



FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

**Multimodální rotace robota s detekcí
klíčového slova**

Yauheni Varabyou



FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Multimodální rotace robota s detekcí klíčového slova

Yauheni Varabyou

Vedoucí práce

Ing. Martin Bulín, M.Sc.

PLZEŇ

2024

© Yauheni Varabyou, 2024.

Všechna práva vyhrazena. Žádná část tohoto dokumentu nesmí být reprodukována ani rozšiřována jakoukoli formou, elektronicky či mechanicky, fotokopírováním, nahráváním nebo jiným způsobem, nebo uložena v systému pro ukládání a vyhledávání informací bez písemného souhlasu držitelů autorských práv.

Citace v seznamu literatury:

VARABYOU, Yauheni. *Multimodální rotace robota s detekcí klíčového slova*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Martin Bulín, M.Sc.

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:

Osobní číslo:

Studijní program: **B3902 Inženýrská informatika**

Studijní obor: **Informatika**

Téma práce:

Zadávající katedra: **Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se se stávajícími výhody. ; zdokumentujte jejich výhody a ne-
2. Seznamte se se stávající podobou
3. Na základě bodů 1 a 2 navrhněte

4. Navržený implementujte a zabezpečte z bodu
2.
5. Výsledné řešení důkladně otestujte na netriviální množině dat
6. Vytvořte komunitně srozumitelný a veřejně dostupný tutoriál.
7. Zhodnoťte dosažené výsledky.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **24. srpna 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 22. ledna 2024

.....
Yauheni Varabyou

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

(i)

Abstrakt

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui.

Klíčová slova

Python • RL • Respeaker

Poděkování

Na tomto místě bych rád poděkoval svému předchůdci v roli „local wizarda“, člověku, který připravil první prakticky použitelnou L^AT_EXovou šablonu kvalifikační práce na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni, panu Ing. Petru Lobazovi, Ph.D. Asi ještě větší a srdceňejší poděkování si zaslouží úžasný člověk, který mě s T_EXem před téměř třiceti lety seznámil: pan Doc. RNDr. František Ježek, CSc.

Rád bych na tomto místě poděkoval i dalšímu matematikovi, Ing. Janu Pospíšilovi, Ph.D., který mne neúnavně přesvědčoval, že FAV novou šablonu kvalifikačních prací potřebuje jako sůl a následně mě nutil na ní pracovat (a samozřejmě přispěl řadou podnětných připomínek).

Nemenší díky si zaslouží mí drazí kolegové z Katedry informatiky a výpočetní techniky, kteří svojí konstruktivní kritikou významně přispěli ke konečné podobě šablony – jmenovitě jsou to (v abecedním pořadí): Doc. Ing. Přemysl Brada, M.Sc., Ph.D., Doc. Ing. Pavel Herout, Ph.D., Ing. František Pártl a Ing. Petr Vaněček, Ph.D.

V neposlední řadě si poděkování zaslouží také studenti, kteří nově vzniklou šablonu testovali: Bc. Veronika Báčová a Petra Ocelíková.

Obsah

1 Hardware	3
1.1 Raspberry Pi 4 model B	3
1.2 ReSpeaker 4-Mic Array	3
1.3 Krokový motor	5
1.4 Světelný senzor	5
1.5 Solární panel	6
2 Technologie	7
2.1 Framework ODAS	7
2.1.1 Lokalizace zdroje zvuku	9
2.2 Platforma Picovoice	9
2.2.1 Porcupine	9
2.3 Framework ROS	11
2.4 Tkinter	12
2.5 Reinforcement Learning	13
2.5.1 Q-learning algoritmus	14
3 Vlastní metody a jejich aplikace	17
3.1 Robotická entita	17
3.2 Detekce polohy řečníka v prostoru	18
3.2.1 Filtrace vstupních zvukových signálů	19
3.2.2 Indikace směru zvuku v reálném čase	20
3.3 Grafická vizualizace a modelování chování cílového zařízení	22
3.3.1 Přehled aplikace a popis funkcionality	22
3.4 Implementace technologie KWS	26
Bibliografie	27
Seznam obrázků	29

Hardware

1

Jedním z úkolů práce bylo ukázat, že je možné pomocí veřejně dostupných technologií vytvořit efektivní zařízení. Pro tento a další účely byl v práci testován a použit následující hardware:

1. Raspberry Pi 4 model B
2. ReSpeaker 4-Mic Array pole čtyř mikrofonů
3. Krovkový motor NEMA17 (47 mm) model 42HD6021-03
4. Světelné senzory TSL2591 v počtu 4 kusů
5. Solární panel

1.1 Raspberry Pi 4 model B

Raspberry Pi 4B¹ je nízkonákladový a malý jednodeskový počítač s dostatečným výkonem pro multifunkční úlohy. Pro naši úlohu byl zvolen operační systém Debian Bullseye² (Desktop, 64-bit). V testovací verzi a v práci byl použit model s novějším typem operační paměti LPDDR4 a 2 GB RAM. Zařízení je vybaveno čtyřjádrovým procesorem ARM Cortex-A72 s taktem 1,5 GHz. Grafika VideoCore (verze VI). Což dohromady zajišťuje dobrý výkon a vysokou rychlosť počítače.

1.2 ReSpeaker 4-Mic Array

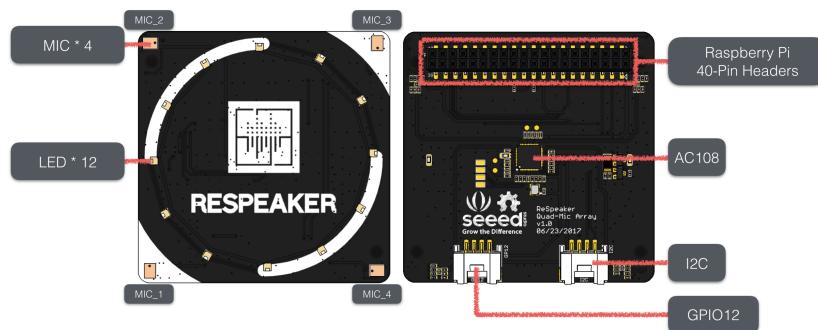
Tato součástka od výrobce Seeed Studio³ je vybavena čtyřmi analogovými mikrofony a audio kodekem AC108 pro snímání a zpracování hlasu ve vysokém rozlišení.

¹<https://rpishop.cz/raspberry-pi-4/1599-raspberry-pi-4-model-b-2gb-ram-765756931175.html>

²<https://wiki.debian.org/DebianBullseye>

³https://wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/

1. Hardware



Obrázek 1.1: Přehled mikrofonu ReSpeaker 4-Mic Array použitého v práci [1]

Zařízení je určeno k použití jako hlasové uživatelské rozhraní primárně v kombinaci s Raspberry Pi. Tato verze mikrofonu je navíc vybavena LED kroužkem s 12 programovatelnými LED diodami APA102. Díky těmto čtyřem mikrofonům a LED kroužku má ReSpeaker 4-Mic Array schopnost detektovat a indikovat hlasovou aktivity (VAD - Voice Activity Detection), odhadovat směr zvuku (DOA - Direction Of Arrival) a provádět detekci klíčových slov (KWS - Keyword Spotting) široce používaných v aplikacích hlasové interakce.

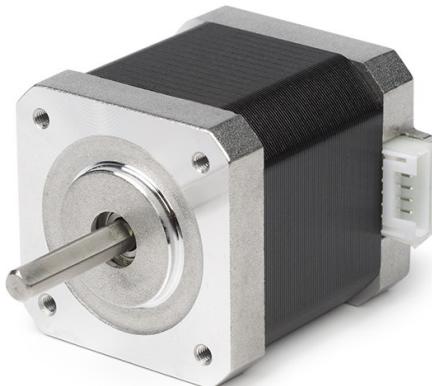


Obrázek 1.2: ReSpeaker 4-Mic Array mikrofon propojený s Raspberry Pi 4B [1]

- *Voice Activity Detection (VAD)* – detekce přítomnosti nebo nepřítomnosti lidské řeči ve vstupním akustickém signálu.
- *Direction Of Arrival (DOA)* – směr, ze kterého se obvykle šíří zvukové vlny.
- *Keyword Spotting (KWS)* – proces rozpoznávání předem definovaných slov z řečového signálu.

1.3 Krokový motor

Pro zajištění rotačního pohybu robota byl vybrán krokový motor od výrobce NEMA17⁴ (konkrétně model 42HD6021-03). Tento malý dvoufázový krokový motor má na svůj rozměr (47 mm) dostatečně velký statický moment (500 mN.m) a vysokou přesnost - 200 kroků na otáčku, což je 1,8 stupně. Motor se jednoduše instaluje a demontuje bez použití speciálních náradí. Často se používá pro 3D tiskárny. V bakalářské práci byl použit k rotaci horní části cílového zařízení.



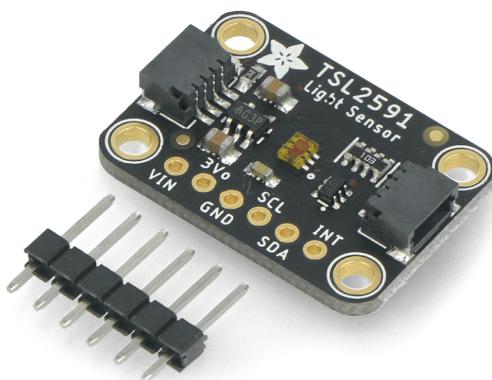
Obrázek 1.3: Krokový motor NEMA17 [2]

1.4 Světelný senzor

Pro určení místa s nejvyšším dopadem slunečního světla v práci byly použity čtyři světelné senzory TSL2591⁵ od výrobce Adafruit. Světelný senzor TSL2591 má vysoký dynamický rozsah (600M:1) a velkou citlivost s účinným maximem do 88.000 Lx. Obsahuje infračervené a celospektrální diody, díky čemu umožnuje měřit odděleně různé typy světla. Sensor lze snadno spojit s počítačem Raspberry Pi přes I2C rozhraní.

⁴<https://novo3d.in/product/nema17-stepper/>

⁵<https://rpishop.cz/adafruit/269-svetelny-senzor-tsl2591.html>



Obrázek 1.4: Světelný senzor TSL2591 [3]

1.5 Solární panel

Polykrystalický solární panel ve velikosti 110 x 60 x 2,5 mm⁶. S výstupním napětím 6 V a s výstupním proudem do 200 mA. Solární panel byl použit k zajištění dodatečného nabíjení cílového zařízení.

⁶<https://dratek.cz/arduino/1589-solarni-panel-6v-1w-az-200ma.html>

Technologie

2

V této části budou podrobně popsány technologie a způsoby jejich aplikace, použité v bakalářské práci. Technologie byly zvoleny na základě poměru jejich snadné integrace a efektivity jejich použití. Některé z technologií byly otestovány a dále využity prostřednictvím realizovaných demo verzí, některé z nich posloužily jako podklad pro vypracování této práce. Základní technologie a nástroje:

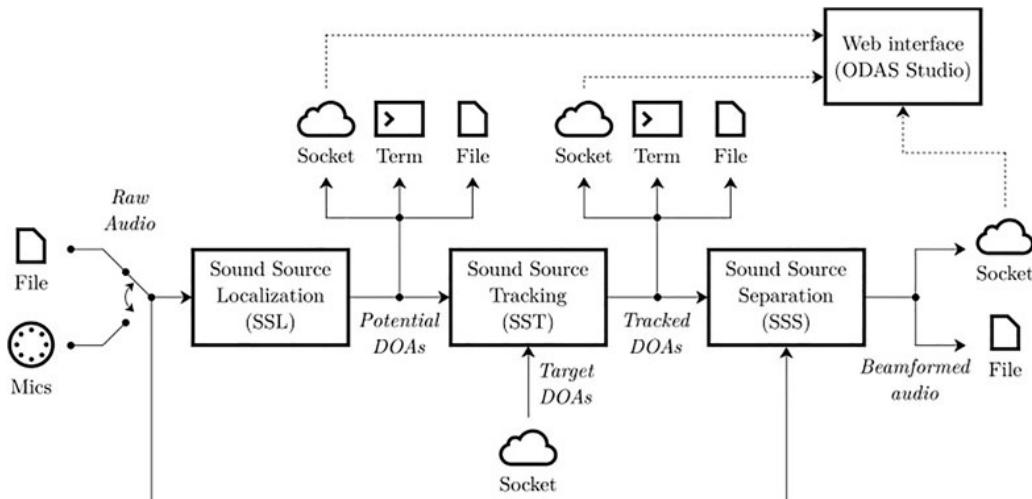
1. ODAS (Open embeddeD Audition System)
2. Platforma Picovoice
3. Porcupine
4. ROS (Robot Operating System)
5. Tkinter

Vše bylo implementováno v programovacím jazyce Python, byl použit operační systém Debian. Pro testování a modelování byl použit také operační systém Windows.

2.1 Framework ODAS

Open embeddeD Audition System (ODAS) je open source knihovna určená k lokalizaci, sledování, separaci a následnému filtrování zdrojů zvuku. ODAS je framework vyvinutý pro použití na nízkonákladovém a nízkoúčinném vestavěném hardwaru. Je napsaný výhradně v programovacím jazyce C, díky čemu má lepší optimalizaci a je lépe přenositelný na různá vývojová prostředí [4]. Maximální přenositelnost frameworku je dosažena tím, že má jenom jednu externí závislost na známé knihovně třetí strany FFTW3, která se používá k efektivnímu výpočtu rychlých Fourierových transformací [5]. V rámci práce ODAS byl použit k lokalizaci zdroje zvuku a jeho postfiltraci.

Na obr. 2.1 je znázorněn zvukový kanál pro zpracování zvuku a vstupně-výstupní rozhraní, z nichž každé běží v samostatném vlákně, aby bylo možné plně využít možností vícejádrových procesorů. Zdrojový zvuk může být reprezentován předem nahraným vícekanálovým audio souborem RAW nebo může být získán přímo z připojeného mikrofonu pro zpracování v reálném čase. Modul lokalizace zdroje zvuku (SSL - Sound Source Localization) generuje pevný počet potenciálních zdrojů zvuku, které se předávají modulu sledování zdroje zvuku (SST - Sound Source Tracking). Modul SST identifikuje sledované zdroje a přenáší tyto DOA (Direction Of Arrival - směr zvukové vlny) do modulu provádějícího separaci zdrojů zvuku (SSS - Sound Source Separation) k provedení formování paprsku pro každý cílový zdroj zvuku. Uživatel může rovnou definovat cílové DOA, pokud je směr zdroje zvuku předem známý a není vyžadována lokalizace ani sledování. Vygenerované segmenty lze zapsat do audio souborů RAW nebo také odeslat prostřednictvím socketu [5].



Obrázek 2.1: ODAS schéma zpracování příchozího zvuku [5]

- *Sound Source Localization (SSL)* – schopnost určit polohu nebo zdroj detekovaného zvuku podle směru a vzdálenosti.
- *Sound Source Tracking (SST)* – schopnost sledovat měnící se zdroj zvuku v reálném čase.
- *Sound Source Separation (SSS)* – schopnost oddělit aktivní řeč nebo určitý signál od šumu na pozadí nebo ticha.

2.1.1 Lokalizace zdroje zvuku

Lokalizace zvuku ve framework se opírá o metodu Generalized Cross-Correlation with Phase Transform (GCC-PHAT), která se vypočítá pro každý pár mikrofonů. ODAS používá Inverse Fast Fourier Transform (IFFT) algoritmus k efektivnímu výpočtu křížové korelace ze signálů ve frekvenční oblasti. Při práci s malými nebo krátkými poli signálů, ODAS může také interpolovat křížovou korelací, aby se zlepšila přesnost lokalizace a eliminoval artefakt diskretizace TDOA, který vzniká při Fourierově transformaci. Hlavním principem výpočtu DOA je metoda Steered-Response Power with Phase Transform (SRP-PHAT). Pro každý DOA framework vypočítá SRP-PHAT součtem hodnoty křížové korelace spojené s odpovídajícím časovým rozdílem (TDOA), získaným pomocí GCC-PHAT pro každý pár mikrofonů a vrátí hodnotu DOA s nejvyšším výkonem.

2.2 Platforma Picovoice

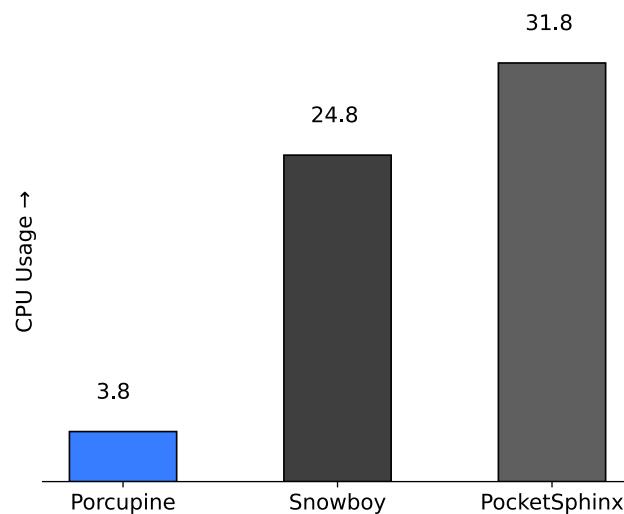
Pro přesné zpracování zvukových dat a zároveň pro rychlé, ale efektivní trénování zvukových konceptů byla zvolená platforma Picovoice¹. Picovoice je multiplatformní vývojářský software určený k vytváření hlasových produktů. Ve srovnání s populárními produkty pro rozpoznávání a zpracování řeči od velkých společností (např. Google, Amazon), které k analýze a práci s hlasem využívají Cloud, Picovoice se zaměřuje výhradně na on-device procesy [6]. Tato metoda umožňuje dosáhnout přesnějších výsledků a pracovat bez připojení k internetu. Platforma Picovoice má webový systém Picovoice Console, pomocí kterého je možné rychle natrénovat a otestovat hlasové koncepty, pro jejich další využití v KWS (Keyword Spotting - detekce klíčového slova), VAD (Voice Activity Detection - detekce hlasu) a dalších systémech.

2.2.1 Porcupine

Porcupine² je vysoce přesný a procesově nenáročný nástroj pro detekci klíčových slov, který rozpoznává jedinečné signály pro přechod softwaru z pasivního stavu na aktivní. Porcupine je jeden z nástrojů platformy Picovoice, který umožňuje vytvářet aplikace s možností neustálého poslechu hlasových signálů. Jedním z cílů bakalářské práce bylo zprovoznění potřebných technologií na low-cost zařízení, což Porcupine umožňuje díky své procesní nenáročnosti (viz Obrázek 2.2) a tomu, že je multiplatformní. Základní překážkou implementace mechanismů typu KWS je jejich závislost na masivním sběru dat pro trénování každého nového modelu. Tento nástroj společně s Picovoice má webové rozhraní, kde je možné snadno a

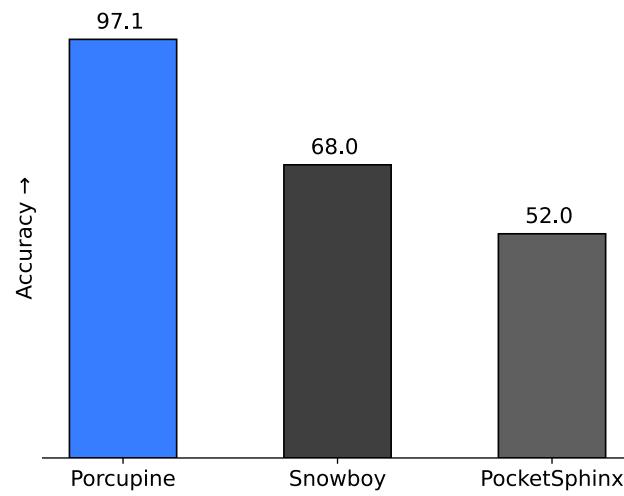
¹<https://picovoice.ai/docs/>

²<https://picovoice.ai/platform/porcupine/>



Obrázek 2.2: Využití CPU (single-core) Porcupine ve srovnání s jinými populárními nástroji [7, 8] pro detekci klíčového slova [9]

rychle natrénovat a otestovat vlastní klíčové slovo a z toho vytvořit vlastní model pro použití v dalším rozpoznávání. Takovým způsobem Porcupine eliminuje potřebu sběru dat pro každý nový model. Dalším důvodem pro výběr tohoto nástroje je jeho přesnost. Obrázek 2.3 zobrazuje přesnost nástrojů, když frekvence falešných poplachů je jednou za 10 hodin s šumem (10 dB SNR) a řečí na pozadí [9]:



Obrázek 2.3: Porovnání přesnosti detekce klíčového slova Porcupine s jinými nástroji [7, 8, 9]

- *Keyword (Wake-Word, Hot-Word, Keyphrase atd.)* – specifické slovo nebo fráze, která po vyslovení může aktivovat neaktivní zařízení, například: "Hey Siri", "OK Google" a tak dále.

2.3 Framework ROS

Robot Operating System³ (ROS) je open source framework spravovaný společností Open Robotics se sadou softwarových knihoven a nástrojů pro vytváření robotických aplikací. ROS poskytuje hardwarovou abstrakci, nízkoúrovňové ovládání zařízení, předávání zpráv mezi procesy, nabízí ovladače zařízení, knihovny, vizualizéry, správu balíčků a další funkce. Tento framework implementuje několik různých stylů komunikace, včetně RPC synchronní komunikace realizovatelnou přes služby (Services), asynchronního streamování dat přes téma (Topics) a ukládání dat na serveru (Parameter Server) používaného uzly (Nodes) za procesním běhu [10]. Mezi základní prvky ROS architektury patří:

- *Package (Balík)* – hlavní jednotka pro organizaci softwaru v ROS. Slouží jako strukturovaný kontejner pro organizaci a správu softwarových komponent nezbytných pro efektivní strukturování, sdílení kódu, dat a zdrojů v rámci ekosystému ROS [11].
- *Node (Uzel)* – nezávislý modul, který v softwarové architektuře zařízení vykonává určitou funkci nebo úkol. Tyto moduly mohou vzájemně komunikovat, aby dosáhly celkové funkcionality. ROS systém se obvykle skládá z velkého množství uzlů [11].
- *Message (Zpráva)* – jednoduchá datová struktura sestávající z typizovaných polí obsahujících standardní primitivní typy (integer, float, boolean, atd.). Uzly komunikují mezi sebou pomocí zpráv [11].
- *Topic (Téma)* – speciální kanál, prostřednictvím čehož uzly mohou publikovat a přijímat zprávy. Je to prostředek pro výměnu dat mezi uzly, který jim umožňuje vzájemnou interakci a spolupráci v ROS systému [11].
- *Services (Služby)* – speciální typ komunikace využívající téma, který umožňuje uzlům provádět vzdálené operace a požadavky. Služby jsou založeny na modelu požadavek-odpověď. Jeden uzel odešle požadavek (request) na provedení určité operace a jiný uzel tento požadavek splní a odešle zpět odpověď (reply) [11].

³<https://www.ros.org/>

- *Actions (Akce)* – komunikační model pro provedení dlouhotrvajících vzdálených příkazů a požadavků, má podobný typ komunikace jako model *Services*. Tento typ komunikace umožňuje sledovat průběh požadavků, získat konečný výsledek a v případě potřeby požadavek zrušit před jeho dokončením. Pro tyto všechny účely slouží balíček *actionlib* [12].

Lze také zdůraznit hlavní přednosti ROS systému, které jsou popsány na oficiálních webových stránkách [10], jmenovitě:

- *Jazyková nezávislost*: framework ROS lze snadno implementovat do jakéhokoli moderního programovacího jazyka, například: Python, C++, Java atd.
- *Snadné testování*: ROS má vestavěný framework pro jednotkové/integrační testování s názvem *rostest*, který usnadňuje vytváření a odstraňování testovacích přípravků.
- *Škálovatelnost*: ROS je vhodný pro velké systémy a velké vývojové procesy.

2.4 Tkinter

V určitých situacích může být velmi obtížné stanovit správnost výpočtů nebo popsat princip fungování zařízení pomocí pouze čistých dat. Aby bylo snazší identifikovat chyby nebo vysvětlit některé z jevů, vytváří se vizuální model, kopírující skutečné chování určitého zařízení. Pro tyto účely v bakalářské práci byl zvolen Tkinter⁴ framework. Tkinter je vestavěný modul jazyka Python, který se používá k vytváření aplikací s grafickým uživatelským rozhraním (GUI). Framework je open-source vydaný pod Python licencí. Tento nástroj má jednoduchou strukturu vytváření vizuálních prvků a snadno se s ním pracuje, proto Tkinter je jeden z nejčastěji používaných modulů pro vytváření GUI aplikací v Pythonu. Rozhraní Tkinter frameworku je založeno na multiplatformní sadě nástrojů Tk⁵, která byla původně navržen pro Tool Command Language⁶ (Tcl) [13, 14].

- *Graphical User Interface (GUI)* – forma uživatelského rozhraní, která umožňuje uživatelům komunikovat s počítačem prostřednictvím vizuálních ukazatelů (ikony, okna, tlačítka atd.).

Widgety jsou hlavními prvky GUI aplikace ve frameworku. Pomocí widgetů, které jsou součástí sady nástrojů Tk, Tkinter poskytuje uživatelům jednoduchý způsob

⁴<https://docs.python.org/3/library/tkinter.html>

⁵[https://en.wikipedia.org/wiki/Tk_\(software\)](https://en.wikipedia.org/wiki/Tk_(software))

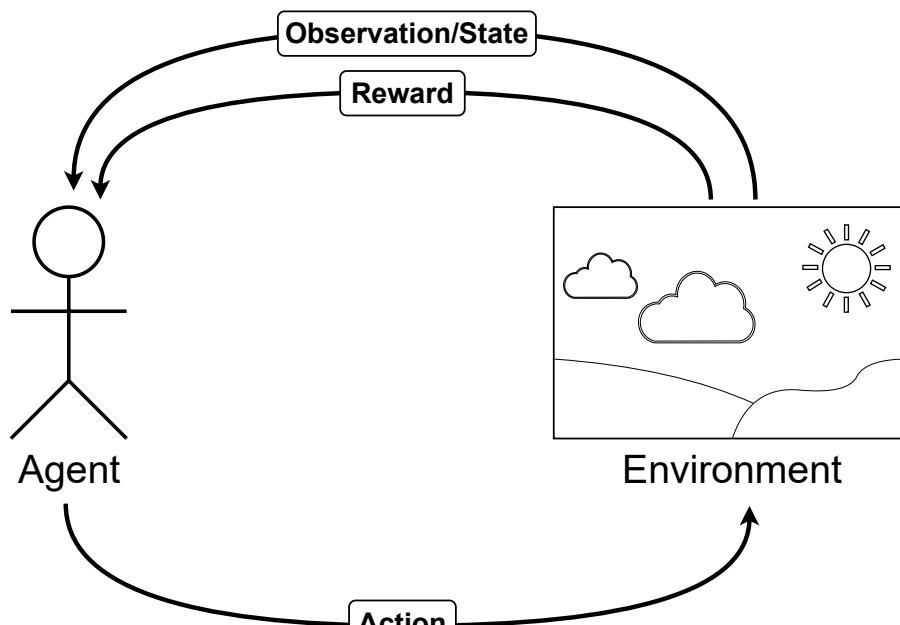
⁶<https://www.tcl.tk/>

vytváření elementů grafického uživatelského rozhraní. V aplikaci lze widgety Tk použít ke konstrukci tlačítek, textových boxů, datových polí atd. Po vytvoření tyto prvky lze propojit nebo interagovat s funkcemi, metodami, datovými objekty nebo dokonce s jinými widgety [13, 14].

2.5 Reinforcement Learning

Učení s posilováním (Reinforcement Learning nebo RL) je oblast vědy o rozhodování, která je v principu podobná strojovému učení. Jde o učení optimálního chování v prostředí s cílem maximalizovat odměnu na základě předem stanovených pravidel.

Učícímu se není řečeno, jaké akce má provádět, ale místo toho musí sám určit, které akce jsou nevýhodnější tím, že je vyzkoušel. Je třeba zmínit, že kvalita akcí se měří nejen okamžitou odměnou, kterou dostanou v daném momentě, ale také pozdějsí odměnou, kterou mohou získat v budoucnu. Tento proces objevování je podobný způsobu hledání metodou pokus-omyl. Tyto dvě charakteristiky, hledání metodou pokus-omyl a pozdější odměna, jsou dvěma nejdůležitějšími distinktivními znaky posilovacího učení [15].



Obrázek 2.4: RL

Úloha učení s posilováním zahrnuje agenta, který zkoumá neznámé prostředí, aby dosáhl cíle pomocí určitých aktivit. Agent se musí naučit vnímat stav prostředí a ovlivňovat ho pomocí svých akcí tak, aby maximalizoval svoji odměnu. Cílem algoritmu RL je najít takovou politiku akcí, která maximalizuje průměrnou hodnotu,

kterou může získat z každého stavu systému. Formální struktura pro RL je převzata z problému optimálního řízení Markovových rozhodovacích procesů (MDP) [15].

Hlavními prvky systému RL jsou [16]:

- *Agent* nebo učící se subjekt
- *Prostředí*, se kterým agent interaguje
- *Pravidla*, kterými se agent řídí při provádění akcí nebo při pozorování prostředí
- *Odměna*, kterou agent obdrží při provádění akcí

Učení s posilováním se od učení s učitelem liší tím, že nevyžaduje předložení označených dvojic vstupních a výstupních dat a nevyžaduje ani explicitní korekci neoptimálních akcí. Místo toho se zaměřuje na hledání rovnováhy mezi zkoumáním prostředí a vlastním rozhodováním [15]. V případě učení s posilováním nejsou k dispozici sady trénovacích dat, posilovací agent se musí sám učit ze svých zkušeností a rozhodnout, co má udělat pro splnění daného úkolu. Díky tomu, že algoritmus nepotřebuje sady trénovacích dat, byl zvolen pro využití v bakalářské práci.

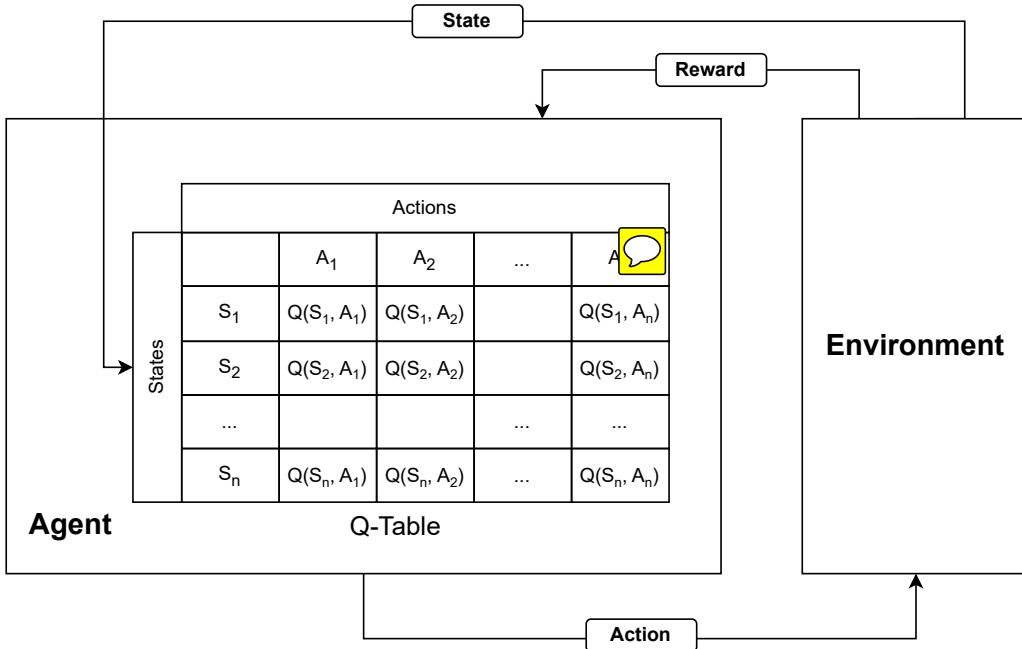


2.5.1 Q-learning algoritmus

Algoritmy RL lze obecně rozdělit na bezmodelové (model-free) a modelové (model-based) [15]. Metody založené na modelu se spoléhají především na *plánování*, zatímco metody bez modelu se spoléhají především na *učení*. V této práci je použit algoritmus Q-learning vycházející z bezmodelového (model-free) typu.

Důležité pojmy v učení pomocí Q-Learning (viz Obrázek 2.5) [17]:

- *Stav (S)*, představuje aktuální pozici agenta v prostředí
- *Akce (A)* je činnost, kterou agent provede, když se nachází v určitém stavu
- *Odměna* každou akci agent dostává kladnou nebo zápornou odměnu.
- *Epizody* je posloupnost stavů, akcí a odměn, která končí konečným stavem
- *Q-Value nebo Q (A, S)*, slouží k určení, jak dobrá je akce A provedená v určitém stavu S
- *Q-Table* je ~~jednoduchá~~ vyhledávací tabulka, ve které se pro každý stav počítá maximální očekávaná budoucí odměna za akci (Q-Value). Pomáhá nám najít nejlepší akci pro každý stav v prostředí



Obrázek 2.5: Q-learning

Q-learning je algoritmus, který nalezne nejlepší možnou strategii vzhledem k aktuálnímu stavu agenta. V závislosti na tom, kde se agent v prostředí nachází, rozhodne o další akci, kterou je třeba provést. Obecně, cílem modelu je najít nejlepší činnost vzhledem k jeho aktuálnímu stavu. Pro získání očekávaného budoucího stavu a odměny v každém stavu se používá Bellmanova rovnice. Jakmile pomocí této rovnice získáme výsledek, uložíme jej do Q-tabulky, abychom mohli v budoucnu jednotlivé stavy vzájemně porovnávat. Rovnice je uvedena níže:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (2.1)$$

$$\text{New } Q(S, A) = Q(S, A) + \alpha [R(S, A) + \gamma \max Q'(S', A') - Q(S, A)]$$

Current Q Value Learning Rate Reward
 ↓ ↓ ↓
 New Q(S, A) = Q(S, A) + α [R(S, A) + γ Max Q'(S', A') - Q(S, A)]

Discount Rate Maximum Expected Future Reward
 ↑ ↑

Obrázek 2.6: Bellmanova rovnice [17]

2. Technologie

K nalezení příštího stavu našeho agenta využívá aktuální stav $Q(S,A)$ a odměnu $R(S,A)$ spojenou s tímto stavem spolu s maximální očekávanou odměnou $\text{Max } Q(S,A)$ a diskontní sazbou γ , která určuje její hodnotu pro aktuální stav. Míra učení α určuje, jak rychle nebo pomalu se bude model učit [15, 17].



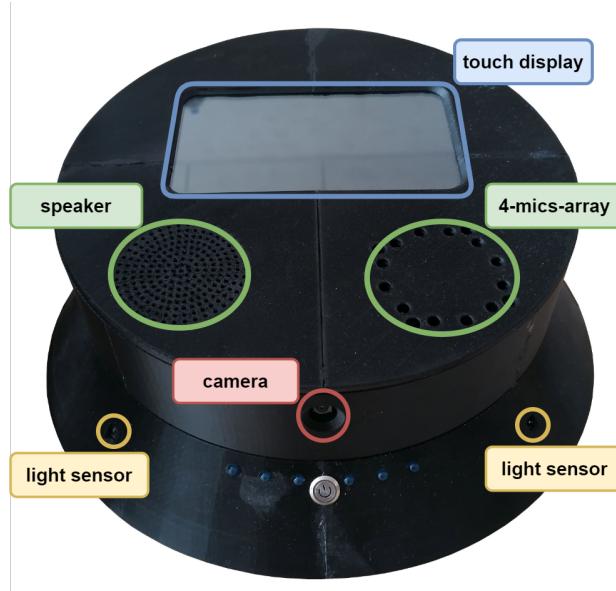
Vlastní metody a jejich aplikace

3

V této části bakalářské práce budou podrobně popsány metody použité k dosažení postavených cílů, příklady vlastních nápadů a vylepšení pro jejich zefektivnění, grafická simulace pro testování a znázornění výstupů metod a jejich konečná implementace na fyzickém zařízení.

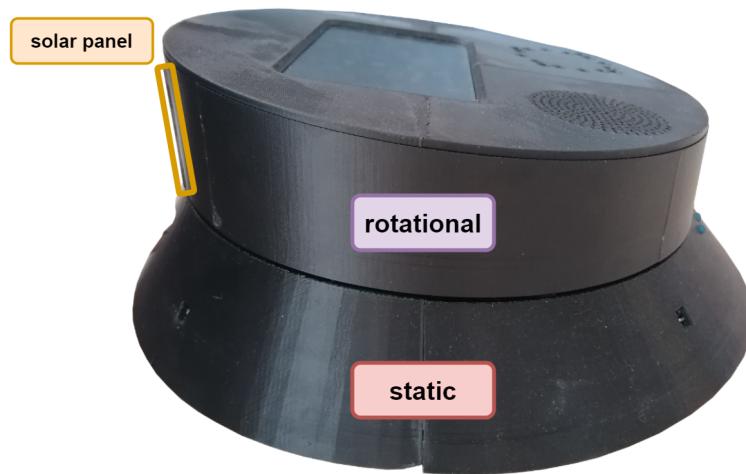
3.1 Robotická entita

Hlavní myšlenkou práce bylo to, aby všechny použité technologie a metody byly implementovány a vyhodnoceny na cílové robotické entitě v reálných podmínkách. K tomuto účelu na katedře kybernetiky Západočeské univerzity v Plzni ve spolupráci bylo vyvinuto unikátní fyzická zařízení s názvem *Robot.v1*.



Obrázek 3.1: Podrobný přehled konstrukce Robot.v1 z přední strany

Robot.v1 je multimodální nízkonákladová fyzická platforma oválného tvaru, která byla vytisknuta na 3D tiskárně. Toto zařízení bylo vyvinuto s cílem poskytnout cenné dostupné řešení pro testování a aplikace ve vědeckých úlohách zaměřených na asistenci, přirozenou komunikaci či jinou aplikaci úloh strojového učení. Rozpočet na celou platformu byl stanoven na cca 10 000 korun.



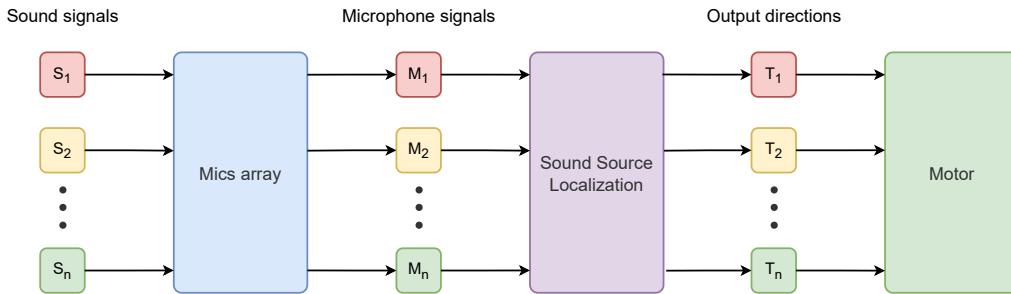
Obrázek 3.2: Přehled konstrukce Robot.v1 z boku

Na obrázkách 3.1 a 3.2 je znázorněna konstrukce robotické entity. Jak můžeme vidět na obrázku 3.2, základní konstrukce robota se skládá ze statické spodní části a horního rotačního prvku. Robot se také skládá z mnoha dalších součástí, které umožňují zařízení provádět určité činnosti. Patří k nim kamera poskytující vizuální informace, reproduktor umožňující komunikaci s uživatelem, displej pro ovládání nastavení robota, světelné senzory a solární panel pro efektivní nabíjení zařízení, krokový motor umožňující otáčení horní části a také vysoce výkonné mikrofonní pole pro vnímání a lokalizaci zdroje zvuku. Výpočetní úlohy se provádějí na počítači Raspberry Pi 4B který slouží jako základní procesorová jednotka. Softwarová implementace *Robot.v1* se opírá o systém ROS (Robot Operating System), který slouží jako hlavní framework pro integraci a koordinaci jednotlivých úloh a zajišťuje stabilní zpracování.

3.2 Detekce polohy řečníka v prostoru

Základní způsob přenosu zvukových signálů přijatých mikrofonem do cílového objektu, v tomto případě motoru, funguje ve vztahu 1:1 (viz Obrázek 3.3). To znamená,

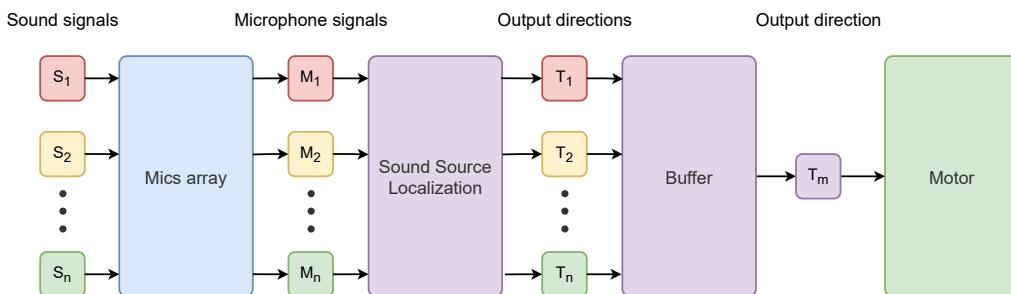
že každý zvukový signál má odpovídající výstup, který nese informaci o jeho relativním směru. Tato metoda má tu nevýhodu, že v procesu lokalizace zvuku a dalšího přenosu DOA (Direction Of Arrival - směr zvukové vlny) do motoru se nepoužívá žádné zpracování nebo filtrování chybných zvukových signálů (šum, cizí zvuky atd.). To může způsobit, že tyto vstupní údaje ovlivní výsledek a počet výstupních dat, což následně způsobí, že robot bude provádět časté a zbytečné otáčky chybným směrem.



Obrázek 3.3: Princip nalezení DOA bez filtrování vstupního signálu

3.2.1 Filtrace vstupních zvukových signálů

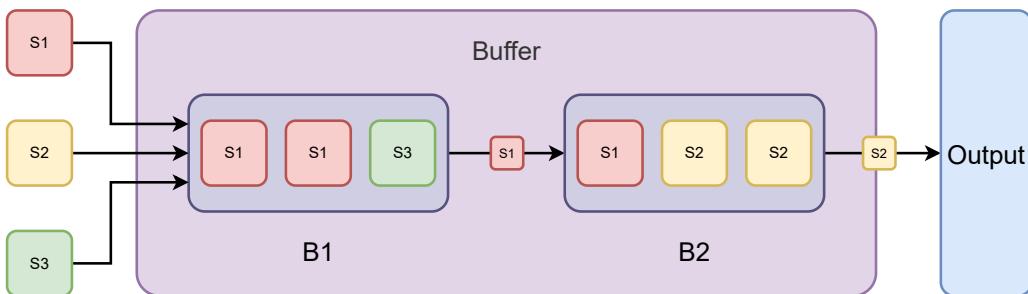
Aby se minimalizovala chyba při určování polohy zdroje zvuku, bylo rozhodnuto použít jednoduchý *Buffer* (viz Obrázek 3.4), který by tento problém vyřešil. Použití *Buffru* také snižuje počet výsledných DOA, které jsou pak prostřednictvím systému ROS předány robotické entitě, aby se otočila daným směrem.



Obrázek 3.4: Diagram vlivu komponenty Buffer na vstupní a výstupní data

Buffer funguje tak, že tato komponenta se skládá ze dvou a případně více částí (na Obrázku 3.5 pojmenovaných B_1, B_2). Tyto buňky jsou sekvenční a každá z nich může uchovávat určitý lichý počet informací (na obr. 3.5 jsou to maximálně tři). Předpokládejme, že S_1, S_2, S_3 jsou tři různé zdroje zvuku postupně přijímané polem mikrofonů v určitém časovém intervalu. Tato data se nejprve vloží do buňky B_1 ,

která shromažďuje informace, dokud nedosáhne své maximální kapacity. Poté detektuje signál, který je ve větším množství, a přesune jej do buňky $B2$ (na obr. 3.5 signál $S1$). V dalším kroku se buňka $B1$ zcela vyčistí a proces se opakuje, dokud se buňka $B2$ zcela nenaplní. Princip plnění a vymazávání buněk je pro každou součástku stejný, jen se tyto manipulace provádějí postupně od začátku ke konci. Jakmile buňka $B2$ dosáhne svém maximální kapacity, vybere se signál, které má v buňce největší zastoupení, což bude výsledkem *Buffer* komponenty (na obr. 3.5 signál $S2$). Je třeba poznamenat, že mohou nastat situace, kdy buňky obsahují odlišné signály, které se neopakují. V takovém případě se předává poslední signál.



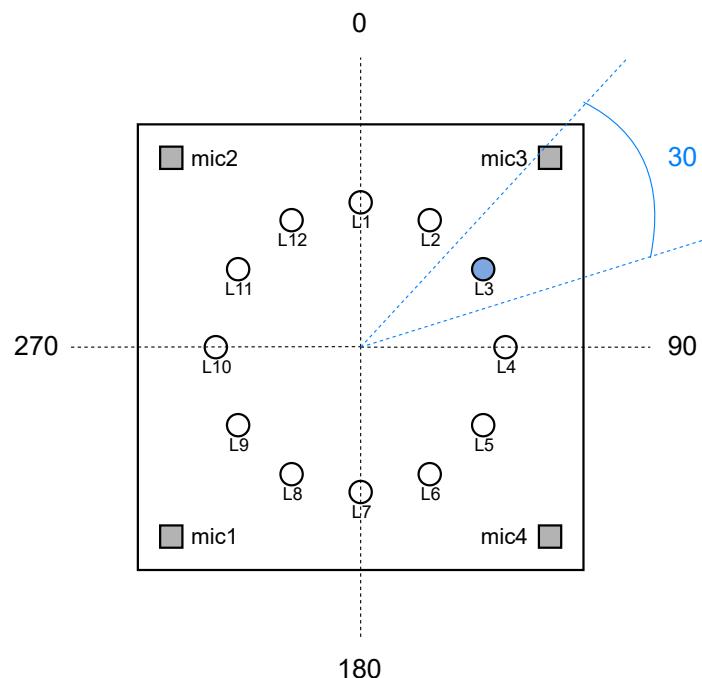
Obrázek 3.5: Detailní přehled komponenty Buffer

Tento filtrační systém lze přizpůsobit přidáním buněk a/nebo zvýšením jejich maximální kapacity.

3.2.2 Indikace směru zvuku v reálném čase

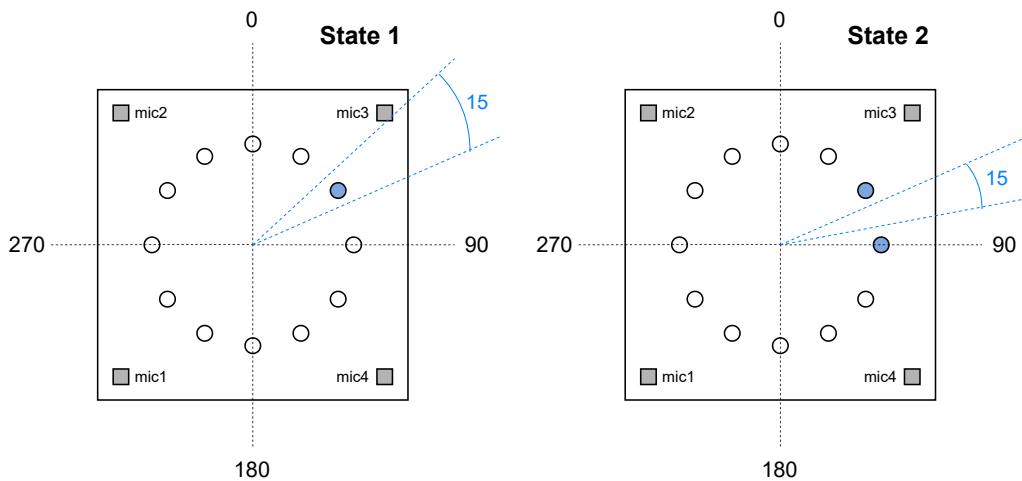
K zachycení zvukových signálů a lokalizaci směru zvuku, sloužilo mikrofonní pole ReSpeaker 4-mic array (kap. 1 sek. 1.2). Toto pole čtyř mikrofonů má integrovaný LED kroužek s 12 programovatelnými LED diodami. Pomocí tohoto kroužku je možné v reálném čase ukazovat, ze kterého směru přicházejí zvukové signály.

Omezíme se na detekci zvuku v dvourozměrném prostoru a dostaneme tak rozsah 360 stupňů pro zachycení zdroje zvuku. Protože diod je celkem 12, znamená to, že každá dioda je schopna indikovat přibližný směr v rozsahu 30 stupňů (viz Obrázek 3.6). Pro zmenšení tohoto rozsahu byl vynalezen princip s využitím přilehlých diod. Tento princip umožnil snížit rozsah relativního směru zvuku na dvojnásobek a ve výsledku činí 15 stupňů (viz Obrázek 3.7).



Obrázek 3.6: Detailní přehled LED kroužku mikrofonního pole

Použití nového principu pro zmenšení oblasti zachycení zvukových signálů mělo vliv na vznik jednoho dodatečného stavu (obr. 3.7). Stav 1 (State 1) ukazuje původní stav při použití jedné diody pouze s menším rozsahem, Stav 2 (State 2) ukazuje nový stav při použití dvou diod.



Obrázek 3.7: Dva stavy LED indikace směru zvuku

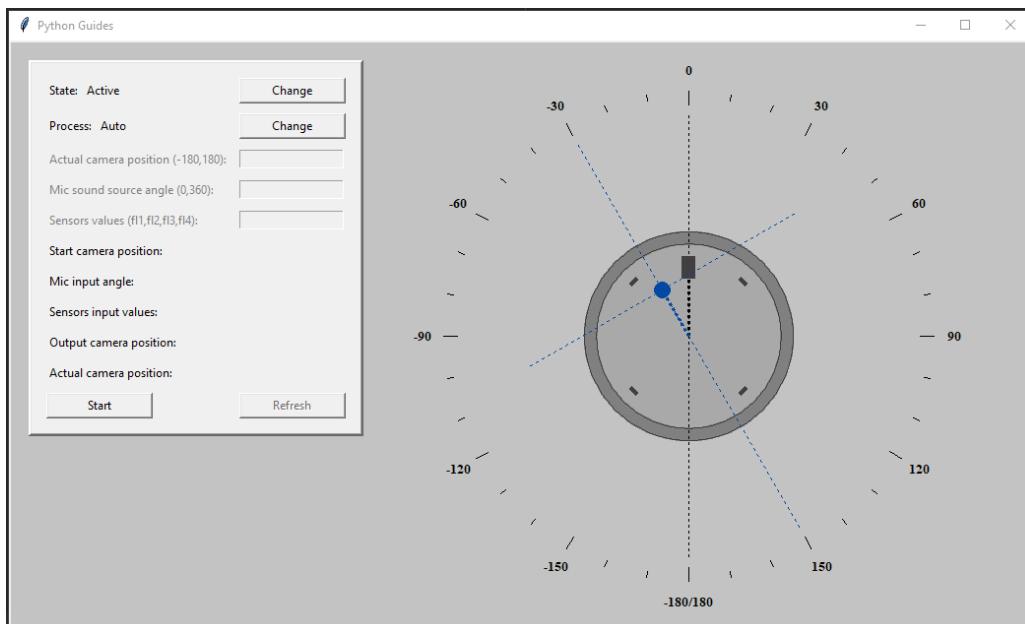
3.3 Grafická vizualizace a modelování chování cílového zařízení

S využitím jazyka Python a Tkinter frameworku (kap. 2 sek. 2.5) byla vytvořena grafická vizualizace ko^lující chování a funkce robotické entity popsané v bakalářské práci. Cílem vytvoření grafické vizualizace bylo:

- Vizuální demonstrace chování a funkcionality robota při určitých vstupních podmínkách.
- Vytvoření prostředí pro testování a realizace experimentů pomocí různých scénářů bez použití fyzického zařízení.
- Vytvoření modelu, pomocí kterého by bylo možné optimalizovat funkčnost a identifikovat případné chyby ve výpočtech.

3.3.1 Přehled aplikace a popis funkcionality

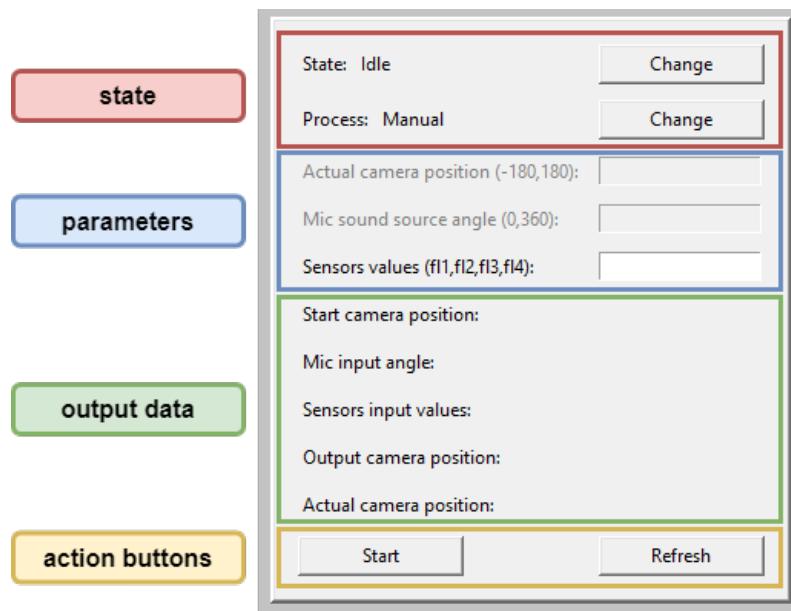
Po spuštění aplikace se zobrazí hlavní okno (Obrázek 3.8), kde se v levém horním rohu nachází sekce s nastavením a výstupními daty, a v pravé části aplikace se nachází grafická vizualizace robota ve výchozím stavu.



Obrázek 3.8: Přehled aplikace

Část s nastaveními a údaji lze rámcově rozdělit do čtyř sekcí (viz Obrázek 3.9), a to na:

- *Stavová (State) sekce*, kde můžeme vybrat stav virtuálního robota a způsob zpracování vstupních dat.
- *Sekce parametrů (Parameters)*, kde jsou umístěna okna pro ruční zadávání dat odpovídající určitému stavu.
- *Sekce s výstupními daty (Output data)*, kde jsou zobrazeny údaje popisující stav robota v prostoru a odpovídající výstup.
- *Akční tlačítka (Action buttons)*, která slouží ke spuštění nebo aktualizaci výchozí pozice vizualizace.



Obrázek 3.9: Detailní přehled nastavení aplikace

Pomocí tlačítek umístěných ve stavové části je možné měnit způsob interakce robota a způsob zpracování vstupních dat. K dispozici jsou pouze dva stavy interakce Aktivní (Active) a Nečinný (Idle), a dva procesy zpracování dat Automatický (Auto) a Ruční (Manual):

- *Aktivní (Active)* znamená, že robot je v aktivním režimu a v reálném čase snímá zvukové signály pomocí mikrofonu.
- *Nečinnost (Idle)* znamená, že robot je v pasivním režimu a pomocí světelných senzorů určuje pozici s nejvyšším dopadem slunečního světla, co zvyšuje efektivitu pasivního nabíjení zařízení.

3. Vlastní metody a jejich aplikace

- *Automatický (Auto) proces* znamená, že vstupní data jsou generována automaticky a náhodně v podmíněných mezích.
- *Ruční (Manual) proces* znamená, že je možné ručně zadat požadované parametry do příslušného pole ve stanoveném formátu.

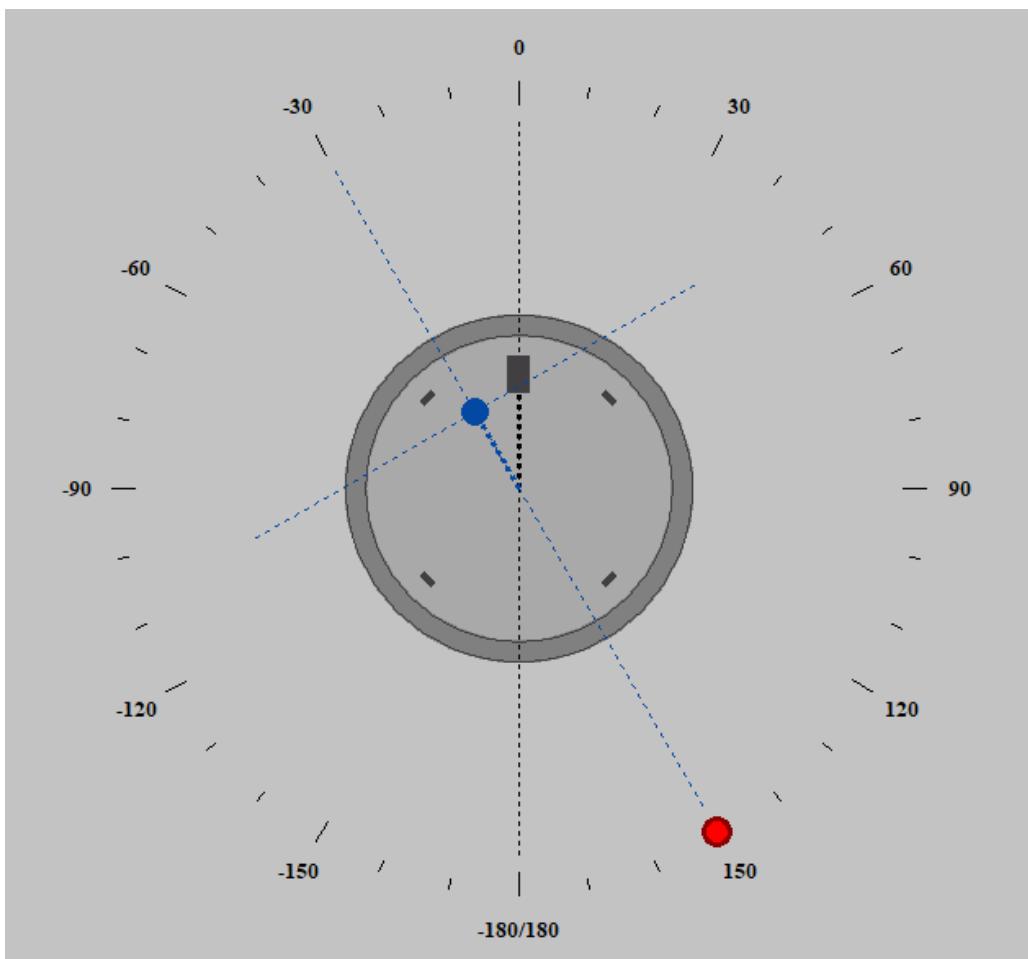
Z toho vyplývá, že celkově existují čtyři možné kombinace (viz Obrázek 3.10), jmenovitě:

- Aktivní stav s automaticky generovanými daty. V tomto stavu není možné zadávat data ručně, takže sekce parametrů není k dispozici.
- Aktivní stav s ručním zadáváním dat. V této situaci máme k dispozici parametry, do kterých můžeme zadat aktuální polohu kamery v prostoru v rozsahu od -180 do 180 stupňů a směr zvukového signálu přijímaného mikrofonem v rozsahu od 0 až 360 stupňů.
- Nečinný stav s automaticky generovanými daty. Při tomto stavu také není možné používat parametry a jsou nedostupné.
- Nečinný stav s ručním zadáváním dat. V tomto případě můžeme použít parametr a zadat řetězec se čtyřmi hodnotami ve formátu float oddělenými čárkami, které označují intenzitu světla přijatou každým světelným senzorem. V tomto pořadí: s1 (poloha -135 stupňů), s2 (poloha -45 stupňů), s3 (poloha 45 stupňů), s4 (poloha 135 stupňů).

Active	Idle																				
<table border="1"><tr><td>State: Active</td><td><input type="button" value="Change"/></td></tr><tr><td>Process: Auto</td><td><input type="button" value="Change"/></td></tr><tr><td>Actual camera position (-180,180):</td><td><input type="text"/></td></tr><tr><td>Mic sound source angle (0,360):</td><td><input type="text"/></td></tr><tr><td>Sensors values (f11,f12,f13,f14):</td><td><input type="text"/></td></tr></table>	State: Active	<input type="button" value="Change"/>	Process: Auto	<input type="button" value="Change"/>	Actual camera position (-180,180):	<input type="text"/>	Mic sound source angle (0,360):	<input type="text"/>	Sensors values (f11,f12,f13,f14):	<input type="text"/>	<table border="1"><tr><td>State: Idle</td><td><input type="button" value="Change"/></td></tr><tr><td>Process: Auto</td><td><input type="button" value="Change"/></td></tr><tr><td>Actual camera position (-180,180):</td><td><input type="text"/></td></tr><tr><td>Mic sound source angle (0,360):</td><td><input type="text"/></td></tr><tr><td>Sensors values (f11,f12,f13,f14):</td><td><input type="text"/></td></tr></table>	State: Idle	<input type="button" value="Change"/>	Process: Auto	<input type="button" value="Change"/>	Actual camera position (-180,180):	<input type="text"/>	Mic sound source angle (0,360):	<input type="text"/>	Sensors values (f11,f12,f13,f14):	<input type="text"/>
State: Active	<input type="button" value="Change"/>																				
Process: Auto	<input type="button" value="Change"/>																				
Actual camera position (-180,180):	<input type="text"/>																				
Mic sound source angle (0,360):	<input type="text"/>																				
Sensors values (f11,f12,f13,f14):	<input type="text"/>																				
State: Idle	<input type="button" value="Change"/>																				
Process: Auto	<input type="button" value="Change"/>																				
Actual camera position (-180,180):	<input type="text"/>																				
Mic sound source angle (0,360):	<input type="text"/>																				
Sensors values (f11,f12,f13,f14):	<input type="text"/>																				
<table border="1"><tr><td>State: Active</td><td><input type="button" value="Change"/></td></tr><tr><td>Process: Manual</td><td><input type="button" value="Change"/></td></tr><tr><td>Actual camera position (-180,180):</td><td><input type="text"/></td></tr><tr><td>Mic sound source angle (0,360):</td><td><input type="text"/></td></tr><tr><td>Sensors values (f11,f12,f13,f14):</td><td><input type="text"/></td></tr></table>	State: Active	<input type="button" value="Change"/>	Process: Manual	<input type="button" value="Change"/>	Actual camera position (-180,180):	<input type="text"/>	Mic sound source angle (0,360):	<input type="text"/>	Sensors values (f11,f12,f13,f14):	<input type="text"/>	<table border="1"><tr><td>State: Idle</td><td><input type="button" value="Change"/></td></tr><tr><td>Process: Manual</td><td><input type="button" value="Change"/></td></tr><tr><td>Actual camera position (-180,180):</td><td><input type="text"/></td></tr><tr><td>Mic sound source angle (0,360):</td><td><input type="text"/></td></tr><tr><td>Sensors values (f11,f12,f13,f14):</td><td><input type="text"/></td></tr></table>	State: Idle	<input type="button" value="Change"/>	Process: Manual	<input type="button" value="Change"/>	Actual camera position (-180,180):	<input type="text"/>	Mic sound source angle (0,360):	<input type="text"/>	Sensors values (f11,f12,f13,f14):	<input type="text"/>
State: Active	<input type="button" value="Change"/>																				
Process: Manual	<input type="button" value="Change"/>																				
Actual camera position (-180,180):	<input type="text"/>																				
Mic sound source angle (0,360):	<input type="text"/>																				
Sensors values (f11,f12,f13,f14):	<input type="text"/>																				
State: Idle	<input type="button" value="Change"/>																				
Process: Manual	<input type="button" value="Change"/>																				
Actual camera position (-180,180):	<input type="text"/>																				
Mic sound source angle (0,360):	<input type="text"/>																				
Sensors values (f11,f12,f13,f14):	<input type="text"/>																				

Obrázek 3.10: Všechny možné stavy aplikace

Sekce s výstupními daty nám poskytuje aktuální informace o robotovi v prostoru, o zadaných hodnotách a hodnotách, které jsou výsledkem výpočtů. V sekci jsou zobrazeny tyto parametry: počáteční poloha kamery, směr zvuku přijímaného mikrofonem, údaje ze světelných senzorů, výsledná poloha kamery, aktuální poloha kamery v reálném čase. V sekci s akčními tlačítka, tlačítko *Start* spustí simulaci zařízení na základě zadaných údajů. Tlačítko *Refresh* je aktivní pouze ve stavu ručního zadávání dat a aktualizuje počáteční polohu zařízení na základě zadaných parameterů.



Obrázek 3.11: Přehled vizuální části aplikace

Na Obrázku 3.11 je zobrazena vizuální část aplikace, s parametry: kde počáteční poloha kamery je ve směru 0 stupňů a zvukový signál přijímaný mikrofonem je ve směru 180 stupňů. Na vizualizaci robota je vidět podlouhlý čtvercový objekt, který je kamerou zařízení. Vlevo od něj je objekt kulatého tvaru, který je mikrofonem zařízení. Kulatý objekt umístěný na 150 stupních je označení zdroje zvuku, v jehož

směru se má kamera natočit. Otáčení zařízení se provádí v rozmezí od -180 do 180 stupňů a uplnou otáčku provádí pouze s průchodem polohy 0 stupňů.

3.4 Implementace technologie KWS

Bibliografie

1. *ReSpeaker 4-Mic Array for Raspberry Pi | Seeed Studio Wiki* — [wiki.seeedstudio.com \[https://wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/\]](https://wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/). [B.r.]. [Accessed: 09-10-2023].
2. *123-3D.co.uk - 3D-printers | kits | parts | filament* — [123-3d.co.uk \[https://www.123-3d.co.uk/123-3D-NEMA17-stepper-motor-1-8-degrees-per-step-47mm-long-5-0-kg-cm-SL42S247A-i2210-t12883.html\]](https://www.123-3d.co.uk/123-3D-NEMA17-stepper-motor-1-8-degrees-per-step-47mm-long-5-0-kg-cm-SL42S247A-i2210-t12883.html). [B.r.]. [Accessed: 09-10-2023].
3. *TSL2591 - High Dynamic Range Digital Light Sensor* — [botland.store \[https://botland.store/sensors-of-light-and-color/18233-tsl2591-high-dynamic-range-digital-light-sensor-stemma-qtqwiic-adafruit01980--5904422308131.html\]](https://botland.store/sensors-of-light-and-color/18233-tsl2591-high-dynamic-range-digital-light-sensor-stemma-qtqwiic-adafruit01980--5904422308131.html). [B.r.]. [Accessed: 09-10-2023].
4. GRONDIN, Francois; MICHAUD, Francois. *Lightweight and Optimized Sound Source Localization and Tracking Methods for Open and Closed Microphone Array Configurations*. 2018. Dostupné z arXiv: 1812.00115 [eess.AS].
5. GRONDIN, François et al. ODAS: Open embeddeD Audition System. *Frontiers in Robotics and AI*. 2022, roč. 9. ISSN 2296-9144. Dostupné z DOI: 10.3389/frobt.2022.854444.
6. *On-device Voice Recognition Intro - Picovoice Docs* — [picovoice.ai \[https://picovoice.ai/docs/\]](https://picovoice.ai/docs/). [B.r.]. [Accessed: 12-10-2023].
7. *GitHub - seasalt-ai/snowboy: DNN based hotword and wake word detection toolkit (model generation included)* — [github.com \[https://github.com/seasalt-ai/snowboy\]](https://github.com/seasalt-ai/snowboy). [B.r.]. [Accessed: 20-10-2023].
8. *GitHub - cmusphinx/pocketsphinx: A small speech recognizer* — [github.com \[https://github.com/cmusphinx/pocketsphinx\]](https://github.com/cmusphinx/pocketsphinx). [B.r.]. [Accessed: 20-10-2023].
9. *Wake Word Detection Engine Benchmark - Picovoice Docs* — [picovoice.ai \[https://picovoice.ai/docs/benchmark/wake-word/\]](https://picovoice.ai/docs/benchmark/wake-word/). [B.r.]. [Accessed: 16-10-2023].
10. *ROS/Introduction - ROS Wiki* — [wiki.ros.org \[http://wiki.ros.org/ROS/Introduction\]](http://wiki.ros.org/ROS/Introduction). [B.r.]. [Accessed: 20-10-2023].

Bibliografie

11. *ROS/Concepts - ROS Wiki* — [wiki.ros.org](http://wiki.ros.org/ROS/Concepts) [<http://wiki.ros.org/ROS/Concepts>]. [B.r.]. [Accessed: 20-10-2023].
12. *actionlib - ROS Wiki* — [wiki.ros.org](http://wiki.ros.org/actionlib) [<http://wiki.ros.org/actionlib>]. [B.r.]. [Accessed 23-10-2023].
13. M, Remi. *What is Tkinter used for and how to install this Python Framework?* — [activestate.com](https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/) [<https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/>]. [B.r.]. [Accessed: 24-10-2023].
14. *Introduction to Tkinter - GeeksforGeeks* — [geeksforgeeks.org](https://www.geeksforgeeks.org/introduction-to-tkinter/) [<https://www.geeksforgeeks.org/introduction-to-tkinter/>]. [B.r.]. [Accessed: 24-10-2023].
15. SUTTON, Richard S.; BARTO, Andrew G. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. Dostupné také z: <http://incompleteideas.net/book/the-book-2nd.html>.
16. *What is Reinforcement Learning? – Overview of How it Works | Synopsys* — [synopsys.com](https://www.synopsys.com/ai/what-is-reinforcement-learning.html) [<https://www.synopsys.com/ai/what-is-reinforcement-learning.html>]. [B.r.]. [Accessed: 12-12-2023].
17. *What is Q-Learning: Everything you Need to Know | Simplilearn* — [simplilearn.com](https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning) [<https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning>]. [B.r.]. [Accessed: 14-12-2023].

Seznam obrázků

1.1	Přehled mikrofonu ReSpeaker 4-Mic Array použitého v práci [1]	4
1.2	ReSpeaker 4-Mic Array mikrofon propojený s Raspberry Pi 4B [1]	4
1.3	Krokový motor NEMA17 [2]	5
1.4	Světelný senzor TSL2591 [3]	6
2.1	ODAS schéma zpracování příchozího zvuku [5]	8
2.2	Využití CPU (single-core) Porcupine ve srovnání s jinými populárními nástroji [7, 8] pro detekci klíčového slova [9]	10
2.3	Porovnání přesnosti detekce klíčového slova Porcupine s jinými nástroji [7, 8, 9]	10
2.4	RL	13
2.5	Q-learning	15
2.6	Bellmanova rovnice [17]	15
3.1	Podrobný přehled konstrukce Robot.v1 z přední strany	17
3.2	Přehled konstrukce Robot.v1 z boku	18
3.3	Princip nalezení DOA bez filtrování vstupního signálu	19
3.4	Diagram vlivu komponenty Buffer na vstupní a výstupní data	19
3.5	Detailní přehled komponenty Buffer	20
3.6	Detailní přehled LED kroužku mikrofonního pole	21
3.7	Dva stavy LED indikace směru zvuku	21
3.8	Přehled aplikace	22
3.9	Detailní přehled nastavení aplikace	23
3.10	Všechny možné stavy aplikace	24
3.11	Přehled visuální části aplikace	25

