vertikálním směru, jak je vidět na obr. 2.4. Natočení hlavového kloubu v tomto směru je totiž omezené, což se o horizontálním pohybu říci nedá, protože je zde možnost natočení o 120° na každou stranu. Obě kamery jsou umístěny mezi očima.[1]



Obrázek 2.4: Vertikální zorné úhly obou kamer [1]

Primární kamera s rozlišením 1,22 Mpx se nachází na čele a sekundární s totožným rozlišením je umístěná v oblasti úst. Snímkovací frekvence kamer je 30 fps. Pomocí hlavní kamery je snímán prostor před robotem a je využívána ve většině případů.[1]

Sekundární kamera nachází své uplatnění např. při sbírání předmětů bezprostředně před robotem, nebo při detekci hrany, či schodu, kdy je sledován prostor pod nohama, aby robot nezakopl.[1]

Mezi kamerami lze libovolně přepínat, ovšem některé předem připravené funkce, jako např. vyhledávání červeného míčku, nebo NAO marků, využívají pouze primární kameru. Takto získaný obrazový záznam není nikterak kvalitní, neboť kamery mají problém s horšími světelnými podmínkami. Je pak problém detekovat předměty, či NAO marky na větší vzdálenost, neboť je zde zastoupeno velké množství šumu. V praxi je problematické detekovat předmět, pokud se za ním nachází roztažené okno. Běžnou hlavu je možné zaměnit za hlavu se stereo viděním, kde jsou kamery umístěné vedle sebe místo očí.

2.7 IR senzor

Namísto očí má robot infračervené diody jako vysílač a přijímač. Díky tomu jej lze ovládat dálkovým ovladačem a spouštět tak na něm uložené programy. Stejně tak lze i pomocí něj jiné zařízení ovládat. Robot tak může třeba zapnout televizi a přepínat kanály. Problém je však s kompatibilitou ovladačů. Ne s každým ovladačem NAO spolupracuje. Nejlépe toho lze využít pro komunikaci mezi roboty, která ale bude také velice nepraktická. Zde se nabízí pouze komunikace z očí do očí a to za předpokladu, že mezi roboty nebude žádná překážka a zároveň nesmí dojít k pohybu rukou v okolí hlavy, protože by došlo k přerušení optické komunikace. Vše tedy

naznačuje tomu, že tento způsob komunikace mezi dvěma roboty nebude nejvydařenější, avšak je to asi nejlepší způsob, jak infračervené senzory využít. Mnohem lepší komunikaci nabídne Wi-Fi už jen z toho důvodu, že můžou vzájemně komunikovat více než 2 zařízení a roboti si nemusí vidět do očí.[1]

2.8 Signalizační LED

Po těle robota jsou umístěny různé signalizační diody sloužící pro vizuální kontakt. V každém oku je kolem IR diod umístěno 8 RGB LED diod. Dalších 10 diod je kolem každého reproduktoru a 12 jich je na vrchní straně hlavy. Dále jsou diody na chodidlech, signalizující smáčknutí nárazníku. Také hlavní vypínací tlačítko je podsvícené a signalizuje některé stavy robota, jako je vypínání, zapínání, stav připravení k použití a stav nabití baterie.[1]

2.9 Mikrofony

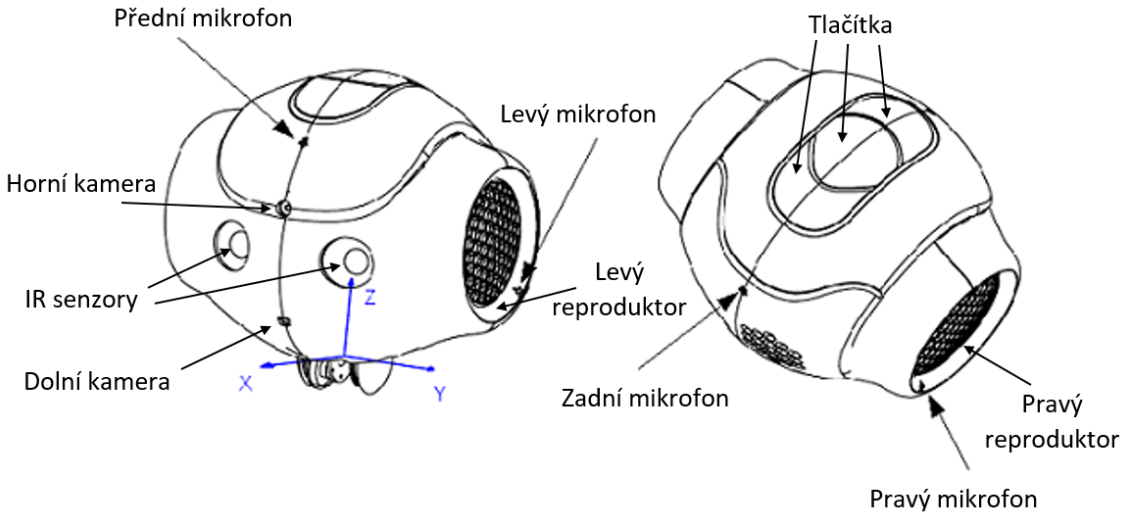
Po obvodu hlavy jsou umístěny 4 mikrofony. Vepředu, vzadu a po stranách hlavy, jak je vidět na obr. 2.5. Díky tomu je NAO schopný detekovat odkud zvuk přichází. To lze využít např. k chůzi za zvukem. Dále je možné říkat robotovi příkazy, na které bude patřičně reagovat. Snímané pásmo zvuku je od 300 Hz do 8 kHz.[1]

2.10 Reproduktory

Namísto uší má NAO stereo reproduktory, které lze použít k přehrávání zvuku, hudby a hlavně k syntetizaci řeči. Robot má zabudovanou syntetizační jednotku, díky které je schopný převodu textu na řeč a komunikace s člověkem. Lze zvolit z velkého množství jazyků obsahujících i Český jazyk.[1]

2.11 Ovládací tlačítka

Na těle se nachází několik uživatelských tlačítek sloužících pro ovládání robota. Pomocí nich lze spouštět předem nahrané programy, nebo je následně ukončovat. Na hlavě se nachází tři dotyková tlačítka, která se dají dobře využít k volbě více programů a vstupu do několika úrovňového menu, jenž lze vytvořit. Další dotyková tlačítka jsou k dispozici na rukou, a to 3 na každé z nich (pouze ve verzi H25). Na hrudi je umístěno hlavní mechanické tlačítko sloužící k zapnutí a vypnutí robota. Pokud je krátce stisknuto, robot řekne své základní údaje, jako chybové hlášky a IP adresu zařízení. Poslední neméně důležitá tlačítka jsou mechanická v podobě nárazníků na špičkách nohou. Pod nárazníkem jsou dvě tlačítka. Na každé straně nohy jedno a to z toho důvodu, aby robot mohl detekovat, jakou stranou nohy do překážky kopl. Využití by mohly nárazníky najít také při příchodu ke schodu, neboť překážka tohoto typu by byla obtížně detekovatelná.[1]



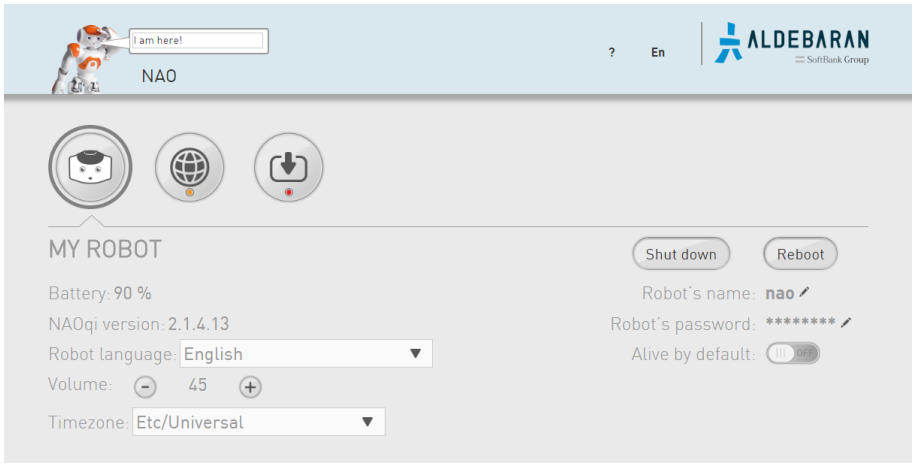
Obrázek 2.5: Popis hlavy robota [1]

2.12 Senzory sloužící pro stabilitu

Pro dobré dynamické vlastnosti při pohybu mezi statickými polohami je třeba neustále vypočítávat pozici robota a udržovat tak jeho stabilitu. K tomu slouží inerciální jednotka využívající 3-osý gyroskop a 3-osý akcelerometr, jenž jsou primárně využívány pro detekci pádu robota, nebo pro zjištění pozice, ve které se nachází torzo. Údaje z nich jsou samozřejmě i uživatelsky dostupné, stejně tak jako z odporových senzorů síly FSR. Ty jsou umístěny v chodidlech a měří sílu, jakou chodidla působí na podložku. Nacházejí se v každém rohu, tedy v každém chodidle 4 a měří sílu od 0 do 25 N. Primárně senzory síly využívá manažer pádu, který když rozpozná, že se obě chodidla nedotýkají země, tedy došlo ke ztrátě rovnováhy, tak robot natáhne ruce před sebe a uvolní klouby (vypne přívod proudu do motorů). Tímto opatřením zmírní následky pádu.[1]

3 Realizace programové části

Prvním úkolem bylo naučit se s robotem zacházet a opatrně s ním manipulovat, neboť to není „laciná hračka“. Po stlačení zapínacího tlačítka se robot zapne a po několika minutách zaujme výchozí polohu v sedě. Opětovným stiskem tlačítka řekne svou IP adresu, která je pokaždé jiná. Ovšem maska sítě (port) zůstává pořád stejná, a to 9559. Po úspěšném navázání komunikace přes Wi-Fi, nebo LAN kabel se lze zadáním IP adresy do prohlížeče dostat na domovskou stránku robota NAO, viditelné na obr. 3.1. Zde je možné kontrolovat stav nabití baterie, aktualizovat firmware robota, nebo napsat text do bubliny, jenž robot po stisku klávesy ENTER řekne.



Obrázek 3.1: Webové prostředí znázorňuje základní údaje o robotu

Nejprve byla snaha robota ovládat přes program Choreographe, který je pro začátek nejlepší volbou. Je možné si vyzkoušet základní pohyby končetin, chůzi, mluvení, vstávání a jednoduše tyto úkony skládá do časové linie. Byla to první možnost vidět chování robota a vyzkoušet si jeho pohyby. Po letmém seznámení se s programem bylo jasné, že s tímto moc daleko nelze postoupit, protože jak již bylo psáno v části 2.3.1 na straně 20, tak zde není ta možnost využít všech dostupných funkcí zařízení. Proto bylo rozhodnuto dále pokračovat ve vyšším programovacím jazyce společně s rozšiřující knihovnou NAOqi SDK určenou přesně pro tyto účely.

Po úspěšném navázání komunikace s robotem přišlo na řadu seznámení se s projekty studentů, kteří s ním pracovali v minulosti. Byly vyzkoušeny jejich programy pro pochopení struktury kódu v jazyce Python, ve kterém bylo rozhodnuto také

programovat i přes to, že nikdy v minulosti nebyla možnost v něm pracovat. Na tuto volbu bylo přistoupeno po prostudování všech možností a zjištění, že pro tento jazyk je nejlépe zpracovaná dokumentace od výrobce a tudíž je i nejčastěji používán uživateli hned po C++.

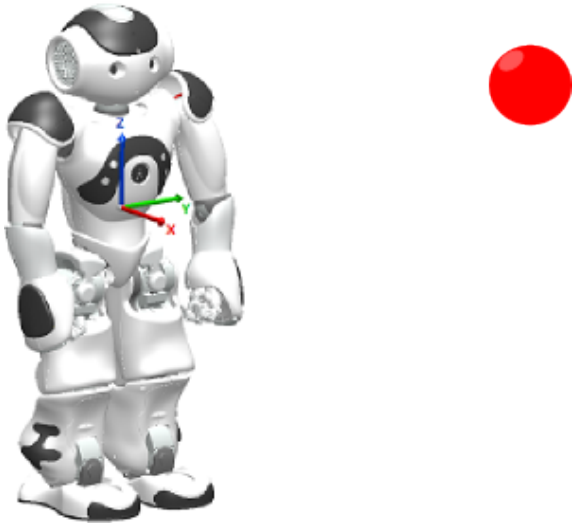
3.1 Navigační a rozpoznávací algoritmy od výrobce

Použít lze několik předem vytvořených ukázkových algoritmů od výrobce, kterým stačí jen předat základní parametry sledovaného objektu.

Sledovat lze :

- `Červený míček`
- `NAO marky`
- `Obličeje`

3.1.1 Sledování červeného míčku



Obrázek 3.2: Detekce červeného pěnového míčku [1]

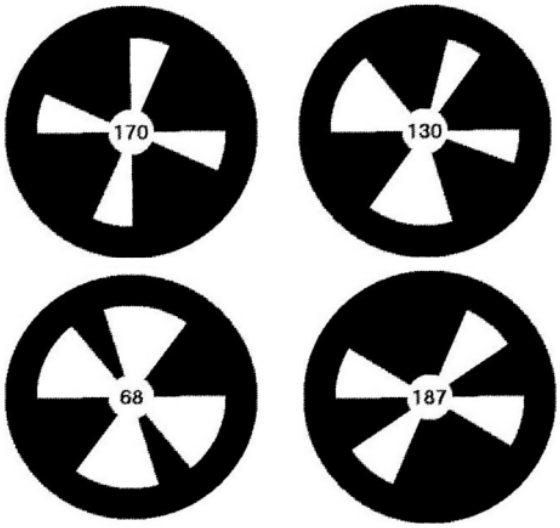
K tomu slouží metoda *ALRedBallDetection()*, která po přijetí parametru velikosti míčku vrací jeho pozici v souřadnicích X, Y a úhlovou velikost míčku v radiánech. Použitím *ALRedBallTracker()* může robot, po zadání stejného parametru, míček sledovat pouze hlavou, kombinací hlava a ruka, nebo chůzí za míčkem. Pro účely navigace k míčku by se hodila chůze za míčkem a posléze sledování míčku pohybem ruky a následným uchopením míčku. Zde ovšem nastal zcela zásadní problém, že do předem vytvořených programů nelze jakkoliv zasahovat. Tyto programy jsou již zkompilované a není ani možnost se dostat do zdrojového kódu a pozměnit ho pro

tyto účely. Třeba sledování míčku rukou funguje tak, že robot má ruku vedle míčku, aby si nezakrýval výhled a stále jej mohl detekovat a kopírovat jeho pohyb.

V případě chůze za míčkem se jedná o stejný případ, kdy robot jde nejkratší možnou cestou a nesleduje přitom okolí a nedetekuje překážky. Zastaví se až na stanovené vzdálenosti před ním. Opět zde nejde vstupovat do spuštěného programu, aby mohly být detekovány překážky. Bylo tedy usouzeno, že tyto demonstrační programy jsou pro účely navigačního algoritmu nepoužitelné a z toho důvodu bylo rozhodnuto o vytvoření vlastního algoritmu pro detekci překážek a vyhledávání předmětů.

3.1.2 Sledování NAO marků

Robot NAO umí detekovat 2D kódy, které může využít k navigaci a orientaci v místnosti. Výrobcem podporované jsou tzv. NAO marky, což jsou kruhové černobílé terčíky viditelné na obr. 3.3, z jejichž středu vychází 4 různě silné paprsky s různou úhlovou orientací. Podle orientace paprsků NAO pozná, jaké číslo marku rozpoznal. Nalepením marků po místnosti mu může být usnadněna orientace, určen referenční bod po zapnutí, nebo podle nich může být navigován na místo určení.



Obrázek 3.3: NAO marky [1]

Pokud je použita metoda *ALLendMarkDetection()*, tak může být detekován jeden, či více marků současně. Vrací se hodnoty, jako počet rozpoznaných značek, jejich číslo, nebo orientace X, Y jednotlivých značek v prostoru a jejich úhlová velikost. Je možné se dostat také k dalším údajům vhodných pro navigaci, jako je horizontální i vertikální úhel detekované značky vůči ose kamery. Stejně jako u míčku je k dispozici chůze za NAO markem, která se neosvědčila ze stejného důvodu, jako v případě míčku.

3.2 Návrh vlastního navigačního algoritmu

Původně bylo uvažováno o částečném využití a upravení již vytvořených navigačních algoritmů, podle vlastních potřeb, pro detekci překážek. Bohužel úprava nebyla možná z již výše jmenovaných důvodů. Proto bylo rozhodnuto o vytvoření vlastního algoritmu a patřičném odladění, aby posléze byl použitelný pro širší působnost a větší využití.

3.2.1 Zjištěné manipulační schopnosti robota

Jako základ byla zvolena tvorba programu pro vyhledávání červeného pěnového míčku a navigace k němu pomocí NAO marků. Tento míček byl zvolen z toho důvodu, že jej jednak podporuje výrobce a dále, že je dobře uchopitelný pro robotovu malou ruku čítající tři prsty. Jelikož je míček pěnový a pružný, tak není třeba příliš řešit sílu stisku. Míček proto může být uchopen maximální možnou silou stisku bez toho, aniž by došlo k poškození uchopeného předmětu, či robotovy ruky. Minimalizuje se tím také riziko možného upuštění předmětu v rámci manipulace s ním.

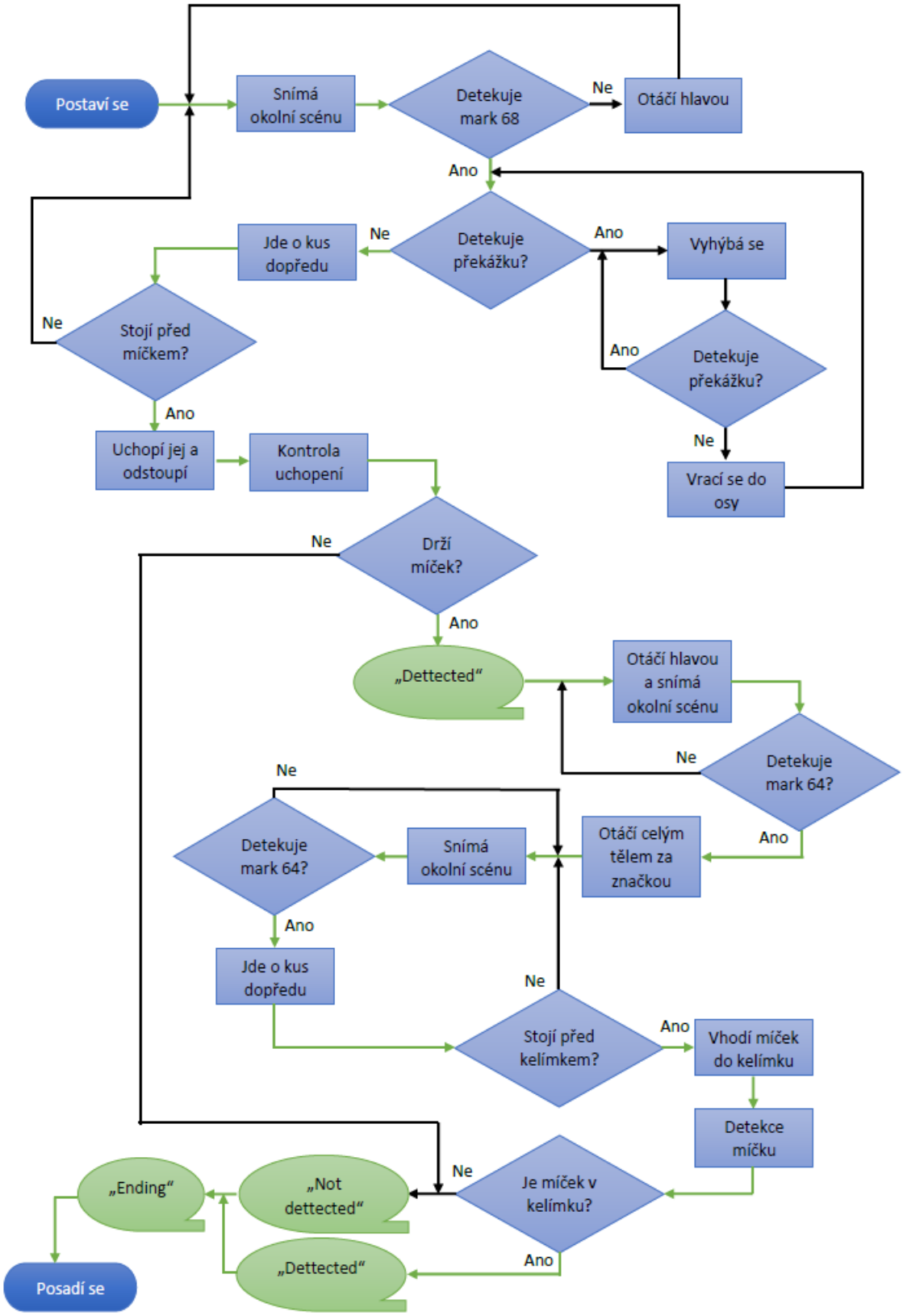
Problém s uchopením neměl robot ani u předmětů malého průměru. Typicky jsou to kancelářské potřeby, nebo sladkosti. U větších předmětů nastal problém jak s uchopením, tak i s manipulací. Zde musí být vyřešen buďto speciální úchop, jako je nějaké držadlo, nebo je potřeba předmět uchopit oběma rukama. Takto velký předmět by měl být lehký, jelikož nosnost robota s nataženými pažemi je velice nízká, protože je třeba udržovat rovnováhu. Pokud se mu podaří předmět uchopit a zvednout, tak dalším problémem je chůze, při které robot pohybem rukou udržuje stabilitu podobně, jako člověk. Z toho důvodu bylo zjištěno, že robot není schopen optimálně nést předměty za pomoci obou paží natažených před sebou.

Přenesení těžkých a rozměrných břemen lze dosáhnout pomocí dvou robotů, kteří ponesou např. rozměrnou krabici spolu. Díky možnosti opření se do krabice, čímž si navzájem roboti z hlediska stability pomáhají, je možné takto objemný předmět přenést.

Pěnový míček byl tedy pro tyto účely ideální, neboť při chůzi pohybem paže dopředu a dozadu není omezen prostor nohou a lze s ním bez problému manipulovat, aniž by došlo ke ztrátě stability. Jediné, na co je třeba myslet je to, že nelze natáhnout paži, pokud se robot ještě pohybuje a přibližuje se k cíli. Toho bylo plánováno využít při uchopení a odložení míčku. Robot by pomalu přicházel k míčku a mezi tím by si připravil paži pro uchopení předmětu. Při tomto pohybu bohužel došlo ke ztrátě rovnováhy, neboť robot za chůze neudrží stabilitu s nataženou paží dopředu. Stále bylo potřeba myslet na to, že pokud robot jde dopředu, tak zaujímá výchozí pozici s nataženými pažemi podél těla.

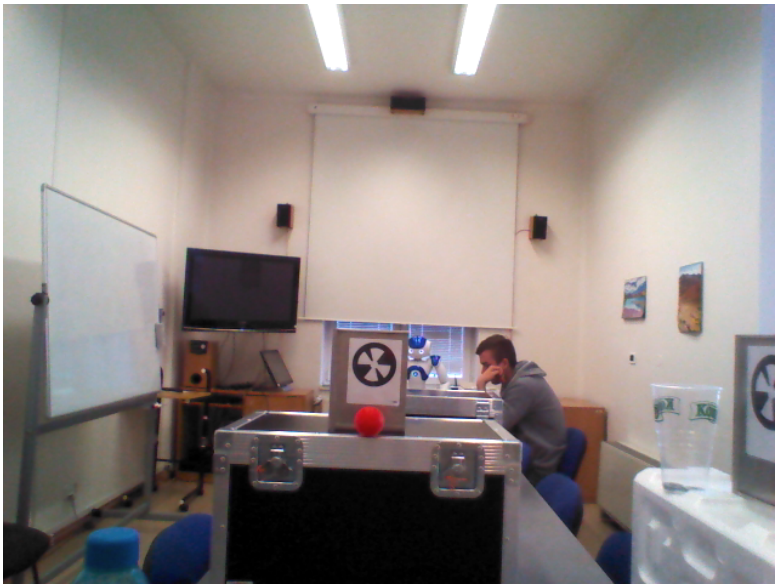
3.2.2 Výchozí podmínky pro navigaci

Základem pro navigaci k cílovému místu je vizuální kontakt s cílovým místem. V tomto případě s NAO markem, který musí být viditelný po celou dobu navi-



Obrázek 3.4: Vývojový diagram popisující navigační program. V zelených bublinách je uvedeno, co robot říká.

gace. Z toho důvodu musí mít také překážky omezenou velikost, hlavně výšku. Ty by měly být vysoké maximálně do výšky robotových ramen a minimálně do výšky jeho boků, aby byly detekovatelné za pomoci sonarů. Ideální je to tedy, pro experimentální účely, 1,5 l lahev od balené vody.



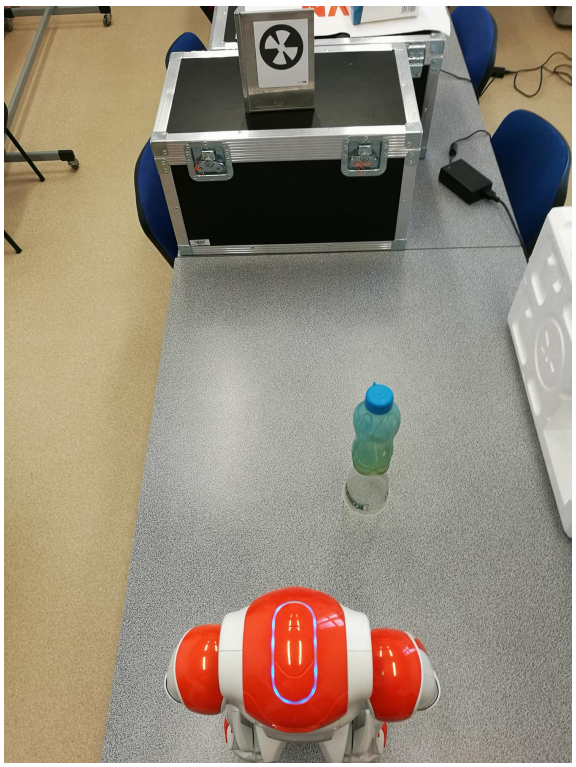
Obrázek 3.5: Pohled z hlavní kamery robota na začátku navigace

3.2.3 Detekce překážek

```
1  Lsensor = Lsonar(memory)
2  Rsensor = Rsonar(memory)
3  #chuze dopredu za predpokladu, ze pred robotem neni zadna#
4  #prekazka a neni v prime blizkosti NAO marku             #
5  if Lsensor > 0.4 and Rsensor > 0.4 and posledni_stav_L == 0
6  and posledni_stav_R == 0 and X_shape_size <= 0.16:
7      if heading_angle != 0.0:    #mark je vyosen na kamere#
8          #chuze o 10 cm vpred + rotace eliminujici chybu  #
9          motion.moveTo(0.1, 0.0, heading_angle)
10     else:   motion.moveTo(0.1, 0.0, 0.0)
11
12 def Lsonar(memory):
13     vzdalenostL = memory.getData("Device/SubDeviceList
14                                  /US/Left/Sensor/Value")
15     return vzdalenostL
16 def Rsonar(memory):
17     vzdalenostR = memory.getData("Device/SubDeviceList
18                                  /US/Right/Sensor/Value")
19     return vzdalenostR
```

K bezchybnému dosažení cíle je třeba pohybovat se po co nejkratší trase a vyhýbat se překážkám, které mohou robotovi přijít do cesty. K detekci překážek bylo využito ultrazvukového senzoru, který je vhodný pro tyto účely. Není bezchybný. Pro tento účel by byla nejideálnější tzv. laserová hlava, jenž snímá okolí ve velice širokém úhlu, čímž by robot získal prostorové vidění a vytvořil by si tak paměťovou mapu, pomocí které by se mohl orientovat v prostoru. Tato nadstavba ovšem nebyla k dispozici, a proto bylo využito pouze ultrazvukových senzorů. Překážka je detekována jedním ze dvou ultrazvukových senzorů vyhodnocujících, na jaké straně se nejspíš překážka nachází, případně na kterou stranu by bylo výhodnější provést úhybný manévr.

Pokud je robotem detekována překážka na vzdálenost menší, než 40 cm na jedné straně, následuje úhybný manévr na stranu druhou. K vyhodnocení slouží rozdíl hodnot vzdáleností z obou senzorů a podmínka vzdálenosti nižší, než 40 cm alespoň na jednom z nich. Po dokončení úhybného manévru je opět zkontrolována vzdálenost překážky, a pokud podmínka vyhnutí stále přetrvává (předmět je velký), tak robot uhýbá opakovaně, načež pootáčí hlavou v opačném směru, aby nedošlo ke ztrátě vizuálního kontaktu s NAO markem.



Obrázek 3.6: Překážka je detekována sonary a následně ji robot obejde tak, aby nedošlo ke kolizi s pravou paží

Pokud již není překážka detekována a robot má volnou cestu, následuje chůze dopředu na vzdálenost 50 cm. Pro laboratorní podmínky a kvůli omezeným možnostem pohybu nebyla zvolena větší vzdálenost, takže se robot takto vyhýbá spíše

menším překážkám, jako je lahev od balené vody. V reálných podmínkách by mohla být nastavena vzdálenost na obcházení větší. Je to tak pevně nastaveno, neboť robot nijak nezjistí hloubku překážky. Jedinou možností by bylo obejít překážku ze strany, otočit se k ní čelem o 90° a takto bokem jít podél překážky, aby byly ultrazvukové senzory schopny rozeznat konec překážky. Toto řešení by bylo příliš zdlouhavé na pohyb robota k cíli, a proto od něj bylo upuštěno. Pro reálné podmínky s dostatkem místa by mohla být pro obcházení nastavena vzdálenost 1 m, čímž by byla vyřešena většina běžných překážek.

Poté, co se robot dostane za překážku, vrátí se bokem o stejnou vzdálenost, jakou urazil před překážkou a otočí hlavu zpět do osy a pokračuje v přibližování se na danou pozici a detekování překážek. Až, když se dostane na malou vzdálenost před podstavec, na kterém se nachází míček a NAO mark, tak nedochází k detekci překážek, protože by byl detekován právě podstavec. Ultrazvukové senzory jsou zde použity už jen ke správnému natočení k podstavci.

3.2.4 Přibližovací manévr k míčku

Původní plány na přiblížení se k podstavci s nataženou paží, připravenou chytit míček, byly zamítnuty kvůli již zmíněném faktu, že robot se nemůže pohybovat dopředu s nataženou paží, neboť by ztratil rovnováhu a převrátil by se. Na základě těchto poznatků tedy bylo zvoleno postupné přibližování se k podstavci na optimální vzdálenost, aby robot dosáhl na míček, a zároveň aby podstavec nebyl v kolizi s pažemi při manipulaci.

Přibližovací manévr, při němž je započata finální korekce dosáhnutí konkrétního místa je započat cca 0,5 m před podstavcem. Tato vzdálenost je hlídána ultrazvukovými senzory a úhlovou velikostí NAO marku, což je mnohem přesnější vyhodnocení vzdálenosti. V tento okamžik také přestávají ultrazvukové senzory plnit funkci monitorování překážek, protože by došlo k vyhodnocení podstavce. Primární roli sehrají při vycentrování robota kolmo k podstavci tak, aby se pohyboval směrem k míčku ideálně pod pravým úhlem. V tomto prostoru by se tedy již neměly nacházet překážky, aby byl robot schopen se dostat na místo určení.

Jsou použity dvě možnosti vycentrování robota ke značce, přičemž každá z nich využívá jiného senzoru.

Prvním způsobem je robot centrován po celou dobu navigování. Použity jsou údaje z kamery, které jsou součástí informace o detekovaném NAO marku.

```
1  if heading_angle != 0.0:    #mark je vyosen na kamere#
2      #chuze o 10 cm vpred + rotace eliminujici chybu  #
3      motion.moveTo(0.1, 0.0, heading_angle)
4  else:   motion.moveTo(0.1, 0.0, 0.0)
```

Základem je horizontální úhel, o jaký je snímaná značka pootočena vůči ose kamery. Tento úhel pak stačí jednoduše zadat jako parametr pro natočení při

chůzi. Díky tomu je dosaženo pravidelné korekce chůze tak, aby robot šel stále rovně. Tímto způsobem sice bude docíleno chůze rovně ke značce, ale nevyřeší se tím přiblížení ke značce pod pravým úhlem. Robot se tak může pohybovat k NAO marku z boku pod jakýmkoliv úhlem, což je znázorněno na obr. 3.7 na str. 35. Důvodem může být výchozí pozice z takového místa, odkud robot sice vidí značku, ale nestojí vůči ní kolmo, nebo se při úhybném manévru vychýlil z osy. Takto se pohybuje po většinu trasy, dokud se nedostane na hranici finální korekce, kdy zahájí přibližovací manévr.

Druhým způsobem je možnost vycentrování robota kolmo k podstavci. Využívány jsou údaje o vzdálenosti z obou ultrazvukových senzorů. Jejich rozdílem je vyhodnoceno, na jaké straně od značky se robot nachází, tedy kam je třeba jej směřovat.

```
1  #vypocet natoceni pro vycentrovani#
2  if v_ose == 0:#robot neni v ose, je zahajeno centrovani#
3      dif_sensor_abs = Lsensor - Rsensor  #rozdil senzoru#
4      if dif_sensor_abs > -0.03 and dif_sensor_abs < 0.03:
5          y = 0.0
6          rot = 0.0   #robot se dostal do osy s malou     #
7          v_ose = 1   #toleranci,centrovani dale neprobiha#
8      elif dif_sensor_abs>-0.25 and dif_sensor_abs<=-0.03:
9          y = -0.05   #pohyb o 5 cm doprava#
10         rot = 0.01745 * 5   #rotace o 5 stupnu#
11     elif dif_sensor_abs>=0.03 and dif_sensor_abs<0.25:
12         y = 0.05    #pohyb o 5 cm doleva#
13         rot = -0.01745 * 5
14 else:   y = 0.0
15         rot = 0.0
```

První pokus o otočení robota přesně o úhel, který zbývá ke kolmici s podstavcem byl zamítnut z toho důvodu, že ne vždy byl údaj o vzdálenosti z obou ultrazvukových senzorů správný. Problémem bylo, že při natočení k podstavci byla vzdálenějším senzorem občas detekována zeď za podstavcem místo jeho rohu. Vše bylo velice dobře spočítáno s tím, že robot se na první pokus dostal do potřebného koridoru. Jenže občas přišel nesprávný údaj z ultrazvukových senzorů a robot se pohyboval buďto na opačnou stranu o největší možný úhel, jaký byl povolen, nebo se ze správného koridoru zase vychýlil o velký úhel a už nebyl schopen se vrátit, protože nebyl detekován NAO mark. Proto bylo od tohoto ustoupeno a byl zvolen konstantní úhel 5° a vzdálenost 5 cm, o které se robot natáčí a posouvá ke kolmici a postupně se po kouscích takto dostane do osy.

Pokud se už robot dostal do koridoru a pohyboval se v něm s malou tolerancí, tak ke korekci za pomoci sonarů nedochází, neboť je plně dostačující korekce pomocí kamery. Tato metoda je robustní vůči náhodné chybě. Pokud by nastala, tak není

fatální, jako v předchozím případě a robot je schopen se při přibližování ještě vrátit do potřebného koridoru.

Přibližování k NAO marku kombinované s centrováním robota probíhá ve třech úrovních podle vzdálenosti, která ještě zbývá. Tato vzdálenost již není kontrolována ultrazvukovými senzory, neboť je nižší než nejmenší rozlišitelná. Je tedy získána z informace o marku. Úrovně jsou rozlišeny podle velikosti NAO marku. V první fázi robot dělá posun dopředu po 5 cm, poté po 3 cm a ke konci již po 1 cm, aby bylo dosaženo přesné výchozí polohy pro uchopení míčku. V pauze mezi pohybem je vždy kontrolována velikost značky.

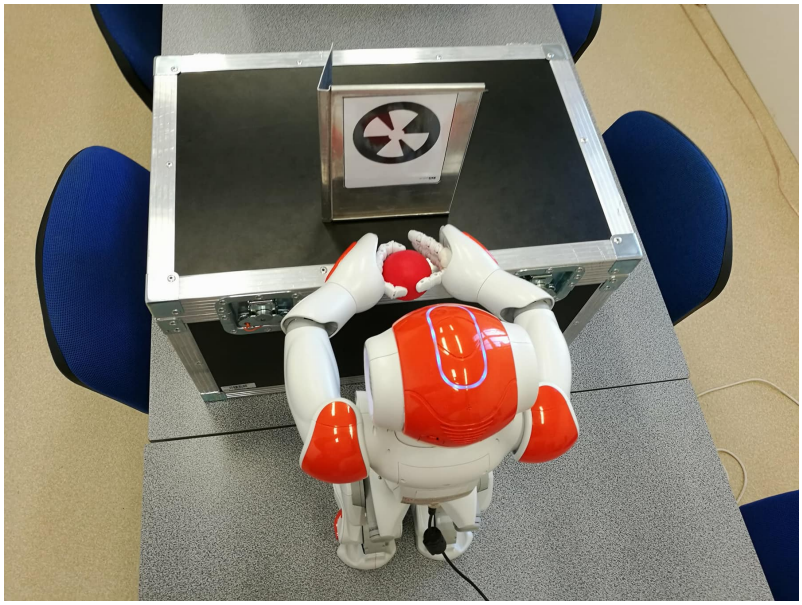
```
1  if X_shape_size < 0.25:     #velikost NAO marku v rad#
2      angel = 0.1
3      motion.setAngles("HeadPitch", angel, 0.2)#sklopeni hlavy#
4      #rotace je souctem uhlu vyhodnoceneho kamerou + sonary#
5      motion.moveTo(0.05, y, heading_angle + (rot))
6  elif X_shape_size < 0.35:
7      motion.moveTo(0.03, y, heading_angle + (rot))
8  elif X_shape_size < 0.4:
9      motion.moveTo(0.01, 0.0, heading_angle)
10 elif X_shape_size > 0.4:
11     chyceni_micku(motion)
12     angel = 0.0
13     motion.setAngles("HeadPitch", angel, 0.2)#narovnani hlavy#
14     motion.moveTo(-0.2, 0.0, -0.1)  #odstoupeni od NAO_marku#
```

Obrázek 3.7: Centrování robota do osy, využívající údaje o vzdálenosti ze sonarů

3.2.5 Manévr uchopení míčku

Stojí-li robot na daném místě, může začít sekvence pohybů paží za účelem chycení míčku. Podmínkou je, že míček je položen na definovaném podstavci a je ve vertikální ose se značkou. Původní verze měla být taková, že robot zjistí souřadnice míčku a na toto místo navede paži a uchopí jej. Problém byl v tom, že určení souřadnic míčku není přesné tak, jako u NAO marku. Zejména pro vzdálenost, která je mezi robotem a míčkem (souřadnice X). Z toho důvodu byl míček umístěn pod značku a robotovy paže jsou navedeny na dané místo.

Dále musela být vytvořena přesná sekvence pohybů paží po trajektorii dané souřadnicemi, jenž byly získány uvolňováním kloubů a nastavením paží do požadované polohy s následným vyčtením těchto souřadnic. K tomu byl použit program *NaoSimplyControlApp* vytvořený za účelem bakalářské práce panem Bc. Jíšem [5], který dělal s tímto robotem v minulosti. Touto naučenou sekvencí se předchází kolizím s podstavcem, která nastala, když byl robotovi dán příkaz natáhnout ruku pro míček. Jelikož robot stojí v malé blízkosti před podstavcem, aby dosáhl na míček, a paže se pohybuje po nejsnazší trajektorii, tak zákonitě muselo dojít ke kolizi. Sekvence tedy musela obsahovat rozpažení s postupným zvednutím paží až nad hlavu, kde dojde k otočení paží v ramenních kloubech a ohnutím předloktí je robot připraven k finálnímu uchycení. Nejprve byl pokus o uchopení míčku jednou paží,



Obrázek 3.8: Uchopení míčku levou paží. Pravá slouží pouze k přidržení.

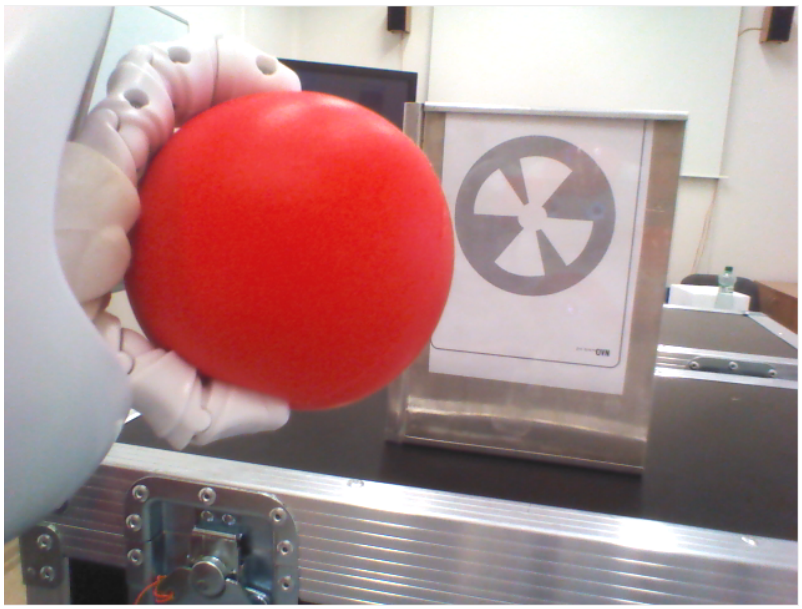
ale jistější způsob je použití paží obou. Předchází se tím náhodnému klepnutí do míčku, následkem čehož by se mohl odkutálet jinam. Robot má tedy rozevřené dlaně a předtím, než levou dlaní stiskne pěnový míček je pravou paží přitlačen, aby nedošlo k vyklouznutí a nesprávnému uchycení, což je zřejmé na obr. 3.8.

Po uchopení míčku levou rukou následuje stejná sekvence pohybů paží, jen v opačném pořadí, aby bylo dosaženo výchozí polohy paží podél těla pro chůzi.

Nakonec robot odstoupí dozadu od podstavce, aby mohl zkontrolovat, že opravdu drží míček a vyhledat místo, kam ho má odložit.

3.2.6 Kontrola uchopení míčku

Po odstoupení od podstavce dojde ke kontrole uchopení míčku. Rychlejší by bylo míček zkontrolovat hned u podstavce, ale neděje se tak z důvodu možného upuštění při zpětné manipulaci paží k tělu podél podstavce a následné chůze dozadu. Proto proběhne kontrola až poté, co robot odstoupí od podstavce. Levá paže je navedena na souřadnice, které znázorňují levý dolní okraj zorného pole vrchní kamery, aby byl míček dobře detekovatelný, což je vidět z pohledu robota na obr. 3.9. K detekování slouží metoda *redBallDetected()*, jenž vrátí pole obsahující pozici středu červeného míčku v souřadnicích X a Y a velikost ve stejných souřadnicích. Dále metoda vrací souřadnice kamery a čas, za který byl objekt detekován. Nicméně tyto údaje už nebyly pro tuto úlohu podstatné.



Obrázek 3.9: Detekce správného uchopení červeného míčku (pohled z horní kamery robota)

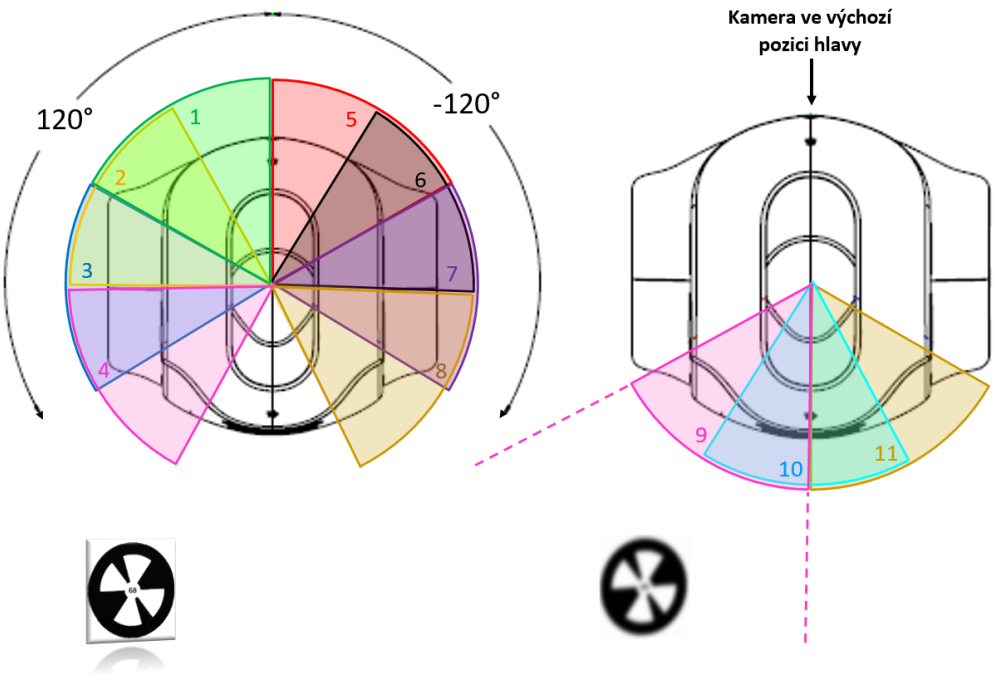
Zarážející bylo, že metoda nevracela žádnou dvoustavovou hodnotu o tom, zda byl míček detekován, či nikoliv. Metoda vracela pole s hodnotami i přes to, že míček detekován nebyl. Hodnoty o pozici středu byly ovšem náhodné a pokaždé jiné. Proto byla zjištěna přibližná poloha míčku při tréninkovém detekování paže s míčkem na pozici, která byla použita i v programu. Tato poloha pak byla odečtena od té aktuální s tím, že pokud je rozdíl menší než daná tolerance, tak je rozhodnuto o tom, že míček je opravdu uchopen. Děje-li se tak, robot ještě řekne: „*Detected!*“. Není-li splněna podmínka, ozve se z reproduktorů: „*Not detected!*“ a program je ukončen.

```
redball_proxy.subscribe("Test_RedBall", 500, 0.0)
val = memory.getData("redBallDetected")
if (val and isinstance(val, list) and len(val) >= 2):
    Ball_info_array = val[1]        #vracene pole hodnot#
    ball_X = Ball_info_array[0]     #souradnice micku v ose X#
    ball_sizeX = Ball_info_array[2] #velikost micku v ose X#
    ballX = ball_X - 0.17           #odecteni ref. hodnoty#
    sizeX = ball_sizeX - 0.4        #odecteni ref. hodnoty#
    #micek je detekovan na spravne pozici#
    #vyhodnoceni, zda je vysledek v toleranci#
    if(((ballX > -0.05)and(ballX < 0.05))
       and((sizeX > -0.03)and(sizeX < 0.03))):
        tts.say("Detected!")
        return 1
    #micek neni detekovan#
    else:   tts.say("Not detected!")
        return 0
```

3.2.7 Vyhledání místa určeného pro odložení míčku

Místo pro odložení míčku musí být taktéž označeno NAO markem ze stejných důvodů, jaké už byly zmíněny. Po úspěšném detekování uchopeného míčku je započato vyhledávání místa pro následné odložení. Před tím se ještě robot natočí čelem k NAO marku stejným způsobem, jako při přibližování se k podstavci. Je využito opět horizontálního úhlu, o který je snímaná značka pootočena vůči ose kamery. Postup je popsán výše v sekci 3.2.4 na straně 33. Tato korekce je nezbytná proto, že při chůzi dozadu se robot téměř vždy pootočí o jiný úhel a to i přes to, že je zadána souřadnice pro chůzi pouze ve směru X. Děje se tak proto, že klouby robota jsou už značně opotřebené, hlavně převody v nich, následkem čehož se každý robot chová zcela odlišně. Styl chůze se liší také na různém povrchu.

Je třeba zmonitorovat 360° úhel prostoru kolem robota, aby bylo možno značku najít téměř v kterýchkoliv místech, kde robotovi nic nepřekáží ve výhledu. Většinu snímaného prostoru je možné pokrýt otáčením hlavy. Takto se pokryje velký úhel kolem robota a není tedy nutné pokaždé pootáčet celým tělem. Konkrétně je to téměř 120° na každou stranu. Jelikož při plném otočení hlavy je pod úhlem 120° osa kamery, která má sama zorný horizontální úhel o velikosti 60°, je možné při plynulém otáčení hlavy na jednu stranu nasnímat zorný horizontální úhel o velikosti 150°. Zbyde tak 60° za zády robota, kvůli kterým je potřeba otočit robota „čelem vzad“. Samozřejmě je třeba počítat s tím, aby se zorné pole z předchozího záznamu překrývalo s monitorovaným prostorem po otočení robota o 180°. Neznamená to tedy, že po otočení robot na první pohled pokryje zbývajících 60° zorného úhlu. NAO mark by mohl být na hraně snímku a byla by vidět např. jen půlka kruhu, což není dostatek pro správnou detekci. Hlavu je nejideálnější otáčet po 30°, díky



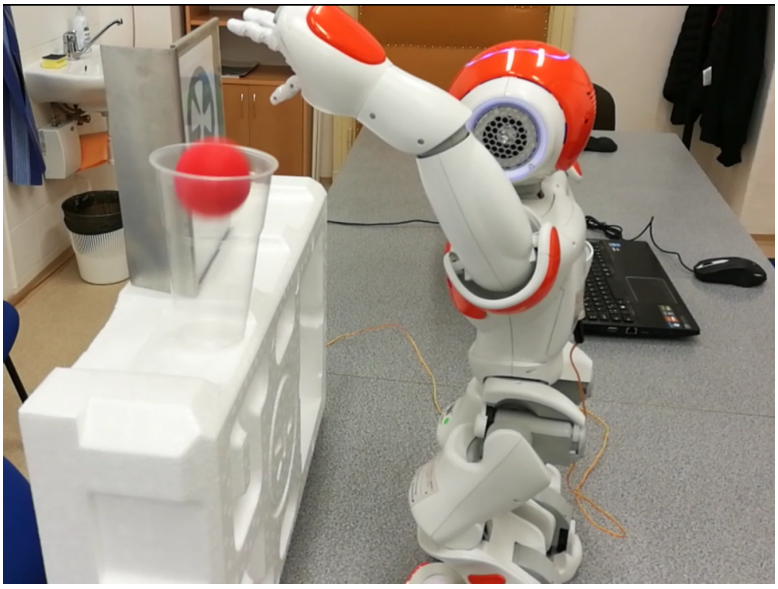
Obrázek 3.10: Vyhledávání NAO marku otáčením hlavy po 30°. V tomto případě je značka detekována v 9. sektoru po otočení robota o 180°.

čemuž je vždy každá polovina zorného pole nasnímána dvakrát. Robot tedy po krocích pootáčí hlavou o 30° a detekuje zornou oblast, dokud není nalezena správná značka, což je vidět na obr. 3.10.

Pro lepší detekci bylo upuštěno od plynulého otáčení hlavy z toho důvodu, že snímací schopnosti kamery pro detekci NAO marku nejsou ideální za pohybu. Po úspěšné detekci hledané značky se robot otočí celým tělem o úhel natočení hlavy vůči výchozí pozici. Tělo se tedy natočí za hlavou, aby byly v jedné ose.

3.2.8 Vhození míčku do kelímku

Poté již následuje stejný proces přibližování se k NAO marku, jako v případě vyhledání míčku. Zde je vedle něj umístěn kelímek, do kterého NAO nalezený míček vhodí. Po přiblížení se ke značce je vše stejné, jako v předchozím případě v sekci 3.2.5 na str. 36 až na to, že k požadovanému úkonu stačí jen jedna paže. Ta je naučeným pohybem navedena nad kelímek, opět z důvodu možné kolize s podstavcem, a robot uvolní dlaň, načež míček vhodí do připraveného kelímku, což je vidět na obr. 3.11. Po odložení míčku vrátí svou paži naučeným pohybem v opačném pořadí do výchozí pozice k tělu a odstoupí od podstavce směrem dozadu.



Obrázek 3.11: Míček je vhozen do kelímku

3.2.9 Detekce míčku v kelímku

Jelikož je kelímek průhledný, je možné stejným způsobem, jako při uchopení míčku na str. 37, zkontrolovat zda-li se míček opravdu nachází v kelímku, což je vidět na obr. 3.12.

Obrázek 3.12: Detekce červeného míčku v kelímku (pohled z horní kamery robota)

NAO u podstavce sklopí hlavu a lehce ji natočí doleva, aby měl míček uprostřed zorného pole horní kamery a za pomoci metody *redBallDetected()* zjistí stejným výpočtem, jako v případě kontroly uchopení míčku na str. 37 , správnou pozici míčku. Poté už jen odstoupí od podstavce směrem dozadu a program, jenž je ukončen, může

být zadním tlačítkem na hlavě spuštěn znovu za stejných podmínek, jako jsou na startu.

3.3 Modifikace navigačního algoritmu

Program byl vyvinut za účelem použitelnosti, jako univerzální algoritmus sloužící pro tvorbu nových úloh. Vytvořený a odladěný algoritmus je tedy možné implementovat pro různé úlohy typu „pick and place“ a vytvořit tak novou úlohu zabývající se podobným typem navigace. Jelikož je v programu využito velkého množství funkcí a schopností robota, tak je tedy možné vyjmout části kódu s příslušnými funkcemi a postavit na jejich základu jinou úlohu.

Pro demonstraci univerzálnosti programu vznikly další dvě úlohy vycházející z výše popsaného programu.

3.3.1 Uvítání robotem a potřesení rukou

Prvním příkladem může být úloha vycházející z původního algoritmu sloužícího pro vyhledání míčku a manipulaci s ním. Tento příklad byl vytvořen za účelem demonstrace rychlosti, za kterou je možné kód modifikovat na lehčí úlohu, neboť práce na úpravě kódu a následném odladění zabrala pouze jednu hodinu.

Vznikla tak úloha, kdy robot najde člověka za pomoci otáčení hlavy (případně celého těla) a detekce obličeje, otočí se směrem k němu a natáhne k němu ruku, aby ho pozdravil. Poté, co mu je podána ruka, čeká se na stisk kapacitního tlačítka na zápěstí, aby se robot představil.

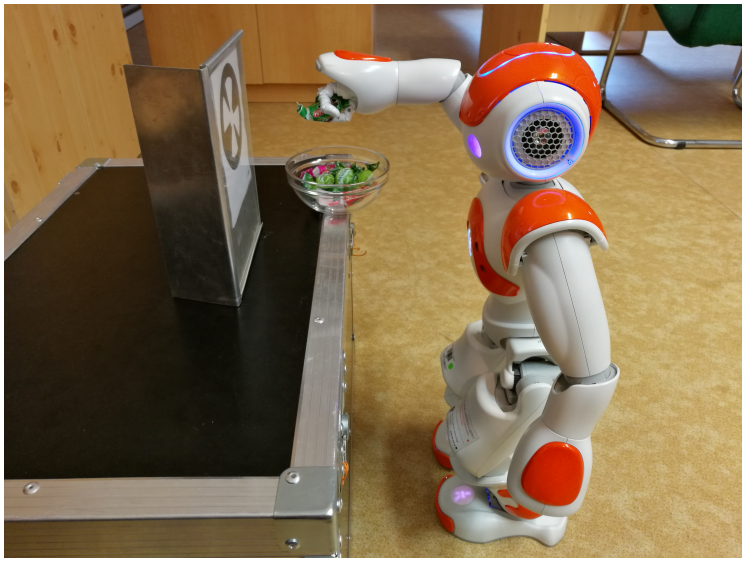
Tato metoda bohužel nemohla být použita, protože bylo zjištěno, že starší typ robota ve verzi H21, s nímž bylo pracováno, ještě nemá na zápěstí kapacitní dotykové senzory, jako novější typ H25, na kterém to bez problémů fungovalo. Nicméně problém byl vyřešen díky možnosti měřit proud tekoucí do jednotlivých motorů v kloubech. To bylo využito místo čekání na stisk kapacitního tlačítka. Bylo změřeno, že do tohoto kloubu teče proud o hodnotě téměř 0,5 A z důvodu kompenzace gravitační síly při předpažení. Stačilo tak čekat, kdy proud klesne pod určitou hodnotu blížící se nule (v tomto případě pod 0,1 A), aby byl vyhodnocen dotek od člověka.

V praxi to znamená, že robot natáhne paži před sebe a člověk mu ji pozvedne, čímž klesne proud tekoucí do ramenního kloubu k nule a robot stiskne dlaň, představí se a potřese člověku rukou.

3.3.2 Robot podává bonbóny do ruky

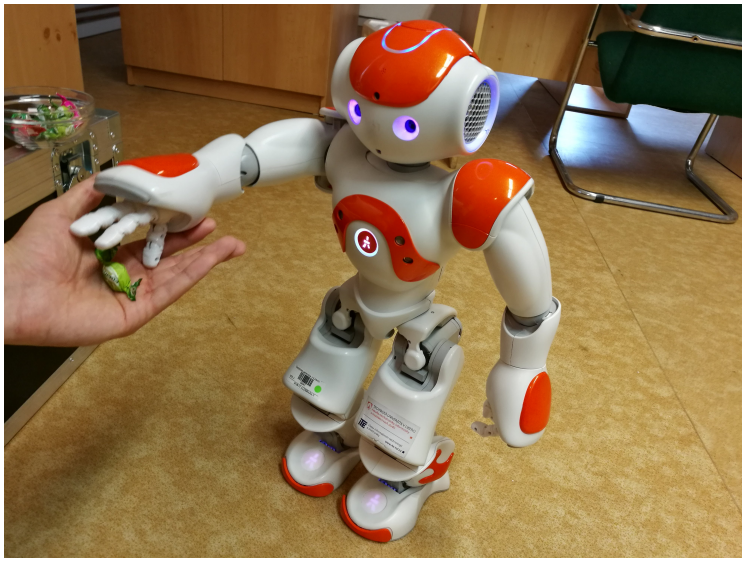
Dalším příkladem může být modifikace úlohy s míčkem a předchozí úlohy 3.3.1, kdy robot vyhledá NAO mark, pod kterým bude místo míčku položena miska s bonbóny, ze které robot nabere do dlaně několik bonbónů. Poté následuje stejná sekvence otáčení hlavy a vyhledávání, jako v případě vyhledávání kelímku v sekci 3.2.7 na

straně 38 s tím rozdílem, že robot vyhledá člověka za pomoci detekce obličeje, otočí se a natáhne paži směrem k němu. Poté, co mu bude paže pozvednuta, aby byl detekován dotek poklesem proudu tekoucího do ramenního kloubu, jako v předchozí úloze, otevře dlaň a podá bonbóny.



Obrázek 3.13: Robot nabírá bonbóny z misky

Tvorba této úlohy trvala téměř stejně dlouho, jako té předchozí. Bylo sice nutné předělat sekvenci pohybů sloužících původně pro uchycení míčku, ale zase už bylo vyřešeno detekování kontaktu s rukou člověka, čímž se tato úloha výrazně zjednodušila. Výhodou zde je, že bonbóny jsou umístěny v misce a nehrozí jejich shození, jako v případě míčku. Stačí tedy do misky sahat pouze jednou paží.

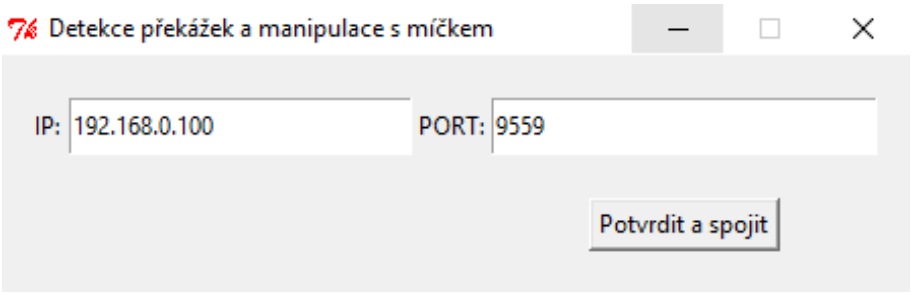


Obrázek 3.14: Robot podává člověku bonbóny do dlaně

3.4 Grafické rozhraní pro spojení s robotem

Pro možnosti spojení s robotem bylo vytvořeno grafické rozhraní za pomoci knihovny Tkinter, která je pro tyto účely ideální, neboť není nutné instalovat navíc další software, protože knihovna je součástí jazyku Python. Ten bohužel nenabízí možnost rozložení grafických prvků, proto tedy přidávání prvků probíhá úpravou zdrojového kódu. [4]

Pro všechny tři úlohy byla vytvořena spustitelná aplikace, která vznikla konvertováním kódu ve formátu .py do formátu .exe, neboli z jazyku Python do Windows aplikace. Díky tomu je možné aplikaci spustit na kterémkoliv počítači. Po spuštění aplikace se otevře okno viditelné na obr. 3.15, v němž jsou dvě textová pole. Do prvního se zadává IP adresa robota a do druhé port robota, který je stále stejný. Hodnoty jsou již předvyplněny první IP adresou, kterou získá robot, jenž se jako první připojí k síti. [4]



Obrázek 3.15: Grafické rozhraní pro spouštění aplikace určené pro vyhledávání červeného míčku

Aplikace, po kliknutí na tlačítko *Potvrdit a spojit*, zkontroluje délku IP adresy a také to, že jsou zadána pouze čísla. V případě chybně zadaných hodnot se zobrazí chyba ve stavovém řádku aplikace. Nestane-li se tak, dojde k pokusu o navázání komunikace s robotem. Po úspěšném spojení je spuštěn konkrétní program, pro který je okno vytvořeno. [4]

4 Zhodnocení výsledků a srovnání se stávajícími projekty

V této kapitole se nachází schrnutí a zhodnocení této práce. Dále jsou zde vyzdvihnuty nedostatky a problémy, se kterými bylo nutno se potýkat.

4.1 Dosažené výsledky

V průběhu práce bylo zjištěno několik nedostatků, s nimiž bylo nutné se vypořádat, jelikož některé z nich byly limitující. Bylo tedy nutné vymyslet opatření, jenž by chybám předcházelo.

Jedním z největších problémů byla chůze rovně. Jak už bylo zmíněno v kapitole 1.2 na straně 16, tak se i zde ověřilo, že každý robot se chová zcela jinak, díky stavu opotřebení kloubů. Různě staří roboti se i jinak rychle pohybují a nekontrolovatelně vybočují ze směru při chůzi vpřed, nebo vzad. To bylo korigováno při chůzi vpřed centrováním za pomoci snímání NAO marku. Robot tedy ujde 10 cm, poté se zastaví a při pauze sloužící ke čtení NAO marku vyhodnotí i úhel, o který se má natočit zpět do osy a tento úhel je připočítán do chůze vpřed. Větší problém nastal při chůzi vzad, kdy byl robot schopen se na krátké vzdálenosti natočit o značný úhel. Někdy bylo natočení tak výrazné, že nebylo možné detekování značky. Naštěstí se tak dělo pouze na jednu stranu, a tak byla chůze vzad korigována přidáním konstantní hodnoty, o který se má robot natočit, aby chybu vykompenzoval. I přesto malá chyba zůstala, nicméně bylo možné použít stejnou korekci, jako při chůzi vpřed. Odlišnosti v chůzi, hlavně při zatáčení, byly patrné i na různém povrchu, neboť chodidla pokaždé jinak prokluzovala. Tento problém byl částečně eliminován přilepením měkké pásky, která tlumí vibrace při chůzi a zmenšuje prokluz. Robot také tolik „nedupe“.

Dalším problémem byla čitelnost NAO marků za zhoršených světelných podmínek a za chůze. To je popsáno už na stránkách výrobce [1] a nelze to nijak odstranit. Jedinou možností je zajistit dostatečné světelné podmínky s co možná nejméně osvětlenými plochami v zorném poli kamery. Těmito plochami jsou míněny například roztažená okna za jasného dne. Pokud stojí NAO mark přímo proti nezataženému oknu, je možné postavit za toto místo nějakou desku, která část přímého světla zastíní. Značky je také obtížné detekovat za chůze. Pohybuje-li

se robot, obraz z kamery je velmi špatný a rozklepaný. Jelikož obraz není nijak stabilizován, je detekce za chůze velmi obtížná. Proto bylo od detekce za chůze upuštěno a robot se v základním režimu přibližuje k podstavci vždy po 10 cm. Při zastavení vždy detekuje NAO mark a zjistí vzdálenost od překážky.

Také u ultrazvukových snímačů byl problém se značným úhlem, pod kterým jsou překážky detekovány. Velice dobře to bylo vyřešeno v projektu, jehož postup je popsán výše na straně 14 v části 1.1 *Stávající stav mapování okolí pomocí robotů NAO*. Nicméně popsaný algoritmus je velice složitý s několikaletým vývojem, jehož kód není nikde k dispozici a samotné napodobení nebylo v kompetenci této práce. Proto od této sofistikované detekce, pomocí laciných a nepřesných ultrazvukových senzorů rozšířených o virtuální senzory, bylo upuštěno. Použity byly pouze hrubé údaje z ultrazvukových senzorů a tomu také bylo uzpůsobeno vyhýbání, které je nastavené mnohem hruběji, než v případě práce popsané na straně 14 v oddílu 1.1, aby bylo dosaženo sice podobné úspěšnosti, ale horší trajektorie pohybu, jenž je v tomto případě pravoúhlá.

I přes nedostatky, se kterými nebylo počítáno, se podařilo vytvořit uspokojivou navigaci k cíli kombinující manipulaci s předmětem. Tato navigace není sice tak rychlá, jako v případě projektu uvedeného v oddílu 1.2, kde bylo dbáno na co nejrychlejší navigaci v přímce, ani jako v oddílu 1.1, kde byl použit propracovaný algoritmus, díky kterému se robot elegantně vyhýbal překážkám a působil více autonomně, avšak bylo dosaženo velmi dobrých výsledků, které se zdají být pro náplň této práce velice uspokojivé.

Díky těmto výsledkům mohly vzniknout další dvě konkrétní úlohy, ve kterých byly uplatněny poznatky, jenž byly získány při tvorbě obecné úlohy. Byla tak dokázána snadná práce s kódem obecné úlohy, na jehož bázi bylo možné v krátkém čase realizovat nové konkrétní úlohy typu „pick and place“, což je velkým přínosem pro nového uživatele. Právě možnosti úpravy kódu na nový se nevěnoval žádný z rešeršních článků, ani nic podobného nebylo možné dohledat v literatuře, či na webu. Vznikl tak nový typ práce nabízející tuto možnost.

5 Závěr

Díky této práci vznikl univerzální kód, použitelný pro tvorbu podobných úloh, jako je tato. S ohledem na stáří robota bylo nutné se v průběhu práce vypořádat s řadou problémů. Například s horším pohybem robota z důvodu opotřebení převodů v kloubech, čímž výrazně klesá opakovatelnost při chůzi a manipulaci s předměty. Dále u staršího robota chybí tlačítka na zápěstích, což bylo vyřešeno pozvednutím paže a měřením proudu tekoucího do motoru ramenního kloubu namísto čekání na stisk tlačítka. Také kamera má na dnešní poměry spíše podprůměrné parametry, takže byl problém se snímáním okolí za zhoršených světelných podmínek a při pohybu.

S ohledem na množství nedostatků, se kterými bylo nutno se potýkat, jsou výsledky práce zcela uspokojivé. Vznikla tak funkční navigační úloha typu „pick and place“, která je použitelná, jako univerzální algoritmus pro tvorbu nových úloh postavených na stejném principu. To bylo také dokázáno tím, že velice rychle vznikly, vyjmutím částí kódu, další dvě konkrétní úlohy na stejném principu. V první úloze robot NAO vyhledá člověka, pozdraví jej a potřese si s ním rukou. Druhá úloha vychází ve velké míře z obecné úlohy, kdy robot vyhledá misku s bonbóny, nabere je a podá je člověku do dlaně. Díky univerzálnosti tohoto algoritmu by neměl být pro nezkušeného uživatele problém si za jeden den vytvořit, vyjmutím částí kódu, svůj vlastní program na podobném principu.

Měla by se tak usnadnit práce začátečníkům s humanoidními roboty NAO ve vyšším programovacím jazyce s využitím všech funkcí robota, které základní program Choreographe nenabízí. Tato práce se ukázala býti velice přínosnou pro práci s humanoidními roboty a možností programovat je v jazyce Python.

Literatura

[1] NAOqi APIs — Aldebaran 2.1.4.13 documentation. SoftBank Robotics Documentation [online],[cit. 20.3.2018]. Dostupné z: http://doc.aldebaran.com/2-1/naoqi/index.htm

[2] G. Brooks, P. Krishnamurthy and F. Khorrami, „*Humanoid robot navigation and obstacle avoidance in unknown environments*," 2013 9th Asian Control Conference (ASCC), Istanbul, 2013, pp. 1-6. doi: 10.1109/ASCC.2013.6606392 [online],[cit. 20.3.2018]. Dostupné z: http://ieeexplore.ieee.org/document/6606392/

[3] L. George and A. Mazel, „*Humanoid robot indoor navigation based on 2D bar codes: application to the NAO robot*," 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA, 2013, pp. 329-335. doi: 10.1109/HUMANOIDS.2013.7029995 [online],[cit. 20.3.2018]. Dostupné z: http://ieeexplore.ieee.org/document/7029995/

[4] VACEK, Jan. „*Možnosti rozpoznávacích algoritmů v robotu NAO*." Liberec, 2017. Bakalářská práce (Bp.). Technická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií, Ústav Informačních technologií a elektroniky (ITE), 06.2017

[5] JÍŠE, Václav. „*Interaktivní ovládání humanoidního robota NAO pomocí SDK*." Liberec, 2017. Bakalářská práce (Bp.). Technická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií, Ústav Informačních technologií a elektroniky (ITE), 06.2017

[6] Robot NAO a výzkum s pokročilou hračkou | VTM.cz. [online],[cit. 20.3.2018]. Dostupné z: http://vtm.e15.cz/robot-nao-a-vyzkum-s-pokrocilou-hrackou

[7] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN - technická literatura, 2005. 256 str., ISBN 80-7300-141-1.

[8] KISUNG, SEO. Using NAO: Introduction to interactive humanoid robots. Francie: Icones, 2013. 276 str.

A Obsah přiloženého CD

- Bakalářská práce - ve formátu .pdf
- Zdrojové kódy všech tří programů - ve formátu .py
- Spustitelné verze programů - ve formátu .exe
- Fotografie a vývojový diagram - ve formátu .png