TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Spolupráce robotů NAO

## Bakalářská práce

*Studijní program:* B2612 – Elektrotechnika a informatika
*Studijní obor:* 2612R011 – Elektronické, informační a řídící systémy

*Autor práce:* **Pavel Vaner**
*Vedoucí práce:* Ing. Miroslav Holada Ph.D.

Liberec 2018

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

# NAO Robot Collaboration

## Bachelor thesis

*Study programme:* B2612 – Electrotechnology and informatics
*Study branch:* 2612R011 – Electronic, Information and control systems

*Author:* **Pavel Vaner**
*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: | **Pavel Vaner** |
| Osobní číslo: | **M15000123** |
| Studijní program: | **B2612 Elektrotechnika a informatika** |
| Studijní obor: | **Elektronické informační a řídicí systémy** |
| Název tématu: | **Spolupráce robotů NAO** |
| Zadávající katedra: | **Ústav informačních technologií a elektroniky** |

Zásady pro vypracování:

1. Seznamte se s roboty NAO na pracovišti školitele. Zaměřte se na možnosti komunikačního sybsystému robotů.
2. Seznamte se s problematikou vzájemné spolupráce více robotů. V rešeršní části uvěďte příklady již řešených projektů ve světě.
3. Navrhněte a realizujte software, který umožní vzájemnou kooperaci mezi dvěma roboty NAO na takové úrovni, aby dokázaly společně řešit zadaný úkol. Jedná se především o předávání stavových informací a dalších nezbytných parametrů.
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. Boston Dynamics [online]. Massachusetts [cit. 2017-05-08]. Dostupné z: http://www.bostondynamics.com**

[2] **ZÁDA, Václav. Robotika: matematické aspekty analýzy a řízení. Vyd. 1. Liberec: Technická universita v Liberci, 2012. ISBN 978-80-7372-882-3.**


| | |
|---|---|
| Vedoucí bakalářské práce: | **Ing. Miroslav Holada, Ph.D.**<br>Ústav informačních technologií a elektroniky |
| Konzultant bakalářské práce: | **Ing. Leoš Petržílka**<br>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 vynaložil 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: 10. 5. 2018

Podpis: Vaner

# Poděkování

Rád bych zde poděkoval svému vedoucímu práce Ing. Miroslavu Holadovi, Ph.D. za jeho věcné poznámky, konzultace a vedení práce.

# Abstrakt

Tématem této bakalářské práce je spolupráce dvou humanoidních robotů NAO. Tento cíl je splněn díky návrhu softwarového řešení, které umožňuje kooperaci robotů. Spolupráci robotů zajišťuje jeden řídící počítač, na němž běží vytvořený software, který komunikuje s roboty a ovládá je. Součástí řešení je také spustitelná počítačová aplikace, v níž je možné připojit se k robotům a spustit jednu ze dvou konkrétních ukázek kooperace robotů. Aplikace je vytvořena pomocí vývojových nástrojů SDK od výrobce robota. Pro vývoj byl zvolen z několika nabízených programovacích jazyků Python, pro jeho velkou podporu výrobcem robota a rozsáhlou dokumentaci. Výsledný kód lze snadno přeprogramovat pro konkrétní požadovanou spolupráci robotů.



## Klíčová slova:

humanoidní robot, NAO, spolupráce robotů, kooperace robotů



# Abstract

The theme of this bachelor thesis is the cooperation of two humanoid robots NAO. This goal is accomplished through the design of a software solution that allows for robot co-operation. Robot collaboration is ensured by a single control computer running the software that communicates with the robots and controls them. The solution also includes an executable computing application in which it is possible to connect the robots and run one of two specific demonstrations of robot co-operation. The application is created using the robot producer‘s SDK development tools. Python was selected from several programming languages for development, because of his great support by the robot’s producer and extensive documentation. The resulting code can be easily reprogrammed for the specific robot collaboration required.



## Key words:

humanoid robot, NAO, robot collaboration, robot co-operation

# Obsah

# Seznam obrázků

# Seznam zkratek

| | |
|---|---|
| **API** | Application Programming Interface, rozhraní pro programování aplikací |
| **FIFO** | First In First Out, první dovnitř první ven |
| **GUI** | Graphical User Interface, grafické uživatelské rozhraní |
| **IEEE** | Institute of Electrical and Electronics Engineers, Institut pro elektrotechnické a elektronické inženýrství |
| **IP** | Internet Protocol, identifikační síťové rozhraní |
| **LED** | Light Emitting Diode, světlo vyzařující dioda |
| **PIL** | Python Imaging Library, zobrazovací knihovny pro Python |
| **RGB** | Red-Green-Blue, červená-zelená-modrá |
| **SDK** | Software Development Kit, vývojové nástroje |
| **SLAM** | Simultaneous Localization And Mapping, současná lokalizace a mapování |
| **Sonar** | SOund Navigation And Ranging, zvuková navigace a zaměřování |
| **TTS** | Text To Speech, převod textu na řeč |
| **TUL** | Technická univerzita v Liberci |
| **USB** | Universal Serial Bus, univerzální sériová sběrnice |
| **WIFI** | Wireless Fidelity, bezdrátová věrnost |

# Úvod

Robotika je jednou z nejkomplexnějších věd v oboru mechatroniky, spojuje elektrotechniku, mechaniku a programování. V industriální sféře jsou roboti již mnoho let. V současné době se ve větší míře zavádí i do venkovního prostředí. Například jako autonomní vozidla, bezpilotní drony nebo roboti určení pro průzkum prostředí.

Jednou z hlavních motivací pro vývoj robotů, je ulehčit člověku především s přenášením těžkých břemen. Ovšem i možnosti robotů jsou omezené a proto je potřeba pro složitější úkoly zapojit více robotů. Spolupráce mezi více robotickými rameny již není problémem. Na velmi dobré úrovni jsou i kolaborativní roboti, kteří spolupracují s člověkem. V oblasti humanoidních robotů je zapotřebí kooperaci ještě zdokonalit. Humanoidní roboti připomínají svou stavbou těla a vzhledem člověka a jejich pohyb je mnohem složitější než pohyb robotických ramen. Jednou z jejich obrovských výhod je, že se mohou s nákladem pohybovat v prostoru podobně jako člověk a předmět uchopit různými způsoby.

Cílem této bakalářské práce je seznámit se s humanoidními roboty NAO a možnostmi jejich vzájemné komunikace. Ze získaných poznatků bych měl navrhnout softwarové řešení pro spolupráci dvou robotů NAO, které by mělo jít snadno přeprogramovat pro různé kooperace. Pomocí navrhnutého řešení pak vyvinout jednoduchou ukázku spolupráce robotů a otestovat ji v reálném prostředí.

Výstupem by měla být aplikace spustitelná na počítači, pomocí které se k robotům bude možné připojit a spustit ukázky kooperace robotů. Pro vývoj softwaru jsem zvolil programovací jazyk Python v kombinaci s vývojovými nástroji SDK od výrobce. Grafické prostředí je vytvořeno také v jazyce Python pomocí knihovny Tkinter. Jazyk Python byl zvolen kvůli široké podpoře výrobcem robotů, uživatelské základně a rozsáhlé dokumentaci. Aplikace je spuštěna na PC, s roboty se spojí pomocí WiFi sítě. Všechna komunikace mezi roboty jde přes řídící počítač, který ví, v jakých fázích se každý robot nachází a jako master určuje jejich další úkony. Roboti jsou v podstatě používáni počítačem jako vstupně-výstupní zařízení.

# 1 Humanoidní robot NAO

Humanoidní robot NAO je produktem francouzské společnosti Aldebaran Robotics, která byla v roce 2012 odkoupena japonskou firmou SoftBank Robotics. Společnost nabízí několik humanoidních robotů, přičemž NAO je jejím prvním robotem. Mezi další pokročilejší roboty v nabídce firmy patří Pepper a Romeo. První generace robota NAO byla představena na trh v roce 2006. Od té doby prošel robot několika změnami až do nejnovější páté generace. Robot se těší celkem velké uživatelské základně a to díky faktu, že se ho po současnost celosvětově prodalo přes 10 000 kusů. Hlavním účelem robota je seznámit studenty a širokou veřejnost s robotikou a jejími základními problémy. Robot se skvěle hodí k prezentacím. Z tohoto důvodu je robot vybaven nejrůznějšími senzory, tlačítky a podporuje několik způsobů interakce s člověkem. Pro vývoj mé práce byli k dispozici dva roboti NAO. Oranžový, verze V3, což je pilotní verze z roku 2006 a modrý, v nejaktuálnější verzi V5. [1] [13]

Obrázek 1.1: Humanoidní robot NAO

## 1.1 Technický přehled

Robot NAO má na výšku 57,3 centimetrů a široký je 27,5 centimetrů. Hloubka včetně natažených rukou robota činí 31,1 centimetrů. Robot je z největší části vyroben z plastů, konkrétně se jedná o plasty s označením ABS-PC, PA-66 a XCF-30. Díky použitým materiálům je robot relativně lehký, váží 5,4 kilogramu.[1] [13]

Mozek robota, jenž se stará o všechny výpočetní operace, je integrovaný procesor od společnosti Intel. Konkrétně Intel Atom Z530 taktovaný na pracovní frekvenci 1,6 GHz. Procesor má k dispozici 1 GB operační paměti RAM. Robot má dvě možnosti kam ukládat uživatelem vytvořené programy, fotografie pořízené jednou z kamer robota, či jiná data. První možnost je ukládat data do 2 GB FLASH paměti. Ta je ovšem primárně určena pro operační systém robota. Druhé úložiště je integrovaná Micro SDHC karta s kapacitou 8 GB. Napájení robota elektrickou energií zajišťuje li-ion baterie o kapacitě 2250 mAh. Baterie se pomocí nabíječky, jež je dodávána s robotem, plně nabije zhruba za 3 hodiny. Robot je na jedno nabití použitelný od 60 do 90 minut. Doba se odvíjí od náročnosti pohybů. Roboti, které jsem měl k dispozici, vydrželi pouze 30 minut kvůli opotřebení baterie.[1] [13]

Na hlavě robota je umístěn standardní port RJ-45. Slouží především pro nastavení robota při prvním připojení, pro aktualizace softwaru nebo pro rychlý přenos souborů mezi robotem a počítačem. Maximální rychlost přenosu pomocí Ethernetu je 1 Gb/s. Hlavním komunikačním kanálem robota je bezdrátová síť WiFi. Nejnovější verze robota V5 podporuje standard IEEE 802.11 a/b/g/n. Starší verze podporují pouze protokoly IEEE 802.11 b/g/n. Vedle ethernetového portu se nachází USB port. Pomocí něj lze k robotovi připojit různé periferie jako například USB disky, Microsoft Kinect, 3D senzor od firmy Asus určený pro mapování okolí a práci s 3D informacemi. Přes USB je také možné připojit Arduino a rozšířit tak senzorické schopnosti robota. Oba porty jsou umístěny na zadní straně hlavy robota. Přístupné jsou pod odnímatelnou krytkou.[1] [13]

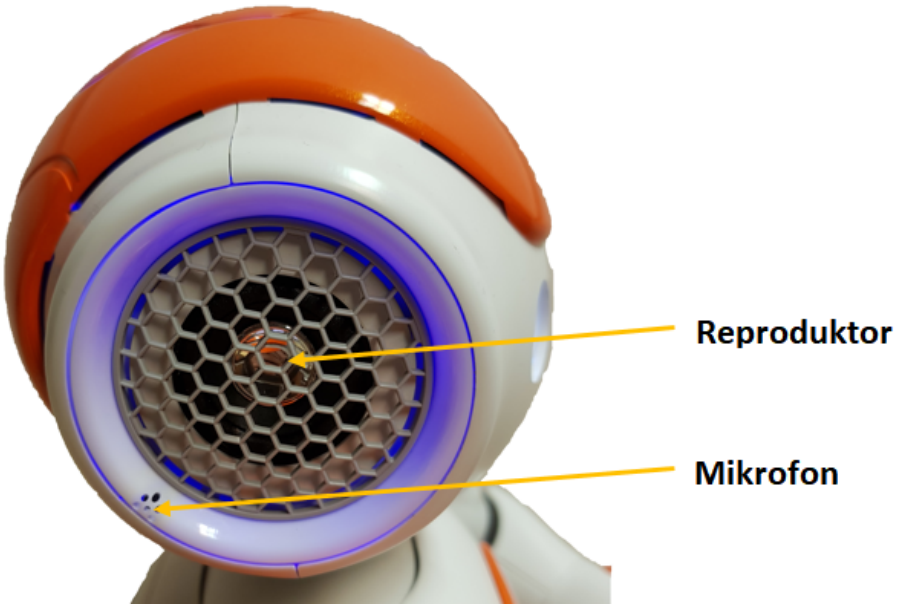
## 1.2 Hardwarové vybavení

### 1.2.1 Mikrofony

Na hlavě robota se nachází celkem 4 mikrofony sloužící k záznamu zvuku. Robota je možné ovládat pomocí hlasových záznamů nebo mohou mikrofony zaznamenat zvuk, který poté robot přehraje. Mikrofony se dají využít pro sledování zvuku. Tímto způsobem se může robot orientovat v prostoru. Přesnost sledování zvuku je pevně provázaná s okolním prostředím. Pokud dochází k odrazům zvukového signálu od okolních objektů, bude výsledná orientace v prostoru zkreslená.[1] [13]

### 1.2.2 Reproduktory

K přehrávání zvuků lze využít dva reproduktory umístěné na stranách hlavy robota NAO. Pomocí dostupného TTS systému dokáže robot syntetizovat řeč a komunikovat tak s okolním prostředím. Robot dále přehrává systémová a chybová hlášení, lze přehrát i audio soubory ve formátu MP3, WAV nebo OGG.[1]




Obrázek 1.2: Reproduktor a mikrofon

### 1.2.3 Kamery

Pro snímání okolí má robot NAO k dispozici dvě kamery umístěné v hlavě. Kamery mohou pořizovat fotografie a videa, díky čemuž se může následně robot orientovat v prostoru nebo rozeznávat a sledovat různé objekty. Vrchní kamera je umístěna na čele robota a snímá obraz přímo před robotem. Spodní kamera nahrazuje ústa robota a je natočena tak, aby mohla zaznamenávat obraz u nohou robota. Díky této kameře je robot schopen rozpoznat objekty umístěné v jeho bezprostřední blízkosti.[1] [13]

Jak horní tak spodní kamera mají stejné parametry. Maximální rozlišení obrazu je 1280x960 pixelů. Maximální frekvence snímkování je 30 snímků za vteřinu. Zorné pole kamer je bezmála 61 stupňů horizontálně a téměř 48 stupňů vertikálně.[1] [13]

Robot pomocí kamer zvládá základní orientaci v prostoru. Pro správnou funkčnost rozpoznávacích algoritmů je potřeba umístit snímané objekty do blízké vzdálenosti od kamer robota. Kvalita záznamu obrazu silně závisí na okolních podmínkách. Pokud je robot v přesvětleném prostředí, či naopak v prostředí s nízkým osvětlením, kvalita obrazu se velmi zhorší. Při těchto podmínkách obraz obsahuje šum a rozpoznávání objektů nefunguje spolehlivě.[1] [13]

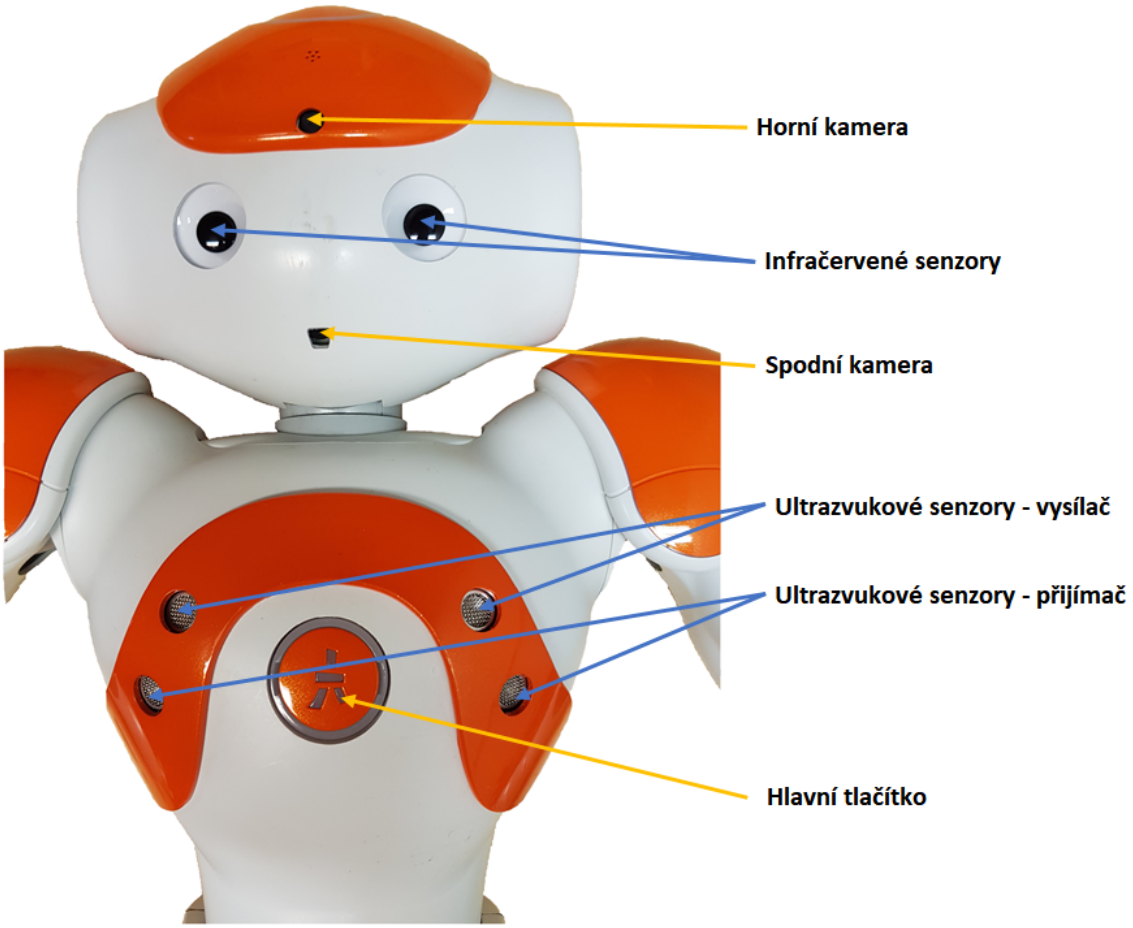
### 1.2.4 Infračervené diody

V každém oku robota NAO je umístěn infračervený senzor. K využití infračervených senzorů je zapotřebí implementovat do programu modul ALInfrared. Ten nabízí tři různé způsoby využití senzorů. První je využít robota jako dálkové ovládání. Druhá možnost je udělat z robota přijímač příkazů z dálkového ovládání. Poslední možností je kombinace dvou předešlých, a to využít modul k vzájemné komunikaci mezi roboty. Tuto možnost ovšem výrobce nedoporučuje.[1]

Modul ALInfrared používá k příjmu a odesílání ovládacích kódů balíček Linux Infrared Remote Control.[1]

### 1.2.5 Ultrazvukové senzory

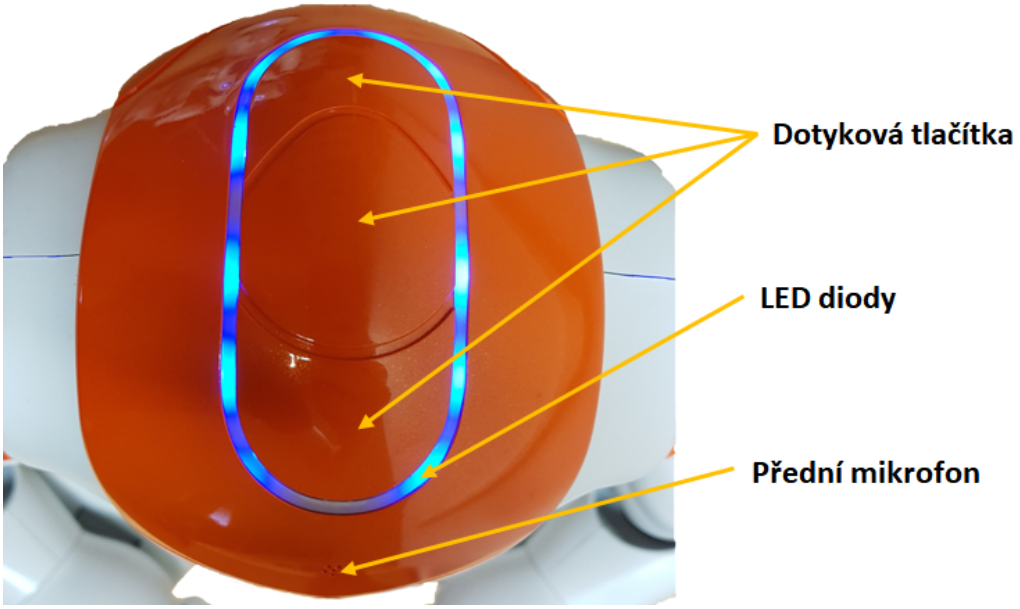
Robot NAO má k dispozici dva ultrazvukové senzory umístěné na hrudníku. Senzory jsou využívány k detekci překážek před robotem a k orientaci v prostoru. Senzory pracují na frekvenci 40 kHz. V nejnovější verzi robota V5 snímají sonary vzdálenost od 20 do 250 centimetrů s přesností 1 až 4 centimetry v závislosti na vzdálenosti. Nejspolehlivěji fungují od 20 do 80 centimetrů. Pokud je objekt vzdálen méně než 20 centimetrů od robota, sonary nedokážou určit, jak daleko se objekt nachází. Uživatel pouze dostane informaci, že je před robotem překážka.[1]




Obrázek 1.3: Kamery, infračervené senzory a ultrazvukové senzory

### 1.2.6 Tlačítka

Na robotovi je umístěno celkem 12 uživatelsky přístupných tlačítek, které lze využít pro řízení robota během chodu programu. Tři dotykové senzory se nachází na vrchní straně hlavy robota, jejich stisk je doprovázen rozsvícením LED diod umístěných okolo tlačítek. Na každé ruce jsou umístěny tři dotykové senzory, při jejich stisku nemá uživatel žádnou zpětnou vazbu. Hlavní tlačítko je umístěno na hrudi robota. Dlouhým stiskem můžeme robota zapnout či vypnout. Po krátkém stisku hlavního tlačítka o sobě robot řekne základní údaje, a to své jméno a IP adresu. Stisk hlavního tlačítka je doprovázen rozsvícením LED diody. Poslední dvě tlačítka jsou umístěna na špičkách nohou. Slouží především jako detekce překážky a jejich stisk doprovází rozsvícení LED diody na nártu nohy.[1]




Obrázek 1.4: Tlačítka, LED diody a mikrofon

### 1.2.7 LED diody

Robot je osazen celkově 51 LED diodami. Diody jsou využity v autonomním režimu k vyjádření nálady robota, mohou signalizovat chybová a systémová hlášení nebo zajišťují zpětnou vazbu pro dokončené uživatelské interakce. Diody jsou uživa-

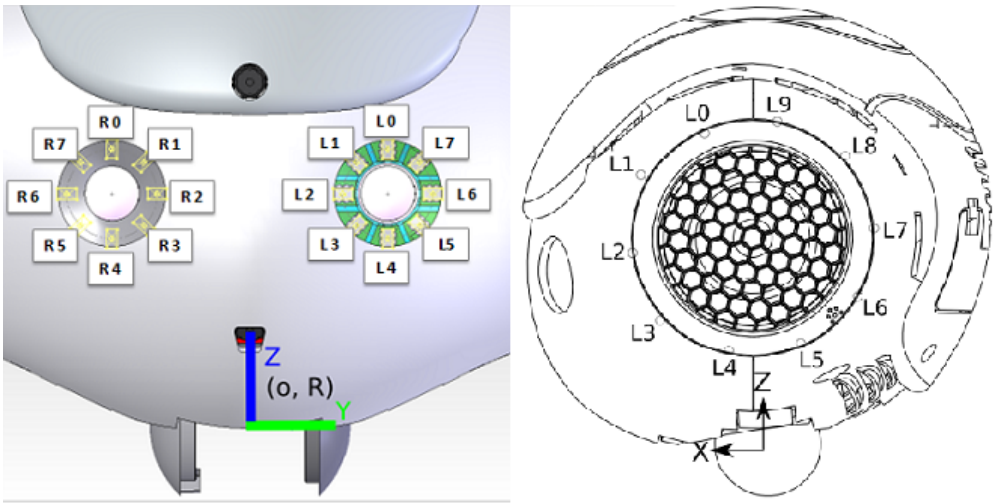
telsky přístupné, každou z diod lze ovládat jednotlivě, uživatel je tedy může využít ve svých programech.[1]

Obě oči jsou vybaveny 8 LED diodami. Jedná se o RGB diody, tudíž jsou schopny svítit více barvami. K dispozici jsou kromě základních tří barev i jejich předdefinované kombinace.[1]

Na stranách hlavy se kolem reproduktorů nachází v kruhu rozmístěných 10 LED diod. Tyto diody jsou stejného typu jako diody v očích. Mohou tedy svítit různými barvami.[1]

Posledních 12 LED diod na hlavě robota je umístěno kolem dotykových tlačítek. Diody slouží jako zpětná vazba pro uživatele při interakci s tlačítkem. Při každém stisku dotykového senzoru se rozsvítí diody umístěné kolem konkrétního tlačítka.[1]

Jedna RGB LED dioda je umístěna pod hlavním tlačítkem na hrudníku robota. Signalizuje zapínání robota či potřebu nabíjení. Poslední dvě LED diody najdeme na nártech robota. Tyto diody reagují na stisk dotykového senzoru umístěného na špičce každé nohy.[1]




Obrázek 1.5: LED diody na hlavě robota [1]

### 1.2.8 Inerciální jednotka

Robot NAO je vybaven inerciální jednotkou. V nejnovější verzi V5 obsahuje 3osý akcelerometr a 3osý gyroskop. Nižší verze mají k dispozici 3osý akcelerometr,

ale pouze 2osý gyroskop, snímání v ose Z není k dispozici. Výstupy z gyroskopu a akcelerometru jsou využívány především systémem robota pro detekci pádu. Data ze senzorů jsou ovšem přístupná i uživateli a lze je využít ve vlastních programech. Inerciální jednotka v nejaktuálnější verzi robota umožňuje detekovat úhly bočení (yaw), klopení (pitch) a klonění (roll). Všechny úhly jsou měřeny v radiánech. Poloha torza robota je standardně vyčítána z hodnot akcelerometru. Při pohybu robota je úhel dopočítáván z gyroskopu, kvůli jeho lepším dynamickým vlastnostem při pohybu.[1]

### 1.2.9 Odporové senzory síly

Robot je vybaven dvěma odporovými senzory síly, které jsou umístěny v chodidlech robota. Senzory lze měřit sílu od 0 do 25 N. Na chodidlo robota působí síla, která mění odpor senzoru, jenž je měřen. Systém robota předává údaje ze senzorů manažeru pádů. Manažer pádů je systém, který se stará o co nejbezpečnější pád robota na zem. Při ztrátě rovnováhy a následném pádu zajistí manažer pádů, aby robot zaujmul bezpečnou pozici. V bezpečné pozici má robot natažené ruce před sebou a vypnuté napájení kloubů robota. Cílem je, aby se robot při pádu nepoškodil.[1]

## 1.3 Softwarové vybavení

Robot běží pod operačním systémem s názvem NAOqi OS, což je speciální linuxová distribuce založená na verzi Gentoo. Operační systém NAOqi OS byl vytvořen speciálně pro ovládání robota NAO na základě požadavků firmy Aldebaran Robotics. Jedná se o vestavěný (embedded) operační systém, jenž poskytuje a načítá mnoho knihoven a dalších programů, které robot NAO potřebuje pro svůj chod. Na webových stránkách výrobce je ke stažení virtuální verze operačního systému NAOqi OS. Tuto verzi lze spustit ve virtualizačním programu, například VMware nebo VirtualBox. Takto lze ladit uživatelské programy bez nutnosti mít robota fyzicky u sebe.[1] [13]

## 1.4 Softwarové rozpoznávací algoritmy

Součástí robota NAO je několik výrobcem implementovaných rozpoznávacích algoritmů, které můžete jako uživatel plně využít. Pomocí těchto rozpoznávacích algoritmů umí robot rozpoznat a následovat různé objekty. Abyste mohli rozpoznávací algoritmy využít, musíte do programu implementovat modul ALTracker, který se stará o všechny funkcionality rozpoznávacích algoritmů. Modul má za úkol propojit detekování žádaného objektu s pohybem robota pro udržení vizuálního kontaktu s rozpoznaným objektem. Modul nabízí několik možností, jak může robot rozpoznaný objekt sledovat.[1]

### 1.4.1 Detekovatelné objekty

Pomocí modulu ALTracker může robot rozpoznat a sledovat různé objekty, které se v kódu specifikují jako parametr targetName. Podle tohoto parametru modul ALTracker následně volá jednotlivé moduly specifické pro každý objekt. Rozpoznat lze červený míček o různých velikostech, obličeje a osoby. Robota můžeme naučit rozpoznávat i jednotlivé obličeje. Ve fázi učení pořídí robot 5 snímků pro každý obličej, z nichž se ho pokusí zapamatovat.[1]

Robot dále rozpoznává speciální značky nazývající se NAO marky, které jsou složeny z tmavého kruhu s bílými trojúhelníkovými lopatkami. Různá umístění lopatek specifikují jednotlivé NAO marky. Pro detekci a rozpoznání značek slouží modul ALLandMarkDetection. Značky mohou sloužit jako orientační body pro robota v prostoru.[1]

Pro rozpoznávání všech detekovatelných objektů jsou velmi důležité okolní podmínky, a to především intenzita osvětlení prostoru. Při špatných světelných podmínkách, což je například šero nebo když do kamery robota svítí přímé světlo, je detekce objektů velmi špatná. Při dobrém osvětlení lze robotem rozpoznat objekty na vzdálenost zhruba 1,5 až 2 metry.[1]

### 1.4.2 Možnosti sledování objektu

Robot je schopen sledovat výše uvedené objekty několika způsoby. Defaultně je v modulu ALTracker nastaven v režimu Head, v něm sleduje objekt pouze pomocí otáčení a naklánění hlavy. V režimu WholeBody udržuje vizuální kontakt s objektem pomocí pohybu celého těla. Robot se za objektem může otáčet celým tělem. V případě, že je objekt nízko, NAO je schopen změnit polohu ze stoje do sedu nebo dřepu. Stabilita je během sledování objektu zajištěna autonomně.[1]

Dalším režimem je Move, ve kterém robot za objektem chodí a snaží se dodržovat vzdálenost od objektu definovanou uživatelem. Stabilita při pohybu je opět zajištěna autonomně, nicméně mezi robotem a sledovaným objektem nesmí být překážky, jelikož je robot v průběhu sledování nedokáže autonomně detekovat.[1]

# 2 Aktuální stav kooperace robotů

Všeobecně v současné době již není problém ovládat jednoho či více průmyslových robotů a řešit pomocí nich složité úlohy. Může se jednat o manipulaci s objekty či úkony s koncovým nástrojem, například svařování, šroubování atd. V oblasti humanoidních robotů je však kooperace dvou a více robotů stále ještě nedokonalá.

Hlavní motivací pro vývoj robotů, ať už se jedná o průmyslová ramena či humanoidní roboty je, že mohou manipulovat s objekty. Tato schopnost může být užitečná pro širokou škálu operací zahrnující přepravu předmětu z jednoho místa na druhé. Humanoidní roboti mají navíc tu výhodu, že nejsou stacionárně umístěni na jednom místě, ale mohou se s objektem pohybovat v prostoru. Problém však nastane u přemisťování větších a těžších objektů, jelikož nosnost robotů je značně omezena. Jeden způsob, jak se vypořádat s tímto problémem, je přemisťovat těžký, či objemný předmět pomocí více robotů zároveň, tak jak to dělají lidé.[2] [10]

Na téma přepravy objektu pomocí více spolupracujících robotů bylo provedeno mnoho studií. Většina z nich ale byla prováděna s více kolovými roboty, kteří si najdou žádaný předmět a rozmístí se kolem něj tak, aby ho mohli odtlačit v požadovaném směru. V tomto řešení přichází manipulátor do styku s přemisťovaným objektem pouze v jednom bodě, což sotva umožňuje jakoukoliv kontrolu nad předmětem při jeho pohybu a manipulaci. Koloví roboti jsou o mnoho méně nároční na ovládání oproti humanoidním robotům a jsou především používáni k přesunutí předmětů ležících na zemi, a to pouze posunutím objektu.[2] [9]

Roboti s humanoidními rameny mají lepší kontrolu nad předmětem a jsou proto vhodnější pro manipulaci s objekty různých tvarů. Vyspělé a specializované modely kolových robotů mohou mít humanoidní torzo a ramena, tak jako například roboti

ARMAR, které jim umožňují udržet vysokou úroveň kontroly při držení a přepravě předmětů. Humanoidní torzo a ramena také robotům dovolují interakci s prostředím či lidmi v reálném čase. Většina prostředí však byla vytvořena člověkem tak, aby pro něj byla vhodná, proto se do těchto prostředí nejvíce hodí plně humanoidní roboti. Tématu přepravy objemných nebo těžkých předmětů s humanoidním robotem byla věnována pozornost v minulých letech a bylo vyvinuto mnoho rozdílných přístupů k řešení tohoto problému. Jedním z řešení byla kooperace mezi člověkem a humanoidním robotem. Toto řešení bylo použito například pro přepravu předmětu či zvedání objektu ze země. Další řešení zkoumají spolupráci více humanoidních robotů k přepravě objektu.[2] [10]

## 2.1 Přístupy kontroly humanoidních robotů při manipulaci s objektem

Jedním z nejoblíbenějších schémat kontroly kooperace více robotů, ať už velkých či malých, je leader (vedoucí) – follower (následovník) kontrolní schéma. Jeden z robotů je vedoucí, v závislosti na jeho pozici a okolí vypočítává společný plán pohybu systému nebo je přímo řízen lidským operátorem. Druhý robot, následovník, jednoduše následuje vedoucího robota. Toto schéma je relativně jednoduché implementovat, ale má mnoho problémů při použití v uzavřené smyčce kooperace, což je důvod proč většina řešení používá nesouvislé objekty nebo má omezený pohyb robotů pouze v jedné ose. Vzhledem k tomu, že následovník reaguje pouze na vedoucího až poté co se začne hýbat, vznikne mezi roboty významné časové zpoždění. Chyby interpretace pohybů vedoucího robota mohou destabilizovat uzavřenou smyčku systému a mohou způsobit neočekávaný pád.[2]

Dalším populárním kontrolním schématem, jak vyřešit problém přemístění objektu, je použít sadu synchronizovaných regulátorů. V tomto schématu neexistuje žádný zřejmý robot leader. Externí centralizovaný ovládací systém řídí všechny roboty současně na základě všech informací poskytnutých od robotů. Výsledkem je, že jsou roboti synchronizovaní, společně začnou i ukončí pohyb. V tomto řešení je sys-

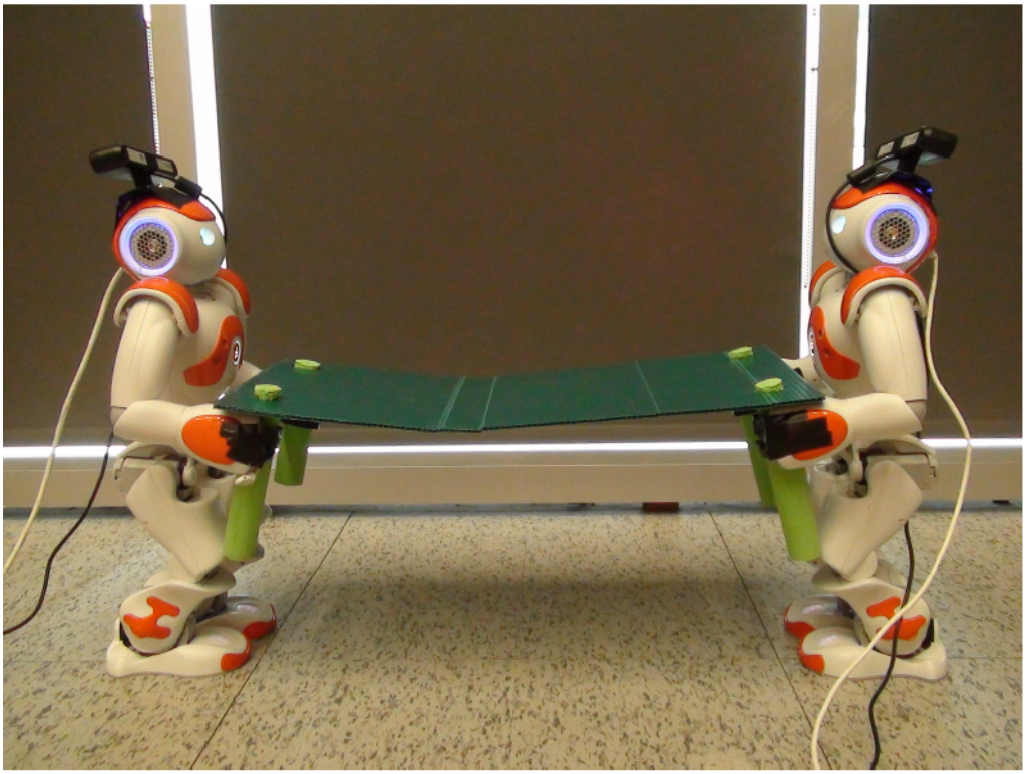
tém velmi citlivý a jakákoliv chyba je sdílena v rámci celého systému. Pro dosažení tohoto cíle lze modelovat celý systém robot-předmět-robot jako čtyřnohý s pevným tělem, čímž systém robotům brání volně manipulovat s objektem. Dynamika systému je jednoduchá a mnoho stupňů volnosti lze odstranit přidáním virtuálních omezení. Studie viz [7] prozkoumala tuto strategii s robotem lidské velikosti HRP2. I přesto, že jejich výsledky v simulaci vypadaly slibně, jejich implementace neuvažuje navigaci mezi překážkami ani lokalizaci robota a mapování. V práci se rozsáhle využívají drahé šestiosé silové senzory umístěné v zápěstí robota, což dělá tento přístup špatně zobecnitelným pro řadu dostupných robotů, kteří nejsou vybaveni takovým snímačem síly.[2]

## 2.2 Komplexní řešení kooperace dvou humanoidních robotů

### 2.2.1 Manipulace objektu spolu s mapováním okolí

Práce viz [2] se zabývá spoluprací dvou humanoidních robotů NAO, kteří manipulují s předmětem po požadované trajektorii a zároveň jsou schopni se vyhýbat překážkám a mapovat své okolí. Roboti mají sdílené informace o mapování prostředí a požadované trajektorii. Část navrhovaného rámce, jako například výpočet požadované trajektorie, mapování prostředí pomocí systému SLAM a vizuální zpětná vazba, je prováděna na externím počítači, jelikož roboti NAO mají v porovnání s ostatními špičkovými humanoidními roboty velmi limitované výpočetní schopnosti. Výsledky jsou poté přenášeny do robotů prostřednictvím ethernetového připojení. Hlavním přínosem této práce je poskytnout kompletní rámec pro navigaci kooperativních autonomních humanoidních robotů v přeplněném prostředí, zatímco manipulují s objektem v uzavřeném kinematickém řetězci. Rámec využívá přístup kontrolního schématu synchronizovaných regulátorů, akce jsou prováděny synchronně na základě centrálně rozhodnuté trajektorie. Ve studii se řeší dílčí problémy spojené s navigací systému robot-předmět-robot v přeplněném prostředí. Je zde naznačeno

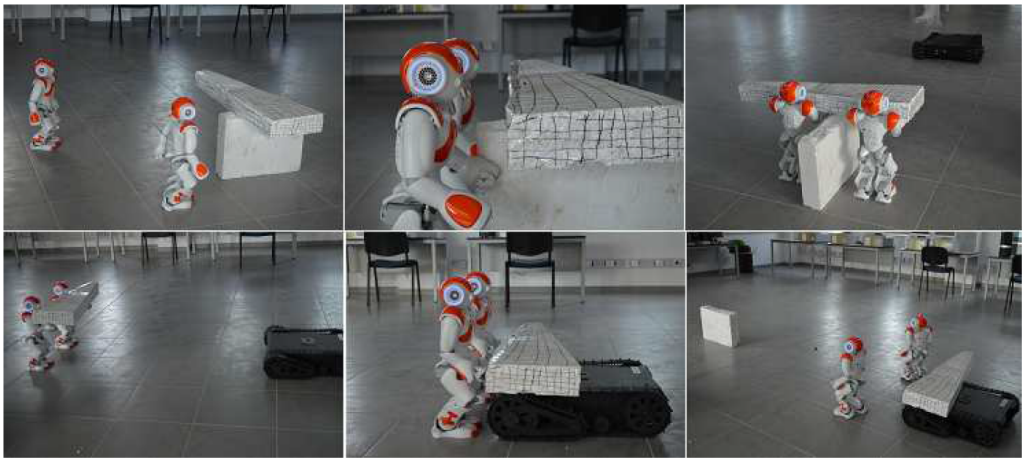
řešení problému plánování, ovládání předmětu, snímání prostředí a synchronizace. V práci ovšem není řešeno, jak roboti objekt identifikují a přijdou k němu. Neuvádí se zde ani to, jak roboti objekt zvednou, práce už počítá s tím, že roboti předmět drží v rukou a mají ho v nich pevně přichycený lepicí páskou.[2]



Obrázek 2.1: Manipulace s objektem a mapování okolí [2]

### 2.2.2 Navigace k objektu a jeho manipulace

Práce viz [3] se zabývá kooperací dvou humanoidních robotů NAO, kteří mají společně udělat komplexní úlohu, a to přenesení předmětu různých tvarů a váhy z podstavce na balicí plošinu. Celý algoritmus je složen ze šesti různých částí. Nejprve roboti identifikují a zaměří objekt pomocí rozpoznání barvy předmětu a zapamatují si souřadnice objektu. V další fázi se přesouvají k předmětu a následně ho uchopí a zvednou. Předmět je umístěn v takové výšce, že si roboti nemusí dřepnout, objekt zvednou ve stoje pouhým pohybem rukou před sebe a nahoru. Ve čtvrté fázi roboti přesouvají objekt na balicí plošinu. V další části objekt položí na balicí plošinu. V poslední fázi se vrací zpět do výchozích pozic.[3]



Obrázek 2.2: Navigace k objektu a jeho manipulace [3]

Ve studii je vcelku jednoduše řešeno uchopení objektu, jelikož je umístěn na vysokém podstavci, robotům stačí k němu přijít, natáhnout ruce a nadzdvihnout objekt. Dále program počítá s tím, že je balicí plošina umístěna v jedné rovině jako místo, kde se nachází předmět a navíc i v konkrétní vzdálenosti. Není zde žádná identifikace místa, kde je balicí plošina. Robotům stačí, aby se po uchopení objektu pohybovali po pevně daných souřadnicích.[3]

## 2.3 Všeobecná kooperace robotů

Spolupráce robotů se netýká pouze humanoidních, či průmyslových robotů, ale rozvíjí se i u dalších typů robotů. V poslední době se vyvíjí kooperace v takzvaných swarmech neboli rojích robotů. Jedná se o několik desítek až stovek většinou menších robotů, kteří společně pracují na dosažení zadaného cíle.

Jedním z hlavních cílů robotického výzkumu rojů je navrhnout robustní, škálovatelné a flexibilní kolektivní chování pro více autonomních robotů. Jednoduchá pravidla a lokální interakce mezi jednotlivými roboty vedou k požadovanému kolektivnímu chováni roje. Mezi roboty probíhají samoorganizované koordinační mechanismy. Chování hmyzu, zkoumané v biologických studiích, lze efektivně uplatnit v robotických rojích.[4]

### 2.3.1 Shromažďování předmětů za pomoci roje robotů

Výzkum viz [4] se soustředí na chování robotů při shromažďování předmětů. Snaží se vyvinout efektivní, decentralizovaný, vyhledávací algoritmus pro sběr předmětů, připomínající chování mravenců. Roboti vyhledávají v prostředí předměty, sbírají je a nosí na určené místo, skladiště. Toto řešení by se dalo v reálném světě využít například pro sběr nebezpečných materiálů.[4]

### 2.3.2 Kooperace dronů pro účely autonomního plánování kinematografie

Zajímavou kooperací se zabývá práce viz [5], a to spoluprací více dronů pro účely autonomního plánování filmování. Použití dronů pro leteckou kinematografii se již stalo trendem. Tím pádem se otevírají nové možnosti pro autonomní aplikace dronů. V práci je uvedena první verze návrhu architektury pro kooperativní plánování v kinematografických aplikacích, jako jsou venkovní sportovní události. Hlavní funkce této architektury jsou následující. Systém by měl být schopen reprodukovat typické záběry z kinematografických pravidel autonomně. Měl by umožňovat snímat jak statické tak pohybující se objekty. Měl by zajistit hladké přechody mezi záběry, implementovat vyhýbání se kolizím a uvědomovat si bezletové zóny. V neposlední řadě musí zajistit bezpečnost v nouzových situacích.[5]

V této práci vidím budoucnost televizních přenosů ze sportovních událostí. Tým autonomních dronů by se mohl uplatnit například při natáčení fotbalových utkání nebo cyklistických závodů. Drony by mohly autonomně sledovat hráče nebo cyklisty z různých pohledů, které jsou nyní nedostupné.

# 3 Možnosti realizace a návrh softwarového řešení kooperace

## 3.1 Možnosti komunikace robotů

Ještě před samotným navrhováním řešení spolupráce robotů NAO jsem si musel zvolit, jak mezi sebou budou roboti komunikovat.

### 3.1.1 Komunikace pomocí infračervených senzorů

Jako jedna z možností se nabízela komunikace, která by využívala robotovi infračervené senzory umístěné v očích. Infračervené senzory lze nastavit jako vysílač nebo přijímač infračerveného signálu. Jeden robot by tedy mohl být master, měl by nastavené vysílání signálu a dával by příkazy druhému robotu, který by byl slave. Slave by byl nastaven jako přijímač a dělal by to, co by mu master poslal. Problémem u tohoto řešení je, že nemá žádnou zpětnou vazbu. Zpráva od mastra by se ke druhému robotovi nemusela vůbec dostat a master by o tom nevěděl. Tento problém se dá vyřešit tak, že poté co master odešle signál slavovi přepne se do módu přijímače signálu. Slave, který přijme signál od mastra, se následně přepne do módu vysílače a odešle mastrovi zpětnou vazbu, že úspěšně dostal jeho zprávu. Tato komunikace již má zpětnou vazbu, ale stále je zapotřebí, aby roboti udržovali neustálý oční kontakt při jejich komunikaci. To může být problém při přenášení velkého objektu, ale například i u nějakého jednoduchého pohybu nebo tance, kdy by se oba roboti otočili jiným směrem a ztratili pojem o tom, kde se nachází ten druhý. Tato možnost počítá s komunikací pouze dvou robotů a není nijak rozšiřitelná.

Tento přístup komunikace navíc nedoporučuje ani výrobce robota. Proto jsem tuto komunikaci jen okrajově vyzkoušel a neuplatňoval ji ve výsledném řešení.

### 3.1.2 Komunikace pomocí řeči

Tento způsob komunikace je inspirován lidskou komunikací. Šlo by o komunikaci pomocí hlasu robotů, tedy využíval bych jejich zabudované reproduktory a mikrofony. Jeden robot by byl opět master a druhý slave. Roboti by komunikovali tak, že by si říkali heslovité, například jen jednoslovné zprávy, a to hlavně z toho důvodu, aby byly dobře rozpoznatelné. Tento způsob dorozumívání se mezi roboty by se dal použít i pro více robotů než pouze pro dva. Zajímavá je u tohoto řešení možnost zapojit člověka do komunikace s roboty. Člověk by mohl dávat robotům slovní příkazy. Tato metoda komunikace má zpětnou vazbu a lze ji použít, aniž by na sebe museli roboti koukat. Problémem je veliká chybovost detekce zvuku. Další nevýhodou tohoto přístupu, je, že v prostředí, kde roboti pracují, musí být ticho, aby nevznikaly šumy, které by komunikaci znemožňovaly. Roboti jsou například při chůzi celkem hluční a celkově je dost náročné udržet v pracovním prostředí ticho. Další nevýhoda je, že se pomocí řeči dá přenést poměrně malé množství informací. Tuto komunikaci jsem tedy kvůli jejím nárokům na okolní prostředí a chybovosti nevyužil ve výsledném řešení.

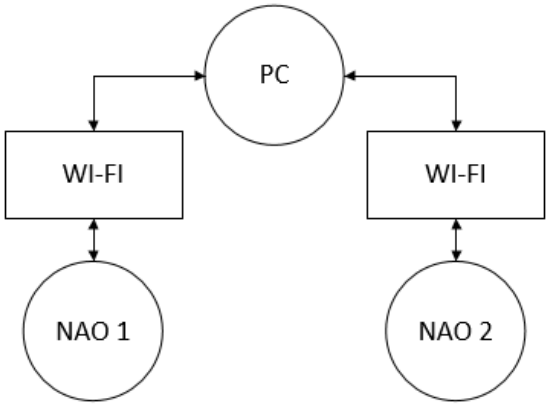
### 3.1.3 Komunikace pomocí WiFi sítě

První možností, jak by mohli roboti komunikovat přes bezdrátovou síť, je vytvořit program, který bude nahrán v jednom robotu, mastru. Na mastru by běžel program, který by se spojoval s dalšími roboty jako se slavy a posílal jim příkazy. Výhodou je, že by roboti mohli program vykonávat ihned po zapnutí. Toto řešení nevyžaduje žádný prvek v síti navíc. Problém je, že roboti již nejsou nejnovější a jejich hardware by některé výpočty prováděl dlouho. Program by musel být vytvářen ve vývojovém prostředí Choregraphe, jelikož jenom programy vytvořené v tomto prostředí jdou spouštět již při zapnutí robota. Z důvodu malého hardwarového výkonu robota jsem

tento přístup nepoužil v mém řešení.

Druhou možností komunikace je vytvořit pro každého robota program s využitím SDK a nějakého programovacího jazyka. Tento program by se nahrál do každého robota a následně by se spouštěl pomocí počítače připojeného ve stejné síti. Roboti by nekomunikovali mezi sebou ale pouze s počítačem. V počítači by běžel hlavní program, který by v robotech spouštěl různé metody, které by měly nadefinované ve svém programu v paměti. Metody by počítač spouštěl na základně informací, které by mu roboti posílali o své pozici atd. Nevýhodou tohoto řešení je, že by se do každého robota musel zvlášť nahrávat program, který by byl v každém robotu obdobný. Proto jsem tento přístup nepoužil.

Poslední možnost komunikace, kterou jsem využil ve výsledném řešení, opět využívá princip komunikace robotů pouze s řídícím počítačem a ne mezi sebou. Oba roboti jsou připojeni k WiFi síti a komunikují pouze s počítačem, který je ve stejné síti. Řídící počítač v podstatě využívá roboty jako vstupně-výstupní zařízení. Všechny výpočty a zprávy mezi roboty jsou prováděny na počítači, ze kterého se robotům posílají již konkrétní úkoly, které mají udělat. Tento přístup komunikace jsem zvolil kvůli tomu, že není omezena okolním prostředím robotů. Všechny výpočetní operace běží na počítači, takže je ulehčeno výpočetnímu výkonu robota, který nepatří mezi nejlepší. Náročnější výpočty by na hardwaru robota mohly trvat dlouho. Díky přesunu všech výpočtů z robota na počítač je ulehčeno baterii robota, která vydrží déle nabitá. Pokud mají roboti vykonávat stejný úkol, stačí naprogramovat metodu jen jednou a spustit ji pro více robotů. Tento přístup komunikace má samozřejmě i své nevýhody. Nejzásadnějším problémem je jiná odezva každého z robotů, přičemž průměrný rozdíl je zhruba 90 milisekund. V důsledku tohoto rozdílu nastává problém se synchronizací, při pohybu obou robotů zároveň. Další nevýhodou je neustálá potřeba řídícího počítače.




Obrázek 3.1: Schéma komunikace mezi roboty a PC

## 3.2 Možnosti realizace

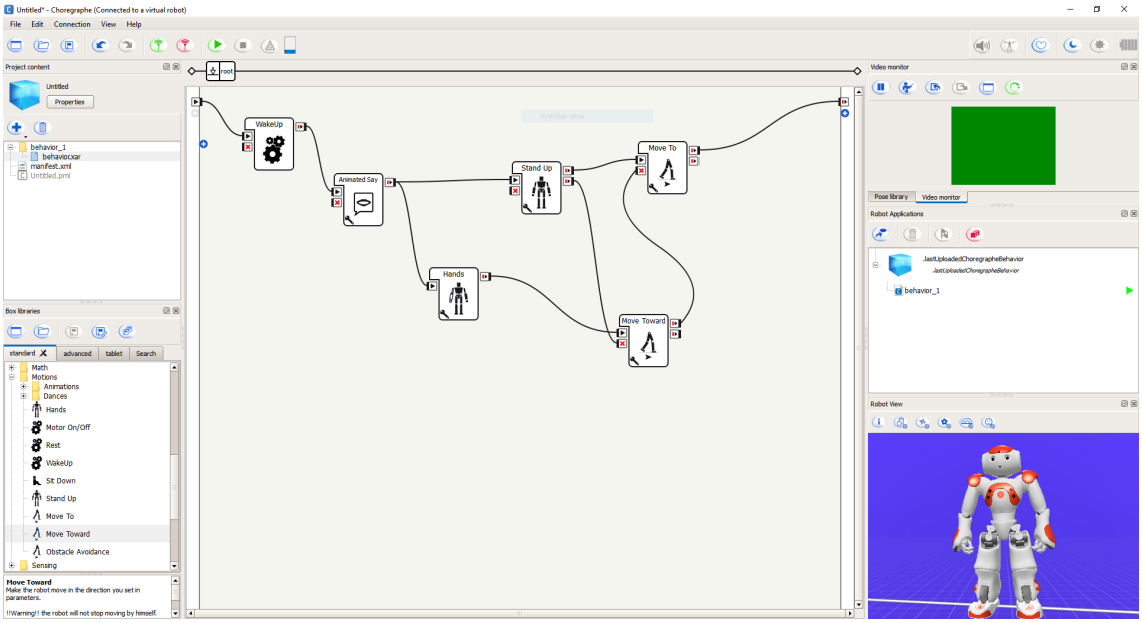
### 3.2.1 Využít vývojové prostředí Choregraphe

Jednou z možností pro realizaci návrhu bylo využít vývojové prostředí Choregraphe viz Obr.3.2, které poskytuje výrobce robota a je určeno speciálně pro programování robota NAO. Jedná se o velmi jednoduché a intuitivní prostředí, které je určeno spíše pro uživatele, kteří s robotem a jeho programováním nemají předešlé zkušenosti. Programování probíhá pomocí předem definovaných bloků, které se mezi sebou spojují a tím vytváří výsledný program pro ovládání robota.[13]

Výhodou Choregraphu je, že kromě jednoduchého a celkem rychlého naprogramování robota nabízí několik dalších funkcí. Jednou z nich je 3D náhled na robota, který v reálném čase kopíruje pohyby reálného robota. Další užitečná funkce je možnost sledovat výstupy z kamer na hlavě robota v reálném čase. Celková přednost realizace pomocí Chopegraphu je, že není potřeba řídícího počítače. Vytvořený program se nahrává přímo do paměti robota, ze které se spouští při jeho startu.

Možnosti programování v Choregraphe jsou omezené právě předpřipravenými bloky. Je zde sice možnost vytvářet vlastní bloky, ale k tomu je zapotřebí znát programovací jazyk Python a pochopit strukturu, kterou jsou bloky programovány. Komplexnější úlohy jsou navíc kvůli stylu programování nepřehledné. Z těchto důvodů jsem se s Choregraphem pouze seznámil, vyzkoušel napsat několik jednoduchých

programů, ale výslednou realizaci mého softwaru jsem v něm nevyvíjel.




Obrázek 3.2: Vývojové prostedí Choregraphe

### 3.2.2 Využít SDK

Možnost realizace mého návrhu, kterou jsem si vybral, je využít výrobcem nabízené vývojové nástroje SDK v kombinaci s podporovaným programovacím jazykem Python, ve verzi 2.7. Existují již novější verze Pythonu, ale tato je jako jediná plně podporovaná SDK od výrobce. Při využití vývojových nástrojů je možné si vybrat, ze dvou způsobů realizace aplikace.

Soubor s programem lze nahrát do paměti robota, kde je následně spuštěn pomocí počítače, který dále k běhu programu není potřeba. Nevýhodou je, že robot již není nejnovější a výkon jeho hardwaru je malý.

Druhou možností, kterou jsem zvolil jako řešení, je spouštět moji aplikaci v počítači, který s roboty naváže spojení přes WiFi síť a využívá je jako vstupně-výstupní zařízení. Výhody tohoto řešení jsou uvedeny viz Komunikace pomocí WiFi sítě.

## 3.3 Návrh softwaru pro kooperaci

Jak už je napsáno výše pro návrh softwaru pro kooperaci jsem využil SDK od výrobce v kombinaci s programovacím jazykem Python. V návrhu je využita knihovna multiprocessing pro Python. Multiprocessing je balíček, který podporuje rozmnožování procesů a je určen k paralelismu, založeném na procesech. Používá stejné API jako modul threading.[6]

Kooperace je řešena tak, že se na řídícím počítači spustí testovací aplikace. Ta běží v hlavním procesu. V hlavním procesu jsou vytvářeny další procesy. Jeden proces pro každého robota zapojeného do spolupráce. Tyto procesy běží paralelně vedle sebe, tudíž roboti mohou pracovat současně na zadaném úkolu. Zároveň s procesem se vytváří i komunikační kanál, a to mezi hlavním procesem a nově vytvořeným procesem.

Všechny procesy, jak hlavní, tak i procesy pro jednotlivé roboty, běží na počítači. Každý proces komunikuje s jedním robotem a ovládá ho jako vstupně-výstupní zařízení. Po vytvoření procesu se v něm spustí libovolná funkce, jejíž parametry musí být minimálně IP adresa robota a port, po kterém bude probíhat komunikace, aby se funkce mohla k robotu připojit a ovládat ho. Po dokončení úkolu, respektive funkce, si procesy pošlou informace o výsledku. V hlavním procesu, který má informace od všech dalších procesů, se dá jednoduše vytvořit synchronizace úkonů. Stačí zadat podmínku, která čeká na informace od dvou, či více procesů. Jakmile je podmínka splněna, tak se do procesů nahraje jiná funkce, která robotům zadá další pokyny.

Více procesů může volat stejnou funkci s jinými vstupními parametry, tudíž pokud mají roboti vykonávat stejnou činnost, použije se jedna funkce pro všechny. Odpadá tedy potřeba programovat pro každého robota speciální program nebo do něj nějaký program nahrávat. Díky tomu, že roboti mohou používat stejné funkce, lze kooperaci snadno přeprogramovat na jinou.

Roboti v tomto přístupu ke kooperaci nejsou autonomní a potřebují pevný scénář toho, co budou dělat. Ovšem je zde možnost větvení, založeném na informacích od robotů. Například pokud roboti nedokážou dokončit úlohu, kvůli překážce, nahlá-

sí to hlavnímu procesu, a ten pokud je na tuto možnost připraven, spustí v procesech ovládající roboty funkce pro vyhnutí se překážce.

### 3.3.1 Ukázka založení procesů a komunikace mezi nimi

```
1  import multiprocessing
2
3  def funkce1(que):
4      #zde je kod funkce#
5
6      #poslani zpravy do fronty#
7      que.put("zprava")
8
9  def funkce2(que):
10     #zde je kod funkce#
11
12     #poslani zpravy do fronty#
13     que.put("zprava")
14
15 if __name__ == '__main__':
16     #zalozeni komunikace Queue#
17     q_1 = multiprocessing.Queue()
18     q_2 = multiprocessing.Queue()
19
20     #zalozeni procesu#
21     p_1 = multiprocessing.Process(target=funkce1, args=(q_1,))
22     p_2 = multiprocessing.Process(target=funkce2, args=(q_1,))
23
24     #start procesu#
25     p_1.start()
26     p_2.start()
27
28     #cekani na zpravy ve frontach#
29     if q_1.get() == "zprava" and q_2.get() == "zprava":
30         #kod, co se ma stat po obdrzeni zprav#
31         #kod se provede po obdrzeni zprav od obou procesu#
```

### 3.3.2 Komunikace mezi procesy

Procesy si mezi sebou mohou vyměňovat informace více způsoby. Já jsem v mém řešení zvolil takzvanou frontu (Queue), kdy jeden z procesů pošle do fronty informaci funkcí put(). Další proces informaci z fronty dostane pomocí funkce get(). Frontu lze nastavit do několika režimů pro manipulaci s daty. Já zvolil režim FIFO. První informace, která je do fronty vložena, je následně jako první vybrána. Fronta se dá využít ke komunikaci mezi více než dvěma procesy, já ji ovšem využívám pouze ke komunikaci mezi dvěma procesy, a to hlavním procesem a jednotlivými procesy robotů.[6]

### 3.3.3 Úskalí synchronizace

Jednou z nevýhod mého návrhu softwarového řešení kooperace robotů je nedokonalá synchronizace. Procesy, které ovládají jednotlivé roboty, jsou v kódu spouštěny postupně za sebou viz Ukázka založení procesů a komunikace mezi nimi. Problémem je latence, která vzniká při spouštění jednotlivých procesů. Například se v programu spustí první proces a než se spustí druhý proces, tak uplyne určitá doba.

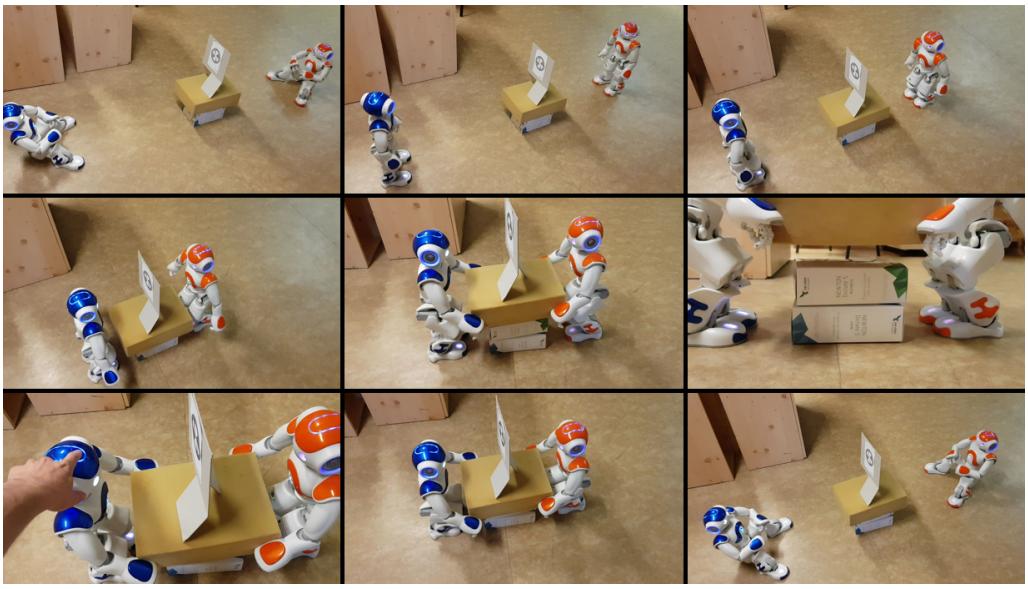
Latenci mezi spouštěním jednotlivých procesů lze jednoduše zjistit a několika měřeními zprůměrovat. Zprůměrovaná hodnota je zhruba 35 milisekund. Problém se synchronizací tedy lze částečně vyřešit tak, že v procesech, které se spouští jako první, se provede funkce čekání, a to po dobu, která byla zjištěna a zprůměrována. Latence má ovšem vždy náhodnou složku, tudíž roboti nebudou nikdy úplně synchronizovaní.

# 4 Realizace ukázek spolupráce a testovací aplikace

Po návrhu softwarového řešení spolupráce robotů bylo potřeba zrealizovat aplikaci, která by prakticky předvedla funkčnost navrhovaného řešení. K dispozici jsem měl dva humanoidní roboty NAO, tudíž jsem mohl realizovat ukázku kooperace pouze těchto dvou robotů.

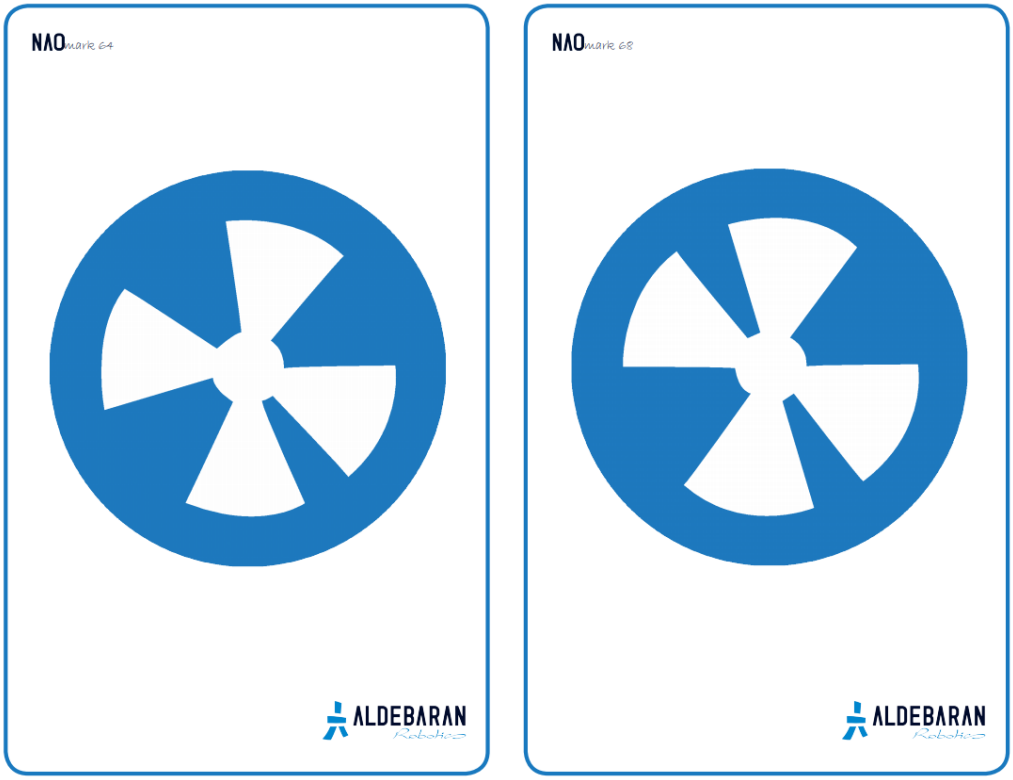
## 4.1 Kooperace - zvednutí objektu

V této praktické ukázce kooperace mají roboti NAO za úkol najít objekt v prostoru, přijít k němu, společně ho zvednout, čekat na interakci člověka, poté objekt položit a sednout si do počáteční polohy.



Obrázek 4.1: Kooperace - zvednutí objektu

Pro vyhledávání objektu v prostoru jsem využil NAO Marky viz Obr.4.2, což jsou speciální značky, které jsou složeny z tmavého kruhu s bílými trojúhelníkovými lopatkami. Různá umístění lopatek specifikují jednotlivé NAO Marky. Pro detekci a rozpoznání značek slouží modul ALLandMarkDetection. NAO Marky jsem umístil na vyhledávaný objekt. Robot dokáže díky modulu ALLandMarkDetection rozpoznat velikost NAO Marku a určit velikosti úhlů pod kterými NAO Mark snímá v osách X a Y. Tato orientace v prostoru je jednoduchá, ale velmi efektivní. Díky snímaným odchylkám je robot schopen se vycentrovat a přijít k objektu rovně.

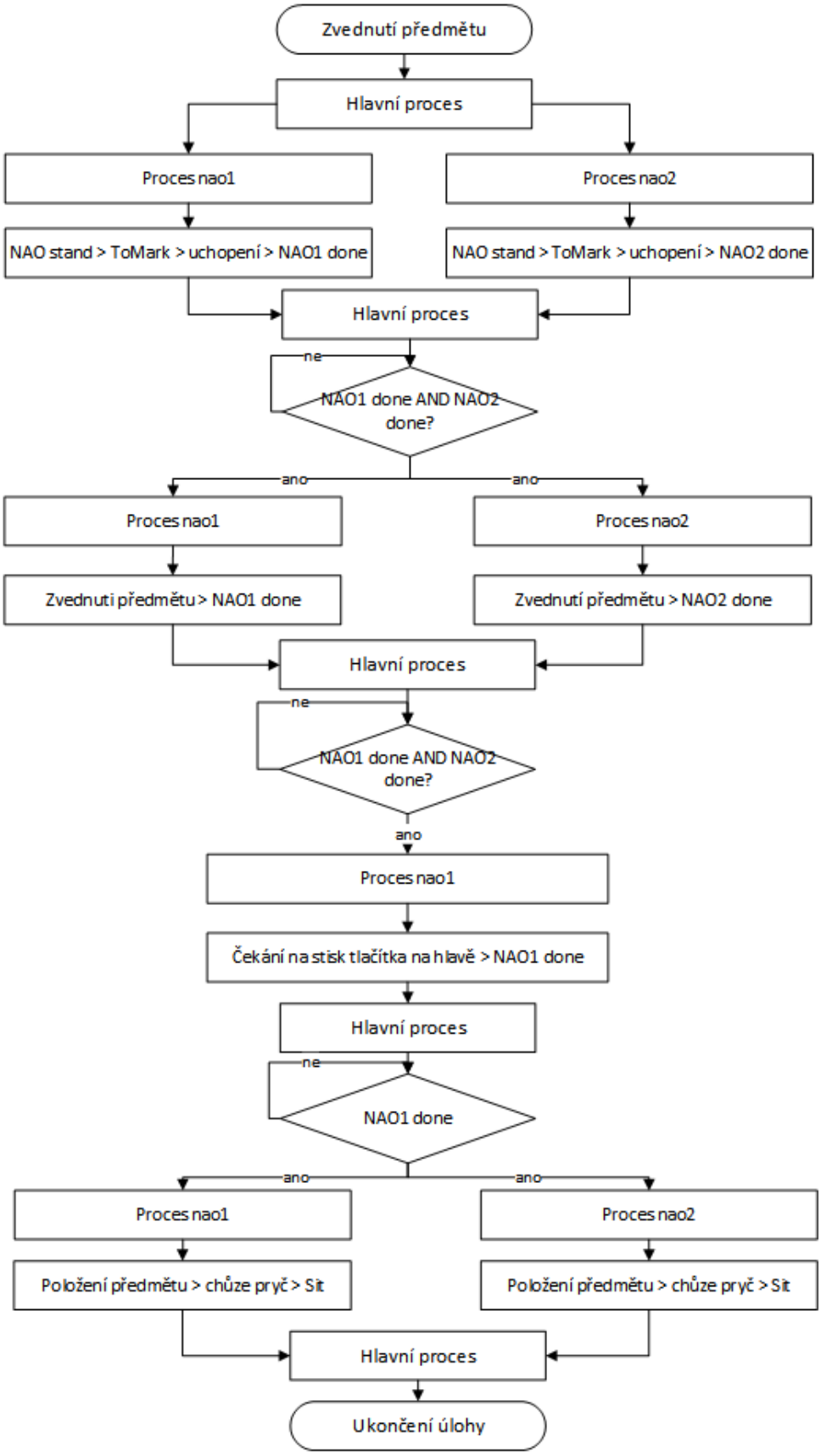



Obrázek 4.2: NAO Marky použité k orientaci v prostoru [1]

Po zapnutí testovací aplikace na počítači, se nejprve musí vyplnit IP adresy robotů a čísla NAO Marků, které mají roboti vyhledávat v prostoru. NAO Marky jsou umístěny na objektu, který budou roboti společně zvedat. Následně si v aplikaci stačí zvolit, aby roboti prováděli úlohu Pick a spustit úlohu tlačítkem start.

### 4.1.1 Popis kooperace

V hlavním procesu se vytvoří dva další procesy, pro každého robota jeden. V procesech se spustí funkce pro uvedení robota do stojící pozice StandInit. Následně je spuštěna funkce pro chůzi za NAO Markem viz Funkce pro chůzi k NAO Makru, ve které se nejprve implementuje funkce pro vyhledávání NAO Marku v prostoru viz Funkce pro hledání NAO Marku. Poté, co je NAO Mark nalezen a robot k němu přijde, je v každém procesu robota spuštěna funkce pro uchopení objektu. Robot nejprve roztáhne ruce, dřepne si a pomalu uchopí objekt svými pažemi. Tento pohyb pro uchopení objektu je pevně dán. Počítá se s tím, že robot přijde k objektu vždy stejně natočen a ve stejné vzdálenosti. Po uchopení objektu pošle proces, který robota ovládá, informaci hlavnímu procesu, že je robot připraven objekt zvednout. Po odeslání informace se proces, který ovládá robota, ukončí. Hlavní proces čeká na potvrzení od obou robotů. Jakmile jsou oba roboti připraveni zvednout předmět, jsou opět paralelně spuštěny procesy pro ovládání robotů a v nich je spuštěna funkce pro zvednutí objektu. Oba roboti společně zvednou předmět a pošlou informaci hlavnímu procesu o úspěšném zvednutí. Hlavní proces počká na zprávy od obou robotů a následně spustí proces, který ovládá pouze jednoho robota označeného jako NAO1. Robot NAO1 nyní čeká, na interakci člověka. Konkrétně na stisk prvního tlačítka na hlavě robota. Jakmile je tlačítko stisknuto, je poslána zpráva do hlavního procesu. Dále jsou opět paralelně spuštěny dva procesy pro oba roboty a v nich funkce pro společné položení předmětu. Po položení předmětu se spustí funkce pro chůzi pryč od objektu. Poslední funkce, která se spouští, zajišťuje, že si roboti sednou. Hlavnímu procesu jsou poslány informace, že jsou roboti v pozici sedu. Procesy, jenž ovládaly roboty, se ukončí, zároveň se ukončí i úloha Pick. Aplikace na počítači ale stále běží a je možné úlohu opět spustit nebo si zvolit jinou.




Obrázek 4.3: Vývojový diagram úlohy zvednutí objektu

### 4.1.2 Funkce pro hledání NAO Marku

Vstupní parametry funkce pro detekci NAO Marku jsou IP adresa robota, port, na kterém bude probíhat komunikace a číslo NAO Marku, který má robot vyhledávat. Funkce nejprve nadefinuje spojení s moduly ALLandMarkDetection, ALMotion, ALMemory a ALVideoDevice. Pomocí prvního z nich lze detekovat NAO Mark. Pomocí ALMotion modulu lze ovládat pohyby robota. Modul ALMemory slouží pro práci s pamětí robota, umožňuje uživateli dostat se k datům uloženým v paměti. Modul ALVideoDetection zastřešuje práci s kamerami robota.

Nejprve je pomocí modulu ALVideoDetection vybrána horní kamera, jako primární kamera pro snímání okolí. Následně se spustí modul ALLandMarkDetection, který konstantně zapisuje informace o detekci NAO Marku do paměti robota. Perioda tohoto zápisu je v programu nastavena na 0,5 sekundy. Robot začíná hledat NAO Mark v okolí vždy stejně. Nejprve otočí hlavu o 90 stupňů doleva a poté otáčí hlavu směrem doprava, vždy po 30 stupních. Po každém otočení čeká 3 sekundy, aby měl čas na detekci NAO Marku a nedocházelo k chybám. Pokud není NAO Mark nalezen, robot se otočí o 180 stupňů a opět otáčí hlavu o 30 stupňů. Tímto způsobem robot prohledá celé své okolí. Horní kamera snímá téměř 61 stupňů do šířky a necelých 48 stupňů na výšku. Robot snímá dvakrát ten stejný prostor, jelikož se hlava otáčí právě o 30 stupňů. Tento přístup detekce je sice pomalejší, ale je sníženo riziko chyb při rozpoznávání NAO Marků. Pokud robot po dvou otočeních neboli prohledání 360 stupňů svého okolí, nerozpozná žádný NAO Mark, je funkce ukončena a vrací informaci o negativním výsledku hledání. Po každém otočení hlavy robota proběhne pokus získat data o NAO Marku z paměti. Pokud se v paměti žádná data nenachází, pokračuje se v hledání. Jestliže se data v paměti nachází, jsou z nich vyčteny informace o čísle NAO Marku a jeho velikosti. Číslo detekovaného NAO Marku se porovná s požadovaným číslem a dále se porovnává jeho velikost s předem definovanou velikostí, pro zjištění vzdálenosti robota od značky. Pokud jsou tyto podmínky splněny, vrací funkce informaci o úspěšném detekování NAO Marku.

### 4.1.3 Funkce pro chůzi k NAO Makru

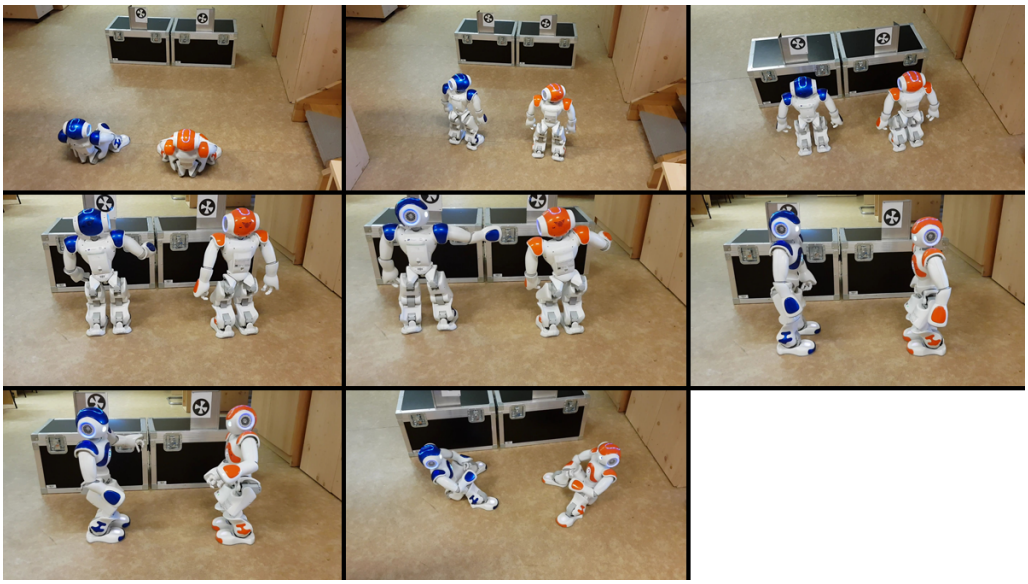
Vstupní parametry funkce pro chůzi k NAO Marku jsou IP adresa robota, port, na kterém bude probíhat komunikace, číslo NAO Marku, za kterým má robot jít a velikost NAO Marku, respektive vzdálenost, na kterou má robot ke značce přijít.

Nejprve se zavolá funkce pro hledání NAO Marku. Pokud je značka úspěšně detekována, tak robot natočí tělo stejným směrem jako je aktuálně natočena jeho hlava. Funkce získává data o detekovaném NAO Marku stejně jako funkce pro hledání viz Funkce pro hledání NAO Marku. Pokud je číslo detekovaného NAO Marku rovno požadovanému a robot ještě nedosáhl žádané vzdálenosti od značky, je robotovi poslán příkaz k chůzi. Robot jde dopředu a pro vycentrování se značkou se během chůze natáčí o úhel, pod kterým snímá NAO Mark. Robot ujde pouze určitou vzdálenost, která závisí na jeho vzdálenosti od značky, čím blíž se nachází, tím je jeho chůze kratší. Tímto způsobem dojde robot k NAO Marku na požadovanou vzdálenost a vrátí informaci o úspěšném dokončení chůze. Pokud robot během cesty k NAO Marku značku ztratí, pokusí se ji jednou znovu detekovat pomocí metody hledání NAO Marku. Pokud je detekce úspěšná, pokračuje v chůzi ke značce. V opačném případě funkce vrátí informaci o ztrátě NAO Marku.

## 4.2 Kooperace - tanec

V této praktické ukázce kooperace mají roboti NAO za úkol najít NAO Mark v okolí, přijít k němu, společně si zatancovat a poté si sednout.

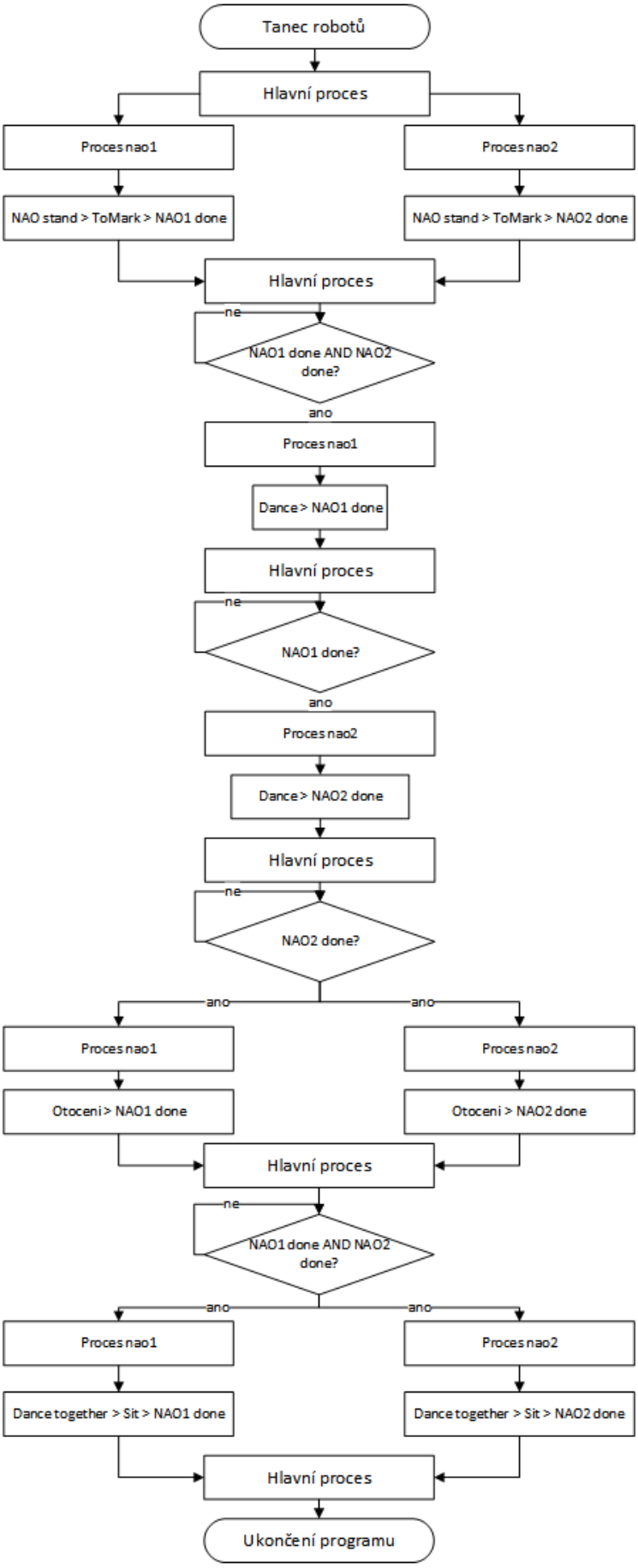
Po zapnutí testovací aplikace na počítači, se nejprve musí vyplnit IP adresy robotů a čísla NAO Marků, které mají roboti vyhledávat v okolí. NAO Marky jsou umístěny v prostoru vedle sebe tak, aby k nim roboti mohli přijít a stáli vedle sebe v jedné přímce. Pozice robotů před značkami slouží jako výchozí pozice k tanci. Následně si v aplikaci stačí zvolit, aby roboti vykonali úlohu Dance a spustit úlohu tlačítkem start.



Obrázek 4.4: Kooperace - tanec robotů

### 4.2.1 Popis kooperace

V hlavním procesu se vytvoří dva procesy, pro každého robota jeden. Procesy se spustí a provede se v nich funkce pro postavení robota do výchozí pozice StandInit. Následně je spuštěna funkce ToMark pro chůzi za NAO Markem. viz Funkce pro chůzi k NAO Makru. Poté co je značka nalezena a robot k ní přijde, odešle hlavnímu procesu informaci, že je na místě. Po odeslání informace se proces, který ovládá robota, ukončí. Hlavní proces čeká na zprávy od obou robotů. Jakmile oba roboti nahlásí, že jsou připraveni na místě, vytvoří se proces pro jednoho robota, ve kterém se spustí funkce tance. Robot zatancuje a poté pošle zprávu hlavnímu procesu. Nyní se vytvoří proces pro druhého robota, který také zatancuje a následně pošle informaci hlavnímu procesu o dokončení činnosti. Poté, co zatancuje i druhý robot, jsou paralelně spuštěny dva procesy, ve kterých běží funkce pro otočení robota o 90 stupňů tak, aby proti sobě roboti stáli čelem. Hlavní proces čeká na oba roboty. Dále jsou paralelně spuštěny dva procesy, pro každého robota jeden, ve kterých běží funkce pro společný tanec robotů. Po dokončení tance si roboti sednou a pošlou hlavnímu procesu zprávu o dokončení úkolu. Procesy, které ovládaly roboty, se ukončí, zároveň se ukončí i úloha Dance. Aplikace na počítači ale stále běží a je možné úlohu opět spustit nebo si zvolit jinou.






Obrázek 4.5: Vývojový diagram úlohy tance

## 4.3 Grafické rozhraní testovací aplikace

### 4.3.1 Realizace GUI

Pro programovací jazyk Python není dostupný žádný funkční grafický editor pro návrh grafického uživatelského rozhraní, jako má k dispozici například jazyk C# nebo Java. Přidávání grafických ovládacích prvků, jejich rozložení v GUI a editace jejich vlastností je prováděna psaním a úpravou zdrojového kódu, což je z hlediska časové náročnosti neefektivní. Proto, aby měl programátor představu, jak jsou ovládací prvky rozmístěny, musí aplikaci spustit.

Pro tvorbu grafického rozhraní byla použita knihovna Tkinter [12]. Tato knihovna je standardním prostředkem pro realizaci grafického rozhraní v jazyce Python a je součástí jeho základní instalace. Mezi alternativy pro tvorbu grafického rozhraní patří například knihovna Qt, Kivy či PyGUI. Tyto alternativy je potřeba doinstalovat a nemusí být tak stabilní jako knihovna Tkinter.

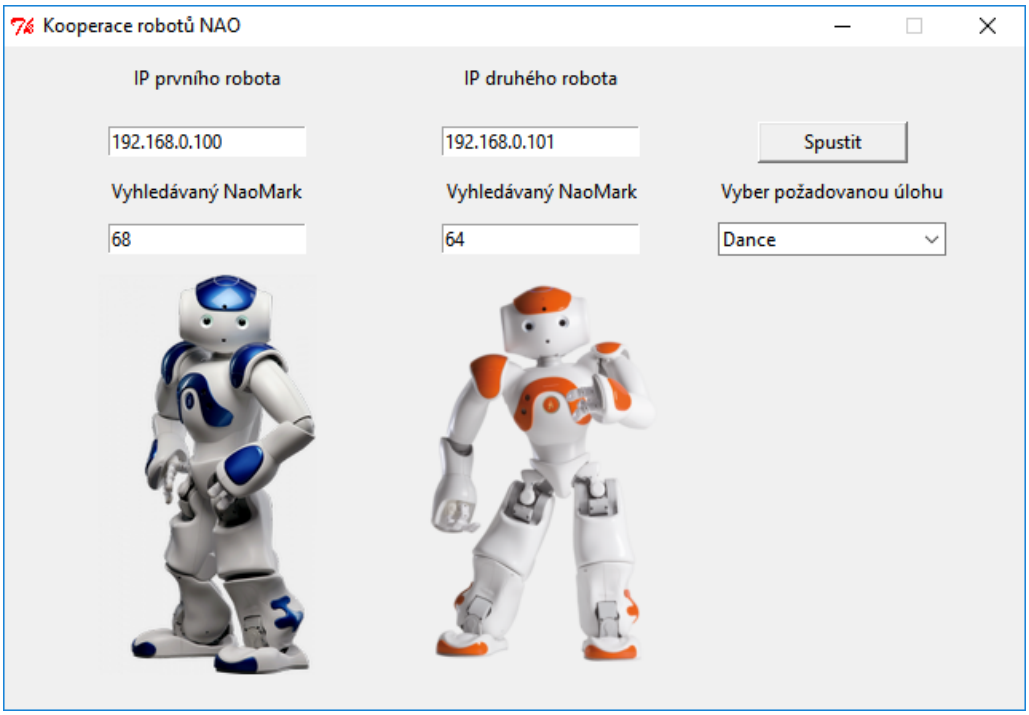
Z důvodu vložení obrázků do grafického rozhraní byla použita knihovna PIL (Python Imaging Library). Tato knihovna podporuje otevírání, manipulaci a ukládání obrázků různých formátů. V aplikaci je využívána pro zmenšení obrázků na požadovanou velikost určenou v pixelech.

### 4.3.2 Popis GUI

Pro zadání IP adres robotů slouží dvě textová pole. Další dvě textová pole slouží pro zadání čísel NAO Marků, které mají roboti hledat v prostoru. Dále se v okně nachází combobox, pomocí kterého si lze vybrat jednu ze tří úloh kooperace. Možnosti kooperace, které již byly popsány výše, jsou Pick, Dance a K NaoMarku. Pro spuštění vybrané úlohy slouží tlačítko Spustit.

Při stisku tlačítka se nejprve zavolá funkce, která kontroluje platnost zadaných IP adres a čísel NAO Marků. Funkce kontroluje, zda jsou všechny údaje vyplněny, správný formát IP adres a to zda IP adresa a čísla NAO Marku obsahují pouze číslice. V případě špatně zadaných údajů je uživateli zobrazena chybová hláška s konkrétním

problémem ve stavovém řádku aplikace, který se nachází v pravé horní části GUI. Pokud jsou zadané údaje vyhodnoceny jako správné, dojde ke spuštění hlavního procesu, ve kterém se spustí vybraná úloha z comboboxu.




Obrázek 4.6: GUI testovací aplikace

# 5 Shrnutí výsledků a porovnání s obdobnými pracemi

## 5.1 Shrnutí výsledků

Povedlo se navrhnout softwarové řešení kooperace robotů, které bylo použito v testovací aplikaci, kde bylo uplatněno ve dvou ukázkách spolupráce dvou humanoidních robotů NAO. První ukázka se zabývá společným zvednutím objektu. Druhá ukázka je společný tanec robotů. V obou úkolech se využívá NAO Marků, k orientaci v prostoru, či identifikaci, kde se nachází objekt.

Softwarové řešení je navrženo v univerzálním programovacím jazyce Python, tudíž lze využít na více druhů robotů. Software lze jednoduše upravit pro jinou spolupráci nebo lze do kooperace přidat více robotů. Pro vytvoření kooperace je potřeba určit si pevný scénář, podle kterého se budou roboti chovat, rozdělit si jej na dílčí části a tyto úlohy pak postupně posílat do řídících procesů.

Řešení je omezeno počtem procesů, které můžou na řídícím počítači běžet, což se odvíjí od hardwarového výkonu počítače.

## 5.2 Srovnání výsledků s obdobnými pracemi

Pokud srovnáme výsledky mé práce se studií viz Manipulace objektu spolu s mapováním okolí, tak v mé práci je řešena detekce objektu, příchod robotů k předmětu a uchopení objektu. Ani jeden z těchto problémů se v této studii neřeší. Naproti tomu je ve studii vyřešen pohyb robotů s objektem, který mají roboti v rukou pevně

uchycen lepicí páskou. Objekt není pevný, jedná se o desku papírového kartonu, která se může trochu ohýbat. Pohyb s předmětem je navíc doplněn mapováním okolí. To zajišťuje detekci překážek, kterým se roboti následně vyhnou. Pro mapování okolí mají roboti ve studii umístěny na hlavách speciální kamery.

Pokud srovnáme výsledky mé práce se studií viz Navigace k objektu a jeho manipulace, tak ve studii je řešena detekce objektu pomocí barvy, uchopení objektu a přenesení na určené místo. Detekce objektu pomocí jeho barvy nemusí být vždy přesná, pokud se v okolí nachází další objekt stejné, či podobné barvy. Chycení objektu je řešeno tak, že roboti společně předmět naberou z jedné strany v podstatě jako bagr, čili objekt podeberou pažemi a nadzdvihnou. S předmětem se pak pohybují stále po stejné přímce pouze dopředu. Jakmile dorazí na určené místo, předmět vyklopí z rukou a vrátí se na startovní pozice. Program je vyvíjen v Choregraphe a spouštěn v počítači, který s roboty, obdobně jako v mé práci, komunikuje přes bezdrátovou síť.

V mé práci není řešen pohyb robotů s objektem. Byly provedeny pokusy o pohyb robotů s objektem do strany. Ty ovšem narazily na několik problémů. Prvním problémem je špatná opakovatelnost robotů. K předmětu přijdou vždy v malinko jiné pozici a trošku jinak natočeni. Při společném pohybu do strany se tato odchylka projevuje každým krokem. Roboti jdou buď směrem od sebe a objekt jim spadne nebo jdou naopak příliš blízko k sobě a spadne jeden z robotů. Dalším problémem je, že roboti, které jsem měl k dispozici, jsou každý jiné verze a každý jinak starý a opotřebený. Díky všem těmto faktorům chodí každý z robotů jinak. Tato problematika je zmíněna v práci viz [8]. Tudíž je náročné roboty synchronizovat při přenášení objektu. Touto problematikou by se mohla zabývat další práce, která by navazovala na tuto bakalářskou práci.

# 6 Závěr

V této bakalářské práci byl navržen softwarový přístup pro kooperaci více robotů, který byl následně realizován v ukázkách spolupráce dvou humanoidních robotů NAO. Výsledkem je testovací aplikace spustitelná na počítači, ve které je možné si vybrat ze dvou ukázek kooperace robotů NAO a to zvednutí objektu nebo tance. Pro vývoj byl zvolen programovací jazyk Python v kombinaci s SDK od výrobce robotů. SDK umožňuje jednoduchým způsobem kontrolovat všechny funkčnosti robota. Základem softwarového řešení je využití více procesů a paralelismu. Pro tento účel je využita knihovna multiprocessing, která podporuje rozmnožování procesů a realizuje mezi nimi komunikaci. Grafické rozhraní testovací aplikace je navrženo pomocí knihovny Tkinter.

Základ navrženého softwarového přístupu je následující. Program neboli testovací aplikace, běží v hlavním procesu. V něm jsou vytvářeny další procesy, ve kterých jsou spouštěny funkce pro ovládání robota. Každý vytvořený proces ovládá jednoho robota jako vstupně-výstupní zařízení. Vytvořené procesy, které ovládají roboty, běží paralelně vedle sebe a komunikují s hlavním procesem. Tím je zajištěna synchronizace robotů a jejich vzájemná komunikace. Komunikace mezi počítačem a jednotlivými roboty probíhá pomocí bezdrátové sítě.

Navrhovaný software lze jednoduše upravit pro účely jiné spolupráce robotů. Jediné, co je potřeba, je mít pevně určený scénář toho, co mají roboti společně vykonat. Jednotlivé úkony napsat do funkcí v jazyce Python a následně tyto funkce volat ve vytvořených procesech pro jednotlivé roboty.

Výhodou softwarového řešení je jeho univerzálnost. Do kooperace lze přidat další roboty, a to pouhým přidáním dalšího procesu, který bude nového robota ovládat.

Software je naprogramován v univerzálním jazyce, tudíž není problém do kooperace zapojit i jiný druh robota než je NAO. Stačilo by naimportovat jeho SDK a napsat funkce pro jeho ovládání.

Vytvořený softwarový návrh pro kooperaci robotů lze použít jako základ pro další úlohy spolupráce robotů a to nejen NAO. Řešení by šlo rozšířit o další roboty nebo ukázky kooperace. Testovací aplikace se dá využít při prezentacích a ukázkách možností kooperace humanoidních robotů.

# Literatura


[1] *NAO Documentation — Aldebaran 2.1.4.13 documentation. SoftBank Robotics Documentation [online] [cit. 01.05.2018]. Dostupné z:* `http://doc.aldebaran.com/2-1/home_nao.html`

[2] Antoine Rioux, Claudia Esteves, Jean-Bernard Hayet, Wael Suleiman. Cooperative Vision-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment. International Journal of Humanoid Robotics, World Scientific Publishing, 2017, 14 (03), pp.1 - 30.

[3] A. N. Panfir et al., "NAO Robots Collaboration for Object Manipulation", Applied Mechanics and Materials, Vol. 332, pp. 218-223, 2013

[4] *Lu, Q., Hecker, J.P. & Moses, M.E. Auton Robot (2018) 42: 909. [cit. 04.05.2018] Dostupné z:* `https://doi.org/10.1007/s10514-017-9693-2`

[5] Torres-González A., Capitán J., Cunha R., Ollero A., Mademlis I. (2018) A Multidrone Approach for Autonomous Cinematography Planning. In: Ollero A., Sanfeliu A., Montano L., Lau N., Cardeira C. (eds) ROBOT 2017: Third Iberian Robotics Conference. ROBOT 2017. Advances in Intelligent Systems and Computing, vol 693. Springer, Cham

[6] *The Python Standard Library — Python 3.3.7 documentation. 302 Found [online]. Copyright © [cit. 04.05.2018]. Dostupné z:* `https://docs.python.org/3.3/library/index.html`

[7] Meng-Hung Wu, A. Konno, S. Ogawa, and S. Komizunai. Symmetry cooperative object transportation by multiple humanoid robots. In IEEE Int. Conf. on Robotics and Automation (ICRA), pages 3446{3451, May 2014.

[8] *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. [cit. 03.05.2018] Dostupné z:* `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7029995&isnumber=7029946`

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

[10] *Boston Dynamics. Boston Dynamics is changing your idea of what robots can do. | Boston Dynamics [online]. Copyright © 2018 Boston Dynamics [cit. 05.05.2018]. Dostupné z:* `http://www.bostondynamics.com`

[11] ZÁDA, Václav. Robotika: matematické aspekty analýzy a řízení. Vyd. 1. Liberec: Technická universita v Liberci, 2012. ISBN 978-80-7372-882-3.

[12] *An Introduction To Tkinter. effbot.org [online][cit. 02.05.2018]. Dostupné z:* `http://effbot.org/tkinterbook/tkinter-index.htm#introduction`

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

# A Obsah přiloženého DVD

- Bakalářská práce ve formátu .pdf
- Zdrojový kód testovací aplikace ve formátu .py
- Spustitelná verze testovací aplikace ve formátu .exe
- Vývojové diagramy a fotografie
- Video ukázky dvou kooperací - zvednutí objektu a tanec