UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

NÁVRH ROBOTICKÉHO MANIPULÁTORU S FUNKCÍ VYHÝBÁNÍ SE PŘEKÁŽKÁM

Matouš Volák

Diplomová práce

2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Matouš Volák**
Osobní číslo:	**I22203**
Studijní program:	**N0714A150005 Automatické řízení**
Téma práce:	**Návrh a vývoj robotického manipulátoru s funkcí vyhýbání se překážkám na základě hloubkové mapy**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je návrh a vývoj robotického manipulátoru, který bude schopen se při přenášení objektů vyhýbat překážkám v určitém prostoru vymezeném fixním umístěním hloubkové kamery. V teoretické části práce student provede krátkou literární rešerši s cílem prozkoumat stávající metody a technologie v oblasti robotické manipulace a zpracování hloubkových map. Konkrétně se zaměřením na aktuální stav v oblasti konstrukce robotických manipulátorů, metod pro zpracování hloubkových map pro detekci a lokalizaci překážek a plánování trasy s ohledem na dynamické prostředí.
V praktické části se student zaměří na samotný návrh a vývoj robotického manipulátoru, zahrnující konstrukční návrh manipulátoru, výběr a implementaci hardwarových komponent a vývoj řídicích algoritmů robota. Klíčovým prvkem praktické části bude vývoj a implementace algoritmů pro zpracování hloubkových dat a detekci možných překážek. Následně student navrhne a provede experimentální ověření vyvinutého manipulátoru, ve kterém bude hodnocena jeho schopnost efektivně manipulovat s objekty a vyhýbat se překážkám na základě zpracování hloubkové mapy. Praktická část bude zakončena vyhodnocením výsledků.

Rozsah pracovní zprávy:	**cca 50 stran**
Rozsah grafických prací:	
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam doporučené literatury:

MAIXNER, Ladislav. *Mechatronika: učebnice*. Brno: Computer Press, [2006]. Učebnice (Computer Press). ISBN 80-251-1299-3.
GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing*. Fourth edition. New York: Pearson, [2018]. ISBN 978-013-3356-724.

Vedoucí diplomové práce:	**Ing. Dominik Štursa** Katedra řízení procesů
Datum zadání diplomové práce:	**8. listopadu 2023**
Termín odevzdání diplomové práce:	**17. května 2024**

Prohlášení

Prohlašuji:

Práci s názvem návrh robotického manipulátoru s funkcí vyhýbání se překážek jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Matouš Volák v. r.

ANOTACE

Práce se zabývá návrhem robotického ramena, návrhem řídicího systému robota, návrhem kontrolního systému robota, návrhem trajektorie koncového efektoru robotického ramena v dynamickém prostředí a detekcí překážek v pracovním prostoru robotického ramena. Robotické rameno je řízeno pomocí Arduina, návrh trajektorie v dynamickém prostředí a detekce překážek je zprostředkován umělou inteligencí, kontrolní systém (program, pomocí kterého ovládá uživatel robota) je naprogramován v programovacím jazyku python.

KLÍČOVÁ SLOVA

manipulátor, robotické rameno, akční člen, aktuátor, neuronová síť, umělá inteligence

TITLE

DESIGN OF ROBOTIC MANIPULATOR WITH OBSTACLE AVOIDANCE FUNCTION

ANNOTATION

The work deals with the design of the robotic arm, the design of the robot control system, the design trajectory of the end effector of the robotic arm in a dynamic environment and the detection of obstacles in the workspace of the robotic arm. The robotic arm is using Arduino, the design of the trajectory in a dynamic environment and the detection of obstacle is mediated by artificial intelligence, the control system (the program with which the user controls the robot) is programmed in the python programming language.

KEYWORDS

manipulator, robotic arm, actuator, neural network, artificial intelligence

OBSAH

SEZNAM ZKRATEK A ZNAČEK

ToF	time of flight
LED	svítivá dioda
CCL	algoritmus pro detekci kontur
YOLO	algoritmus pro detekci objektů
SSD	algoritmus pro detekci objektů
RoI	potenciální místa pro výskyt objektu
RPN	regional proposal network
COCO	common objects in context
CNN	konvoluční neuronová síť
SPP	spatial pyramid pooling
RGB	barevný model
PRM	probabilistic roadmap
T-PRM	temporal probabilistic roadmap
RRT	rapidly expanding tree
PMP	princip Pontryaginova maxima
DQN	Deep q network
DDPG	Deep deterministic policy gradient
DDQN	Double deep q network
A3C	Actor-Critic
SVM	support vector machine
POM	polyoxymethylen
I2C	sériová sběrnice
I/O	vstup/výstup
EEPROM	přepisovatelná nevolatilní pamět
GIO	general input/output

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

n	převodový poměr
d	hloubka pixelu,
px/py	x a y souřadnice pixelu,
ppx/ppy	x a y souřadnice středu promítání obrazu,
fx/fy	ohnisková vzdálenost,
x/y	reálné souřadnice.

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

Umělá inteligence se stala velice populární a přístupnou, díky knihovnám jako je tensorflow nebo pytorch, které umožňují nejen jednoduše vytvořit model, ale zároveň model vytrénovat. Díky umělé inteligenci je možné řešit různé problémy velmi efektivně.

Jedním z těchto problémů je vyhýbání se překážkám. Jde o problém, při kterém je nutno jednak rozpoznat překážku v cestě a zároveň navrhnou možnou trajektorii, která překážku spolehlivě obejde. Nemusí být snadné detekovat překážky v pracovním prostoru robota, a tak jedním z řešení se nabízí umělá inteligence stejně jako pro generování trajektorie, která obchází překážku.

Cílem práce je jednak navrhnout a sestavit robotický manipulátor, který bude použit pro testování umělé inteligence a následně navrhnout algoritmy, které budou využívat umělou inteligenci k detekování překážek a ke generování trajektorie.

Výsledkem práce by měl být manipulátor, který by měl být schopen detekovat případné překážky, navrhnout trajektorii z jednoho bodu do druhého v rovině a tím se vyhnout překážkám. Manipulátor by měl být schopen tímto způsobem manipulovat s objekty bez kolize s překážkami.

1. HLOUBKOVÁ MAPA

Hloubková mapa je obrázek, většinou ve stupních šedé, který zobrazuje třetí rozměr jednotlivých pixelů neboli vzdálenost jednotlivých pixelů od čočky fotoaparátu nebo kamery. Tuto informaci pak můžeme použít k detekci objektů, segmentaci obrázku nebo měření velikosti objektů. Hloubkové mapy se používají k rekonstrukci 3D objektů, používají se v robotice k přesné navigaci robota v prostředí ale také k správné manipulaci/uchopení objektů, využívá se v augmentované nebo virtuální realitě. Pro získání hloubkové mapy můžeme využít:

- stereovidění,
- ToF (Time of Flight) kamery,
- promítání strukturovaného světla.

1.1. TVORBA HLOUBKOVÉ MAPY

V případě stereovidění hloubková kamera využívá dvou normálních dvourozměrných kamer. Kamery jsou umístěny s odstupem v jedné rovině, takže snímají stejnou scénu každá z jiného úhlu. Výsledkem jsou dva podobné obrázky, které jsou zpracovány různými algoritmy pro výpočet vzdálenosti jednotlivých pixelů. Principiálně fungují tyto kamery stejně jako člověk vnímá vzdálenost objektů pomocí očí.

ToF kamera měří dobu odrazu světla, které vysílá, od objektů scény, kterou právě snímá a na základě této doby pak určuje hloubku jednotlivých pixelů. Zdrojem světla bývá buď laser nebo LED (Kumar, 2023).

Hloubkovou mapu lze získat tím, že budeme promítat nějaký vzor na scénu, kterou snímáme kamerou. Jestliže se v scéně vyskytují objekty, nebo nerovný povrch dojde k deformaci a zkreslení vzoru, který promítáme a pomocí těchto deformací a zkreslení lze pak zkonstruovat hloubkovou mapu (Zivid, 2023).

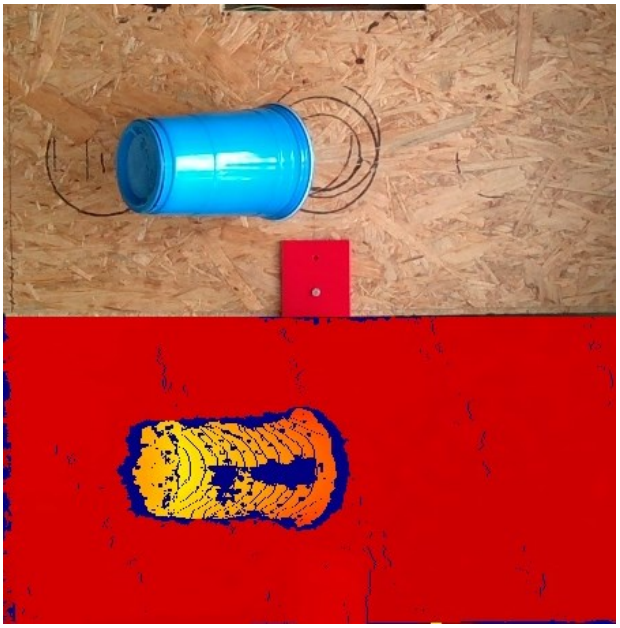
Pro získání hloubkové mapy lze také využít 3D skener případně pokud chceme získat hloubkovou mapu nějakého prostředí, pořídit množství fotek z různých úhlů a využít softwaru, který z těchto fotografií zkonstruuje hloubkovou mapu, nicméně technologie zmíněné výše se používají především pro aplikace, které potřebují konstrukci hloubkové mapy v reálném čase.

Porovnáme-li tři technologie pro tvorbu hloubkové mapy, zmíněné výše, nejpřesnější metodou je promítání strukturovaného světla, která dosahuje přesnosti až mikrometry. Jelikož má vlastní zdroj světla, vzor, který promítá na scénu, není tato metoda ovlivněna nízkým

osvětlením. ToF kamera dosahuje přesnosti až milimetry a díky svému světelnému zdroji je použitelná při nízkém osvětlení. ToF kamera je ovšem co se týká odezvy nejrychlejší z těchto technologií. Obě tyto technologie jsou použitelné potenciálně i pro větší vzdálenosti, naproti tomu stereovidění lze použít pouze do určité vzdálenosti, jelikož pak není možné vyhodnocovat rozdíly ze dvou obrázků, poněvadž už nejsou tak patrné. Ačkoliv získávat hloubkovou mapu pomocí stereovidění má jisté nedostatky a přesností dosahuje pouze na centimetry, jde taky o metodu relativně levnou, lze využít i mimo uzavřené prostory. Dále se předpokládají hloubkové mapy vytvořená pomocí stereovidění (Kumar, 2023).

1.2. ZPRACOVÁNÍ HLOUBKOVÉ MAPY

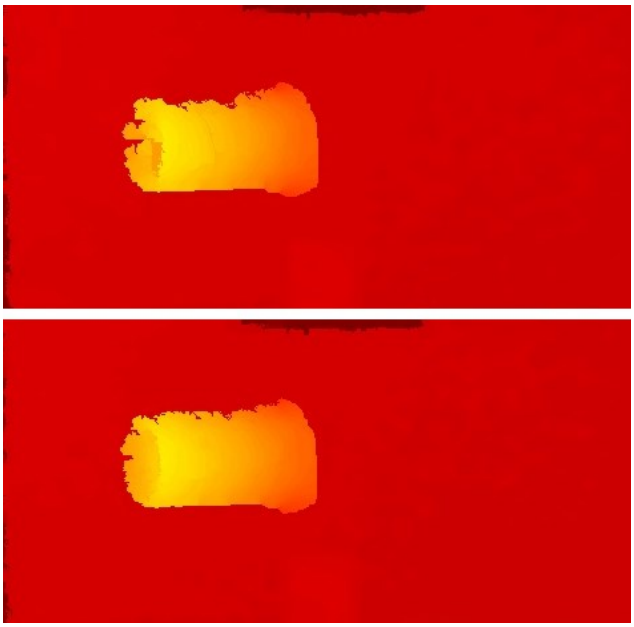
Na Obr. 1.1 je vidět v horní polovině obrázku barevný snímek pořízen hloubkovou kamerou kelímku nakloněného na podložce a v dolní polovině obrázku hloubkovou mapu tohoto snímku zobrazenou barevně, kde modrá barva odpovídá hloubce nula mm. Čím jsou objekty na scéně ke kameře blíž, barva přechází z červené na žlutou. Je zřejmé, že uprostřed kelímku, nebo na jeho okrajích nebude vzdálenost od kamery nula. Tyto segmenty, kde hloubka odpovídá nulové vzdálenosti se nazývají díry. Je možné se s nimi vypořádat pomocí filtrů.



Obr. 1.1 – Barvený obrázek, hloubková mapa

Na Obr. 2.1 je vidět hloubková mapa stejné scény, ale s aplikací filtrů, konkrétně prostorový filtr a filtr pro vyplnění děr. Dále je možné vidět, jak hodně závisí přesnost hloubkové mapy na osvětlení. V horní polovině obrázku je vidět scéna bez osvětlení horní

strany kelímku a ve spodní polovině obrázku je kelímek nasvícený. Tyto snímky byli pořízeny hloubkovou kamerou využívající stereovidění Intel Realsense D456.



Obr. 1.2 – Filtrovaná hloubková mapa, filtrovaná a nasvícená hloubková mapa

1.3. FILTRACE HLOUBKOVÝCH MAP

Pro práci s hloubkovými mapami, ať už jsou určeny k měření, vytvoření 3D modelu/mesh nebo k detekci objektů, je zapotřebí hloubkovou mapu filtrovat, avšak volba filtrů není triviální záležitost. Filtry můžeme rozdělit podle účelu:

- filtry pro vyhlazení,
- filtry pro zvýraznění hran,
- morfologické filtry.

Vyhlazovací filtry, jako je například Gaussovo rozostření nebo mediánový filtr, se používají především v případě, kdy je hloubková mapa určena pro zobrazení. Dochází pak k vyhlazení zobrazení hloubkové mapy ve stupních šedi.

Pokud však chceme s hloubkovou mapou dále pracovat, a ne ji pouze zobrazovat, např. pokud chceme detekovat objekty, je vhodné použít filtry pro zvýraznění hran. Dochází tak ke zvýraznění některých vlastností objektu, což nám dále pomůže s detekcí objektů. Mezi tyto filtr patří např. Sobelův filtr nebo Cannyho filtr.

Jak bylo ukázáno na Obr. 1.1 při tvorbě hloubkové mapy pomocí stereovidění, dochází k tvorbě tzv. děr a zároveň k nepřesnostem. K tomu abychom odstranily tyto díry a nepřesnosti

použijeme morfologické filtry, jako je např. filtr pro vyplnění děr, nebo filtr pro odstranění šumu z hloubkové mapy.

Existují také filtry využívající kombinaci hloubkové mapy a barevného obrázku, pro zvýšení rozlišení hloubkové mapy, a dále modifikace filtrů, které dokáží vyhladit hloubkovou mapu, ale zachovat výrazné hrany atd.

2. DETEKCE OBJEKTŮ POMOCÍ HLOUBKOVÉ MAPY

Detekci objektů pomocí hloubkové mapy můžeme provádět několika způsoby:

- Převést hloubkovou mapu na point cloud a použít metodu pro detekování 3D objektů.
- Použít metody pro segmentaci na hloubkovou mapu.
- Použít metody pro zvýraznění hran na hloubkovou mapu.
- Použít neuronovou síť.

Pokud nepřevedeme hloubkovou mapu na point cloud, jde stále ve své podstatě o barevný obrázek obsahující pouze stupně šedi. Lze pak aplikovat postupy/metody, které lze použít u barevných obrázků.

2.1. DETEKCE POMOCÍ SEGMENTACE

Segmentace hloubkové mapy, případně barevného obrázku, znamená rozdělení mapy/obrázku na oblasti na základě určitých parametrů/požadavků. Lze tak rozdělit hloubkovou mapu na segment pozadí, podložky atp. a segment/segmenty, které odpovídají detekovaným objektům. Pro segmentaci se využívají:

- prahové metody,
- metody shlukování,
- regionální růst.

Některé z problémů, kterým čelí segmentace jsou osvětlení, variace objektů tříd a komplexnost pozadí. Pokud segmentujeme hloubkovou mapu, osvětlení nijak neztěžuje segmentaci, jelikož hloubková mapa nenese informaci o osvětlení, nýbrž o hloubce. Problém s variací objektů tříd může být obejit tím způsobem, že objekt detekujeme na základě hloubky neboli vzdálenosti od pozadí. Problém s komplexním pozadím je v případě segmentace hloubkové mapy nepravděpodobný, jelikož můžeme tyto nerovnosti vyfiltrovat při tvorbě hloubkové mapy. Výhodou použití hloubkové mapy je skutečnost, že hloubková mapa je již reprezentace obrázku ve stupních šedi, což může ulehčit segmentaci u některých metod.

2.1.1. Regionální růst

Regionální růst nejprve zvolí počáteční pixel a na základě určených kritérií přidává okolní pixely do regionu. Nejprve porovnává okolní pixely s původním zvoleným pixelem a dále opakuje tento krok pro přidané pixely do regionu do té doby, kdy žádné okolní pixely

nevyhovují kritériím, která byli určeny. Pro zvolení počátečního pixelu může být vypočítán histogram hloubky. Tento histogram bude pravděpodobně obsahovat špičky hloubek, které můžou odpovídat buď podložce, pozadí atd. nebo objektům, které se snažíme detekovat. Dále zvolíme jeden z pixelů, který hloubkou odpovídá jedné ze špiček histogramu (Zhang, 2013).

Přidávat pixely nebo skupiny pixelů do regionu můžeme na základě charakteristik regionů, které můžou být:

- průměr jasu,
- barva,
- textura,
- tvar,
- velikost.

Tato metoda má určité nedostatky. Použití v kombinaci s hloubkovou mapou může být náročné na výpočetní techniku, zároveň nelze oddělit dvě plochy různých materiálů. Může také dojít k nesprávné segmentaci, pokud se dva segmenty spojují do jednoho, což může nastat, pokud se budou dva objekty právě překrývat (Bebis, 2004).

2.1.2. Metoda shlukování

Tato metoda má určité podmínky, které splňuje: alespoň jeden pixel v každém shluku, pixel bude právě v jednom shluku, množina všech shluků reprezentuje celý obrázek. Metodu shlukování lze rozdělit do dvou kategorií na základě použitého algoritmu shlukování na:

- Hierarchickou,
- Tvrdé shlukování.

Hierarchickou metodu shlukování lze dále dělit na rozdělující a aglomerační. V případě rozdělující hierarchické metody, všechny pixely patří do jednoho shluku a dále jsou rekursivně shluky děleny do dalších shluků, dokud není dosažené dané kritérium. V případě aglomerační hierarchické metody shlukování probíhá přesně naopak. Každý pixel tvoří shluk a tyto shluky jsou pak slučovány opět do té doby, dokud není dosaženo dané kritérium. Tyto shluky lze reprezentovat jako hierarchická struktura, která se nazývá dendogram (Mittal, 2021).

Metoda tvrdého shlukování je populárnější metodou, díky své efektivnosti. Tato metoda shlukuje pixely do shluků na základě funkce, která vyhodnocuje podobnost v daném shluku a to, jak se liší od ostatních shluků. V každém shluku se vyhodnocuje podobnost jednotlivých

pixelů. Metoda se snaží minimalizovat kritérium podobnosti pro každý shluk. Tato metrika je počítána pomocí Euklidovské vzdálenosti (Mittal, 2021).

V obou případech je nutno definovat předem počet výsledných shluků.

2.1.3. Prahová metoda

Jde o nejjednodušší metodu pro segmentaci obrázku. Prahovou metodu lze rozdělit na:

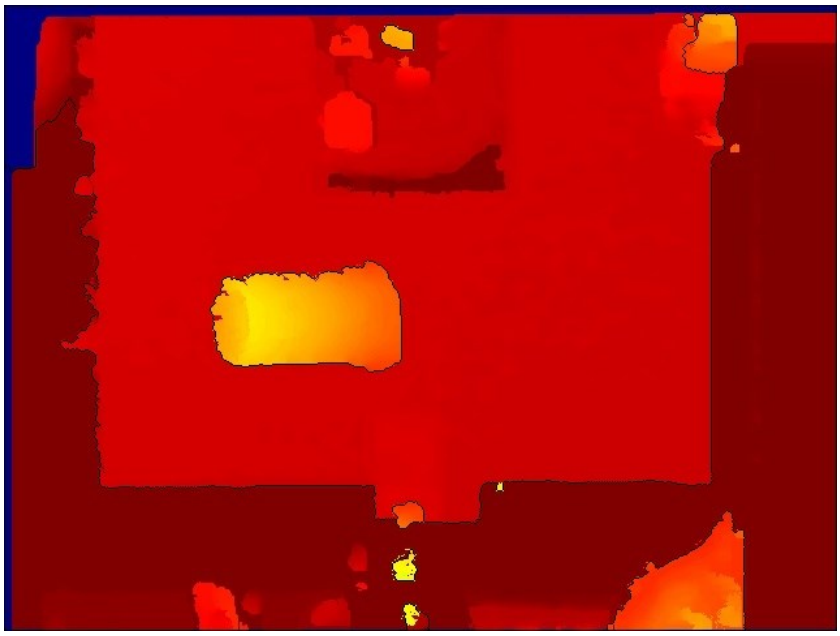
- globální práh,
- lokální práh.

Metoda převede barevný obrázek na obrázek ve stupních šedé a je určena hodnota nebo intenzita pixelu, podle které je pixelu přiřazena hodnota buď jedna nebo nula a tím je obrázek převeden na binární obrázek a dojde k oddělení popředí od pozadí.

Globální práh znamená určení jedné hranice, která rozhoduje o výsledné hodnotě pixelů, pro celý obrázek a je určen na základě průměru intenzity pixelů na obrázku. Tento způsob segmentace nemusí být vhodný, pokud má obrázek komplexní pozadí nebo není dobře nasvícen, avšak pro dobře nasvícené prostředí s jednoduchým pozadím jde o metodu velmi efektivní a výpočetně nenáročnou (Buhl, 2023).

Nedostatky předešlé metody jsou vyřešeny metodou lokálního prahu. Tato metoda určí práh pro jednotlivé sekce obrázku, které určí v závislosti na intenzitě okolních pixelů. Je pak možné oddělit i komplexnější pozadí od objektů nacházejících se v popředí, ovšem tato metoda je výpočetně náročnější. Pro volbu prahů jednotlivých sekcí obrázku existují dvě metody: mean a Gausovská adaptivní prahová metoda. V případě mean adaptivní prahové metody je práh vypočítán jako průměr intenzity pixelů jednotlivých sekcí a v případě Gausovské prahové adaptivní metody jde o vážený průměr intenzity pixelů, zatímco je dána větší priorita pixelům nacházejícím se ve středu sekce (Buhl, 2023).

Na Obr. 2.1 je vidět ukázka použití segmentace k detekování objektů pomocí hloubkové mapy. Na ukázce je vidět i okolí pracovního prostoru robota. Je vidět detekce menších oblastí v okolí pracovního prostoru, je tedy důležité zaručit podmínky, co se týče pracovního prostoru, pro správnou detekci.



Obr. 2.1 – Segmentace pomocí prahové metody

2.2. DETEKCE OBJEKTŮ POMOCÍ DETEKCE HRAN

Použití metod pro detekci hran na hloubkovou mapu opět přináší jisté výhody. Pokud máme k dispozici kvalitní hloubkovou mapu, vyhneme se problémům s osvětlením, které mohou komplikovat detekci hran a zároveň hloubková mapa má ve své podstatě zabudovanou informaci o hranách, což může ulehčit detekci hran a zvýšit kvalitu detekce.

Pro detekci objektů pomocí detekce hran postupujeme následovně:

1. předzpracovat hloubkovou mapu,
2. použití algoritmu pro detekci hran,
3. detekce kontur,
4. konstrukce objektu z kontur.

Předzpracování hloubkové mapy není nutné, ale často se používá Gaussovské vyhlazení před samotnou detekcí hran. Předzpracování je také vhodné v případě, kdy má hloubková mapa nízký kontrast, a tak je vhodné zvýšit kontrast pro lepší detekci hran.

Existují dva hlavní přístupy k detekci hran, buď na základě gradientu nebo Laplaceův přístup.

Přístup na základě gradientu nejdřív udělá první derivaci obrázku a následně vyhodnocuje tuto hodnotu. Derivace odpovídá určité změně a velikost této derivace pak odpovídá tomu o jak silnou hranu jde a zároveň směr derivace určuje směr hrany (Mutneja, 2015).

Laplaceův přístup používá druhou derivaci obrázku. V případě první derivace, čím větší hodnota derivace, tím větší je změna, a tedy silnější hrana. V tomto případě hrany odpovídají nulové hodnotě druhé derivace. Tato metoda se používá v kombinaci s Gaussovským filtrem, jelikož dochází vlivem druhé derivace k zašumění obrázku (Mutneja, 2015).

2.2.1. Metody pro detekci hran

Pro detekci hran je možno využít následujících metod:

- Sobelova metoda,
- Prewittova metoda,
- Robertsova metoda,
- Laplacian Gaussiánu,
- Cannyho metoda.

Sobelova metoda využívá konvoluční masku tak, aby detekovala změny ve vertikálních a horizontálních osách. Používá dvou kernelů, zobrazených v Tab. 2.1 a Tab. 2.2, k zvýraznění hran. Lze buď použít každý kernel zvlášť a jednotlivé výsledky sečíst anebo použít oba kernely najednou (Abushaala, 2020).

Tab. 2.1 – Kernel Sobelovy metody pro vertikální osy

1	2	1
0	0	0
-1	-2	-1

Tab. 2.2 – Kernel Sobelovy metody pro horizontální osy

-1	0	1
-2	0	2
-1	0	1

Aplikací těchto kernelů na obrázek dojde k aproximaci derivace v jednotlivých směrech. Výsledkem této metody je obrázek ve stupních šedi, kde intenzita pixelu odpovídá pravděpodobnosti výskytu hrany v původním obrázku. Výhodou této metody je nenáročnost nejen na implementaci, ale i na výpočetní techniku (Abushaala, 2020).

Prewittova metoda se od Sobelovy liší jen použitým kernelem. Jde opět o rychlou a jednoduchou metodu, ale nevýhodou je, že je zapotřebí obrázek s vysokým kontrastem bez šumu. Metoda opět využívá 3x3 konvoluční masku k aproximaci derivace v horizontálních a vertikálních osách, jejíž kernely jsou vidět v Tab. 2.3 a Tab. 2.4 (Abushaala, 2020).

Tab. 2.3 – Kernel Perwittowy metody pro horziontální osy

1	1	1
0	0	0
-1	-1	-1

Tab. 2.4 – Kernel Perwittovy metody pro vertikální osy

-1	0	1
-1	0	1
-1	0	1

Robertsova metoda využívá výpočtu 2D prostorového gradientu, jehož hodnota koresponduje s hranami. Využívá opět konvoluční masku a dvou kernelů, které zvýrazňují hrany, jež jsou pod úhlem 45 ° k mřížce pixelů v obou směrech. Jednotlivé kernely jsou vidět v Tab. 2.5 a Tab. 2.6 (Abushaala, 2020).

Tab. 2.5 – kernel Robertsovy metody v jednom směru

1	0
0	-1

Tab. 2.6 - kernel Robertsovy metody ve druhém směru

0	1
-1	0

Nejčastěji se jako vstup pro tuto metodu používá obrázek ve stupních šedé a výstupem je také obrázek ve stupních šedé. Opět je možné použít každý kernel zvlášť. Tato metoda je velmi rychlá, ale díky své jednoduchosti je náchylná na šum, který může negativně ovlivnit výslednou detekci (Abushaala, 2020).

Laplacián Gausiánu využívá Gaussovského filtru k vyhlazení, následuje výpočet druhé derivace, jejíž hodnota detekuje hrany, neboť pro oblasti, s vysokou pravděpodobností výskytu hran, bude druhá derivace nulová. Kernely, použité touto metodou, které aproximují druhou derivaci jsou vidět v Tab. 2.7 a Tab. 2.8 (Abushaala, 2020).

Tab. 2.7 – Kernel Laplacian Gausiánu pro aproximaci druhé derivace

1	1	1
1	-8	1
1	1	1

Tab. 2.8 – Kernel Laplacian Gausiánu pro aproximaci druhé derivace

-1	2	-1
2	-4	2
-1	2	-1

Cannyho metoda je nejpoužívanější metoda a jde o metodu, která je schopná detekovat jak slabé, tak silné hrany a jednou z její výhod je odolnost proti šumu. Metoda je prováděna v několika krocích:

1. filtrace obrázku,
2. výpočet derivace,
3. potlačení malých hodnot,
4. hysterezní prahování.

Cannyho metoda nejdříve aplikuje Gaussovský filtr na obrázek a následně využije Seobelovy metody pro výpočet gradientu ve vodorovné a horizontální ose. Následně v oblastech, kde jsou vysoké hodnoty a pravděpodobně hrany, aplikuje potlačení nemaximálních hodnot ve směru gradientu, takže dojde k ztenčení hran. Nyní určíme dva prahy, horní práh a dolní práh, což rozdělí hrany do tří kategorií. Pokud je pixel nad horním prahem, tak jde o hranu silnou a je zachován. Pokud je pixel pod horním prahem, ale stále nad spodním prahem, jde o hranu slabou, ale není irelevantní. Nakonec pixely, které jsou pod spodním prahem nejsou považovány za hrany. Poté pokud se nachází nějaký slabý pixel vedle silného, je mu přiřazena hodnota silného pixelu, pokud nikoliv, dále již není uvažován jako platný. Výsledkem je binární obrázek (Abushaala, 2020).

2.2.2. Detekce kontur

Výstupem metod pro detekci hran zmíněných výše je buď obrázek ve stupních šedé, nebo již binární obrázek. Pokud ne je nutno předzpracovat obrázek do binární podoby. Následně je využito algoritmů, např. označení spojených komponent (CCL), k extrakci kontur. Algoritmus přiřadí označení spojeným komponentům, v tomto případě pixelům, a dále rekonstruuje pro každé označení ohraničení objektu. Lze pak určit kritéria pro to jaké

objekty (označení) zanechat a jakým způsobem určujeme, zda jsou jednotlivé pixely spojeny (Walczyk, 2010).

Existuje několik variací toho algoritmu, mezi něž patří:

- Two pass algoritmus,
- Multiple scan algoritmus,
- Parallel processing algorimus,
- Contour tracing algoritmus,
- Single pass algoritms.

Tyto variace se liší počtem zpracování binárního obrázku, tím i složitostí algoritmu a náročností na výpočetní techniku.

V praxi se spíše využívají funkce knihoven, které jsou určeny ke zpracování obrázků např. knihovna openCV, případně se používají algoritmy pro detekci kontur bez předzpracování obrázku filtry pro zvýraznění hran.

Na Obr. 2.2 je vidět ukázka použití detekce hran k detekci objektu pomocí hloubkové mapy. Je opět vidět pracovní prostor společně s okolím, nicméně lze vidět, že metoda nezachycuje pozadí. Je vidět, že tato metoda opět zachytí menší oblasti v okolí pracovního prostoru a je opět důležité, zajistit správné podmínky pro optimální funkčnost.

Obr. 2.2 – Segmentace pomocí detekce hran

2.3. DETEKCE OBJEKTŮ POMOCÍ NEURONOVÉ SÍTĚ

Použití neuronových sítí pro detekci objektů je nejpopulárnější metoda pro detekci objektů s jednou velkou nevýhodou. Pro využití neuronových sítí spolehlivě je zapotřebí velké množství dat pro trénování neuronové sítě. Tuto nevýhodu však vykompenzují svojí přesností a v některých případech i rychlostí, nicméně použití neuronových sítí je náročné na výpočetní techniku. Např. v případě použití konvolučních sítí je nutno vstupní obrázek nejprve předzpracovat pro extrakci vlastností objektu. I navzdory těmto nevýhodám jde o metodu velmi využívanou.

Detekce objektů pomocí neuronových sítí se sestává z těchto kroků:

1. vytvoření datasetu,
2. zvolení architektury,
3. trénování modelu,
4. zhodnoceni výsledků, případné doladění.

Jak již bylo řečeno, je zapotřebí velké množství dat k vytrénování modelu. Čím větší množství dat, tím lépe, ale je důležité nepřetrénovat model, jinými slovy nedostat se do lokálního minima, a zároveň mít dostatek různorodých dat, aby nedošlo k vytrénování modelu na jednom specifickém případě. K vytvoření datasetu je zapotřebí hloubkové mapy, v nichž orámujeme objekt, který chceme detekovat a přiřadíme jim popisek. Dataset je možné „uměle“ zvětšit např. pootočením obrázku, rozmazáním nebo oříznutím, čímž dojde i k zvětšení různorodosti datasetu.

2.3.1. Volba architektury

Volba architektury, k čemuž se váže i volba přístupu detekce, není triviální záležitost. Přístup k detekci objektů pomocí konvolučních sítí můžeme rozdělit na:

- jednostupňový,
- dvoustupňový.

Pro detekci objektů je zapotřebí nejdříve učinit tzv. návrh objektů, které se nacházejí na obrázku a následně použít model, který přesně vybere objekty nacházející se na obrázku a jejich pozici. Tyto kroky se dějí separátně u dvoustupňových modelů a tím pádem tyto modely můžou být pomalejší, ale přesnější, zatímco u jednostupňových modelů jsou tyto kroky prováděny zároveň, tím pádem jsou rychlejší ale může docházet k větším nepřesnostem.

Jednostupňové modely používají konvoluční modely jejichž výstup je ve formě např. 7x7x512, což znamená 512 map výrazných vlastností vstupního obrázku. Dále rozdělíme obrázek do mřížky stejných rozměrů jako mapa výrazných vlastností a použijeme konvoluční filtr pro extrakci pravděpodobnosti toho, že se objekt nachází v určité buňce mřížky, popis objektu a velikost ohraničujícího boxu jehož souřadnice jsou vztaženy relativně k buňce. Z těchto informací jsme schopni sestavit výsledný box ohraničující detekovaný objekt. Těchto boxů však obdržíme několik, každý s vlastní pravděpodobností výskytu objektu, takže je použita komprese nemaximálních hodnot, pro kterou zvolíme hranici pravděpodobnosti a podle této hranice buď ponecháme nebo zahodíme boxy a vybereme box s největší pravděpodobností. Pokud neexistuje box s pravděpodobností větší, jaká je stanovená hranice objekt není detekován (Jordan, 2018).

Mezi tyto architektury patří:

- YOLO,
- SSD.

Dvoustupňový model, stejně jako jednostupňový, použije konvoluční síť (páteř) k extrakci výrazných vlastností do několika map. Následně jsou navrženy oblasti (RoI, Region of Interest), kde se mohou nacházet objekty. K tomu slouží RPN (regional proposal network), jehož výstup je pravděpodobnost výskytu objektu. Tyto oblasti jsou dále prozkoumány tím způsobem, že jsou opět extrahovány vlastnosti, ale jen pro dané oblasti a je opět určena pravděpodobnost výskytu objektu a upraví ohrazující box, tak aby lépe ohraničoval objekt. Výstupem opět může být více boxů ohraničující jeden objekt, a tak opět dochází ke kompresi nemaximálních hodnot stejně jako u jednostupňových architektur (Park, 2021).

Konvoluční síť extrahující vlastnosti z obrázku, tzv. páteř, je již předtrénovaná síť, většinou na velkém obecném datasetu jako je např. COCO (Common objects in contex), nicméně je možné síť doladit podle vlastních potřeb.

Mezi tyto architektury patří:

- R-CNN,
- Fast R-CNN,
- Mask R-CNN,
- SPP-Net.

Následně je nutné architekturu vytrénovat pomocí datasetu, a nakonec zhodnotit výsledky. Pro detekci objektů je vhodné vytrénovat konvoluční síť, páteř architektury, na

vlastnosti extrahované z hloubkové mapy, jelikož reprezentuje rozdílný set vlastností než barevný obrázek a následně vytrénovat architekturu na vytvořeném datasetu hloubkových map.

2.3.2. Extrakce vlastností z hloubkové mapy

Zde budou zmíněny dvě metody pro extrahování vlastností z hloubkové mapy pro konvoluční síť:

- extrakce vlastností pouze z hloubkové mapy;
- kombinace vlastností z hloubkové mapy a RGB obrázku.

Hloubková mapa obsahuje ve své podstatě informace o geometrii, hranách, natočení, výšce, které lze extrahovat.

V případě kombinace vlastností hloubkové mapy a RBG obrázku jsou tyto vlastnosti extrahovány společně s vlastnostmi z RGB obrázku a dále jsou spojeny do jedné mapy vlastností, která je použita jako vstup pro konvoluční síť. Učení konvoluční sítě probíhá pro jednotlivé vlastnosti zvlášť a vlastnosti jsou sloučené až při vstupu do klasifikátoru. Pro identifikaci bylo využito následujících vlastností: geometrie kontur, horizontální nepoměr, výška, úhel natočení (Hou, 2016).

Extrakce vlastností pouze z hloubkové mapy se jeví efektivní pouze v případě, kdy využijeme histogram normál, který je srovnatelný s histogramem gradientů u barevných obrázků (Jordan, 2013).

Na Obr. 2.3 je vidět použití algoritmu YOLO s již před trénovaným modelem. Je vidět správná detekce osob v popředí, dokonce jedné rozmazané osoby v pozadí a míče v popředí.

Obr. 2.3 – Detekce objektů pomocí YOLOv8

Výhodou této metody je, že dojde pouze k detekci definovaných objektů, na kterých byla neuronová síť natrénovaná.

2.4. POINT CLOUD DETEKCE OBJEKTŮ

Point cloud je množina bodů, z nichž každý reprezentuje x, y a z souřadnice v třírozměrném prostoru. Jde tedy o reprezentaci scény v třírozměrném prostor. Point cloud můžeme obdržet jako výstup z ToF kamery, nicméně point cloud také získáme převodem hloubkové mapy. Na point cloud pak lze aplikovat algoritmy na detekci objektů jako jsou:

- shlukování,
- segmentace,
- konvoluční síť.

Shlukování probíhá na stejném principu jako sdružování v případě RBG obrázku nebo hloubkové mapy. V závislosti na kritériu, podle kterého sdružujeme jednotlivé body se rozlišují různé algoritmy:

- K-means shlukování,
- DBSCAN,
- hierarchické shlukování,
- spektrální shlukování,

patří mezi jedny z nejpoužívanější algoritmů (Neuvition, inc, 2023).

V případě segmentace rozdělíme prostor point cloud na vexely. Vexel je kostka, tím pádem dojde k rozdělení point cloud do trojrozměrné mřížky. Dále jsou tyto vexely analyzovány a na základě daných vlastností můžeme aplikovat buď prahovou metodu, shlukování nebo regionální růst jako u RGB obrázku. Tato metoda je velmi efektivní pro velké množství dat, jelikož je možné data zpracovávat paralelně (Hrutka, 2022).

Konvoluční sítě mají opět jednu velkou nevýhodu, a to je opět potřeba velkého množství dat a zároveň jde o dataset, který není jednoduché vytvořit. I navzdory tomu, je možné využít:

- VoteNet,
- 3DETR,

které jsou určeny k 3D detekci objektů pomocí point cloud, jejichž výstup je opět box ohraničující objekt s tím rozdílem, že jde o trojrozměrný box v tomto případě (Loeb, 2022).

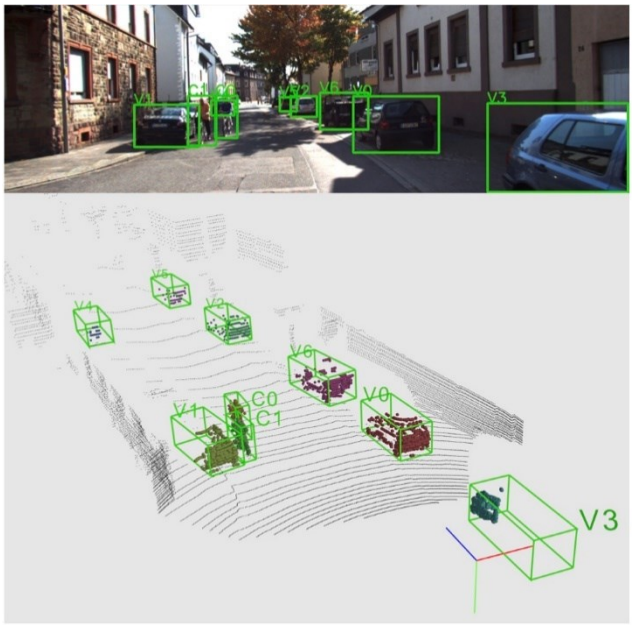
VoteNet používá Pointnet++ páteř k extrakci charakteristik. Původní point cloud je podvzorkován, ale každý bod obsahuje informace o geometrii a charakteristiky okolí. VoteNet

dále „volí“ zda je bod v oblasti, kde se potenciálně nachází objekt. Tímto způsobem dojde k vygenerování několika boxů, odhadujících, kde by se mohl objekt nacházet (Loeb, 2022).

3DETR používá transformer-based architekturu, která umožňuje soustředit se na určité vlastnosti a využití informací o delších vzdálenostech mezi body point cloudu. Používá opět páteř k předzpracování dat, následně se skládá z enkodéru. Zpracuje vstupní data a enkodér vygeneruje boxy ohraničující objekty, za předpokladu že byli některé objekty detekovány (Loeb, 2022).

V obou případech je opět nutná nemaximální komprese redundantních boxů.

Na Obr. 2.4 je vidět ukázka 3D detekce objektů pomocí point cloud a hloubkového učení. Tato metoda se používá pro detekci objektů ve větších prostorech. Point cloud je obdržen pomocí Lidaru. Zde je vidět konkrétní aplikace u autonomního auta.



Obr. 2.4 – Point-cloud detekce objektů, online, (CVPR, 2018)

3. NÁVRH CESTY V DYNAMICKÉM PROSTŘEDÍ

Návrhem cesty v dynamické prostředí je myšleno cesta, úsečka nebo křivka, po které se robot pohybuje v prostředí, které se mění případně se mění v reálném čase. Je třeba počítat s několika předpoklady. Prostředí je zjednodušeno na dvourozměrný prostor, ve kterém jsou překážky reprezentovány jako geometrické obrazce a cesta je množina bodů, které jsou spojeny. Cesta začíná a končí v námi určenými body. V závislosti na ovládaném robotu můžeme místo cesty plánovat směr a rychlost pohybu, ovšem opět diskrétně nastavovat určité hodnoty v jednotlivých časových úsecích. Cílem je tedy navrhnout cestu, po které když se bude robot pohybovat, dostane se z počátečního bodu do koncového, zatímco se bude vyhýbat překážkám. Pro tento účel lze použít:

- reaktivní návrh cesty,
- pravděpodobnostní metody,
- optimalizace trajektorie,
- Machine learning.

3.1. REAKTIVNÍ NÁVRH CESTY

Reaktivní návrh cesty se často používá u mobilních robotů, autonomních automobilů atp. Metoda nenavrhuje cestu, ale udává směr a případně rychlost pohybu robota. Z toho důvodu většinou použita u mobilních robotů v prostředích jako je např. sklad, kde se robot musí vyhýbat lidem, zatímco je jeho cílem dostat se do cílového bodu. Metoda je založena na snímání svého okolí, snímání překážek kolem sebe a následně vyhodnocování směru pohybu.

Robot má informaci o své pozici v prostoru, pozici cílového bodu a svém okolí. Plánování pohybu pak lze rozdělit na tři části:

- Pohyb k překážce – Základní chování robota, pokud nedetekuje žádné překážky v cestě. Robot detekuje překážky pouze v určité vzdálenosti a proto, pokud nedetekuje žádné překážky, pohybuje se přímou čarou k cílovému bodu;
- Vyhnutí se překážce – Pokud selže pohyb k cíli, robot detekuje překážku v určité vzdálenosti, která je v cestě a znemožňuje mu se dostat do cílového bodu, robot vypočítá novou hodnotu směru pohybu na základě okolních bodů a jejich vzdálenosti k překážce případně potenciálních kolizí nebo dalších výskytů překážek v cestě.
- Změna lokálního směru – Změna směru v předchozí fázi je v rozmezí lokálních hodnot. V případě, kdy ani to nepomůže, nebo nemůže najít cestu která by nebyla blokovaná překážkami, dojde k otočení robota o 180 °. Jde o případy, kdy se dostal robot do slepé uličky, ze které by se neměl jak dostat a mohlo by dojít k zacyklení.

(Mediavilla, 2002)

Dále byli vyvinuty modifikace tohoto algoritmu pro efektivnější plánovaní cesty:

- pole potenciálů,
- velocity obstacle algoritmus,
- hybridní metody.

První metoda využívá polí, které buď přitahují robot, nebo jej odpuzují. Přitažlivé pole je určeno vzdáleností a směrem cílového bodu, zatímco odpudivé pole je určeno překážkami v prostření. Následně je určen směr robota, který je závislý na výsledné „síle“, která ho přitahuje, která je ovlivněna překážkami. Směr je normálový vektor této „síly“, která je určena součtem přitahujícího pole a odpudivých polí. Využívá se váhových koeficientů jednolitých polí k doladění chování robota (Robotics MP, 2022).

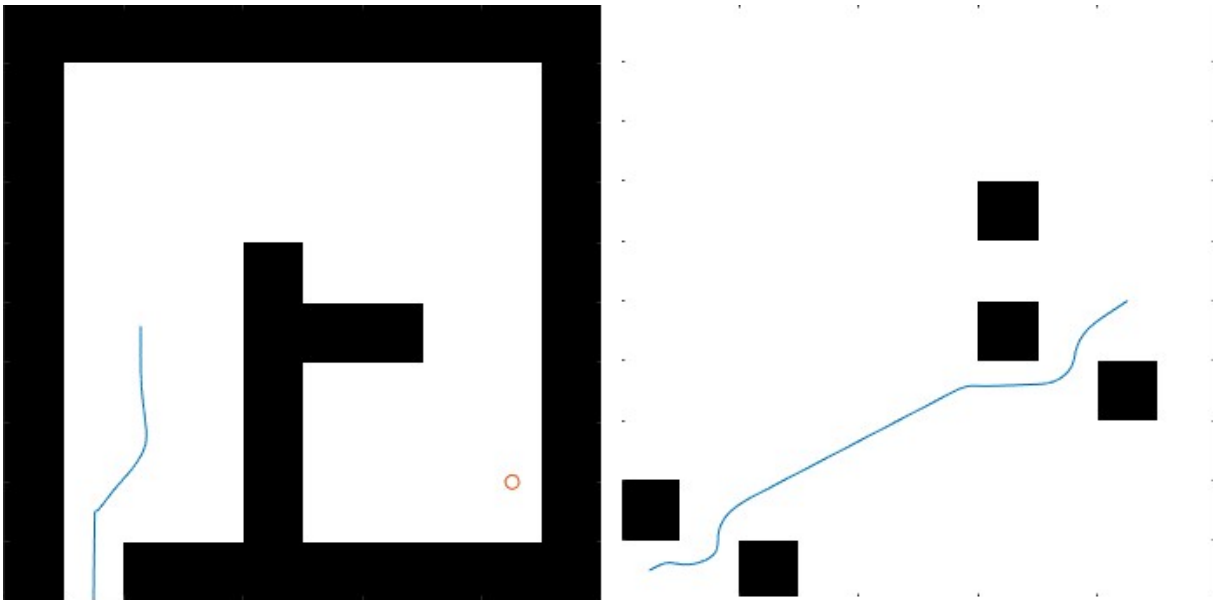
Velocity obstacle algoritmus využívá měření rychlosti překážek v jeho okolí. Je vytvořena množina rychlostí, které by vedli k potenciálnímu střetu s překážkami na základě předpovědí rychlostí překážek v prostoru (Robotics MP, 2022).

Dále existují hybridní metody, které spojují/rozšiřují zmíněné metody pro efektivnější a spolehlivější plánování cesty. Mezi ně lze řadit např.: algoritmu velocity obstacles nebo spojení algoritmů velocity obstacle algoritmu a pole potenciálů (Robotics MP, 2022).

Jednoduchost těchto algoritmů je jedna z výhod, neboť není třeba velké množství výpočetního výkonu, nicméně jde o algoritmy, které pouze reagují na okolí, což znamená že se mohou lehce ocitnout ve slepých uličkách a pokud používáme parametry pro nastavení chování robota, jsou velice citlivé na změnu těchto parametrů a nezaručují, že se dostanou do cílového bodu.

V levé polovině Obr. 3.1 je vidět pokus využití reaktivní metody potenciálu polí v testovacím prostředí, avšak algoritmus nebyl schopen navrhnout cestu, která by došla do cíle a došlo k zacyklení. Zacyklení může být způsobeno nesprávným nastavením parametrů, na což

je algoritmus velmi citlivý. V pravé polovině je pak vidět využití algoritmu v jednodušším prostředí. Algoritmus dosáhl cíle.



Obr. 3.1 – Návrh trajektorie pomocí reaktivní metody

3.2. PRAVDĚPODOBNOSTNÍ METODY

Tyto metody jsou použitelné jak u mobilních robotů, tak i u manipulátorů, avšak vzhledem k tomu, že generují cestu v podobě množiny bodů, které jsou spolu spojeny od počátečního bodu ke koncovému, hodí se spíše pro robotické manipulátory. Metody nejsou přímo určeny pro plánování cesty v dynamickém prostředí, ale je možné přizpůsobit je k plánování cesty i v dynamickém prostředí. Tyto metody využívají informace o celém prostředí, ve které se robot pohybuje a má k dispozici informace o všech překážkách nacházejících se v prostředí a z toho důvodu jsou také vhodnější pro robotické manipulátory, neboť jejich pracovní prostor je výrazně menší než u mobilních robotů, u kterých ani nemusí být možnost celý pracovní prostor v reálné čase monitorovat. K plánování cesty se hlavně využívá:

- PRM,
- T-PRM,
- RRT,
- RRT-star.

Tyto metody využívají prohledávání prostoru náhodným generováním bodů, které jsou spojeny hranami a tvoří tzv. graf. Liší se metodou generování bodů a hledáním výsledné cesty.

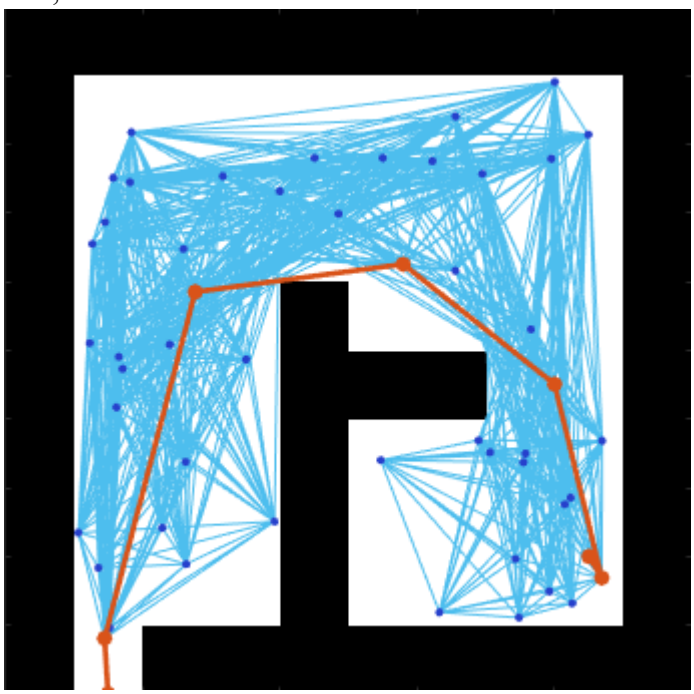
3.2.1. Probabilistic roadmap (PRM)

PRM metoda nejprve vygeneruje náhodně body ve stavovém prostoru a při každém vygenerování zkontroluje, zda nedochází ke kolizi daného bodu s překážkou, jelikož jde o potenciální souřadnice pro efektor robotického manipulátoru nebo pozice mobilního robota. Dále spojí tyto body tak, aby nedocházelo ke kolizi překážky s hranou, která vznikne spojením dvou bodů, jelikož jde o cestu, po které se bude potenciálně robot pohybovat. Tímto způsobem dojde k vytvoření grafu, jehož vrcholy a hrany se nachází ve volném prostoru. Využije se jednoho z algoritmů jako je: A-star nebo Dijkstrů algoritmus, které vyhledají nejkratší cestu k cílovému bodu. V případě, kdy buď počáteční nebo koncový bod nejsou součástí grafu, jsou spojeny s nejbližšími hranami grafu (Khokhar, 2021).

Jak již bylo řečeno, tento algoritmus není primárně určen pro plánování cesty v dynamickém prostředí a je tedy zapotřebí ho modifikovat. Jednou z možných modifikací je periodicky zkontrolovat kolize grafu a překážek a případně vygenerovat nové body a začlenit je do grafu, nebo odstranit hrany, které kolidují s překážkami. Toto může být výpočetně náročné, a tak lze využít T-PRM, který bere v potaz i čas a je tak možné zachovat původní graf, ale je nutné vědět, nebo alespoň předpovědět rychlost a směr pohybu překážky (Huppi, 2022).

Tyto algoritmy nemusí zaručit optimální cestu, ale jde o algoritmy spolehlivé. Nicméně pokud algoritmus selže generovat cestu k cílovému bodu, není možné zjistit, zda cesta neexistuje nebo algoritmus nevygeneroval potřebné body k nalezení cesty.

Na Obr. 3.2 je vidět využití algoritmu PRM pro navržení cesty v testovacím prostředí. Algoritmus navrhl cestu, která se dostala do cíle.



Obr. 3.2 – Návrh trajektorie pomocí PRM

3.2.2. Rapidly exploring random tree (RRT)

Tento algoritmus místo toho, aby vygeneroval body v celém stavovém prostoru generuje body iterativně a dochází k vytvoření a k růstu tzv. stromu. Tento strom se skládá z bodů, které jsou iterativně generovány a z hran, které je spojují. Nejdříve dojde k vygenerování náhodného bodu ve stavovém prostoru a z počátečního bodu, nebo bodu naposled přidaného do stromu, dojde k malému krůčku k tomuto bodu a dojde k vytvoření a přidání nového bodu do stromu. V určitých intervalech není jako bod, ke kterému bude strom přiblížen, vytvořen náhodně, ale jde o koncový bod, takže je algoritmus zaujatý ke koncovému bodu. K tomu dochází z toho důvodu, aby strom hledal efektivnější cestu ke koncovému bodu a nedocházelo pouze k náhodnému prohledávání stavového prostoru. RRT-star je pouze optimalizovaná verze RRT, která funguje na podobném principu jako A-star algoritmus (Govind, 2023).

Tento algoritmus je pro plánování cesty v dynamickém prostoru v reálném čase vhodnější než PRM, jelikož můžeme kontrolovat kolizi při vytváření nového bodu a není potřeba kontrolovat celou mapu, za předpokladu že dochází k pohybu robota hned po vygenerování nového bodu. Výhodou je vhodné použití jak pro robotický manipulátor, tak

mobilní robot, neboť není nutné mít informaci o celém stavovém prostoru, ale jen o lokálním okolí bodu, ve kterém se zrovna robot nachází.

3.3. OPTIMALIZACE TRAJEKTORIE

Optimalizaci trajektorie lze zařadit mezi algoritmy pro plánování cesty i navzdory tomu, že samotná metoda potřebuje nějakou cestu, ze které může vycházet. Tyto cesty můžou být buď naprosto náhodné, ale optimalizace pak může zabrat víc času a je náročnější, nebo můžou být zkombinovaná s předchozími metodami, jako je RRT nebo PRM. Pravděpodobnostní metody nezaručují cestu, která se dostane do cílového bodu ani cestu, která je optimální, ale navrhnou cestu, která není daleko od optimální a optimalizace je díky tomu rychlejší.

Optimalizace trajektorie probíhá na základě stanovených kritérii, které můžou být délka trajektorie, vyhnutí se všem překážkám, ale i např. spotřebovaná energie. V případě, kdy chceme tuto metodu použít v dynamickém prostření je nutné, stejně jako u probabilistických metod, předem znát celé prostředí a zároveň umístění a pohyb překážek.

Pro optimalizaci trajektorie lze použít:

- gradient-based metody,
- přímé metody,
- nepřímé metody,
- evoluční algoritmy.

Gradient-based metody využívají tvz. cost funkce. Jde o funkci, která reprezentuje určitým způsobem kvalitu trajektorie. Dále jsou určeny parametry, které ovlivňují to, jak vypadá trajektorie. Je určen gradient funkce vzhledem k těmto parametrům. Na základě tohoto gradientu jsou parametry přizpůsobovány tak, aby došlo k minimalizaci cost funkce. Výpočet gradientu a přizpůsobení parametrů je opakováno do té doby, dokud trajektorie nesplňuje daná kritéria. Dle využití gradientu tyto metody lze dělit na: Gradient descent, Stochastic gradient descent a Newtonova metoda. Gradient descent přizpůsobuje parametry na základě záporné derivace. Stochastic gradient descent používá náhodně vybrané body z datasetu. Newtonova metoda používá k přizpůsobení parametrů i druhého gradientu. Jelikož je trajektorie reprezentována funkcí, je nutné následně převést spojitou funkci do diskrétního popisu prostoru (Lavalle, 2020).

Přímá metoda optimalizace trajektorie optimalizuje parametry trajektorie jako jsou: časové body, řídicí signál, rychlost atd. Po určení parametrů, kterými bude optimalizováno, je

určena cílová funkce, která zahrnuje kritéria, jež budou optimalizována. Na základě zvolené metody pro optimalizaci, dojde k výpočtu nových parametrů. Výpočet je opět prováděn iterativně do té doby, dokud není splněná určená podmínka. Mezi tyto metody patří:

- Shooting metoda – metoda pracuje v diskrétním čase s počátečními stavovými hodnotami, které upravuje. Po úpravě počátečních hodnot vypočítá numericky cílovou funkci systému, a tak převádí optimalizaci na hledání správných počátečních hodnot, které dosáhnou minima cílové funkce.
- Collocation metoda – opět pracuje v diskrétním čase, ale nesnaží se najít počáteční hodnoty minimalizující cílovou funkci, nýbrž optimalizuje hodnoty stavů v diskrétních bodech trajektorie a opět používá numerického výpočtu pro zjištění hodnoty cílové funkce. Diskrétní body trajektorie se nazývají kolokační body.

(Gul, 2021)

Jelikož nevyužívá modelu systému je tato metoda vhodná v případech, kdy model není dostupný a zároveň pokud je menší množství parametrů nebo je cílová funkce relativně prostá. V případě, kdy jde o komplexnější systém je vhodné použít nepřímé metody.

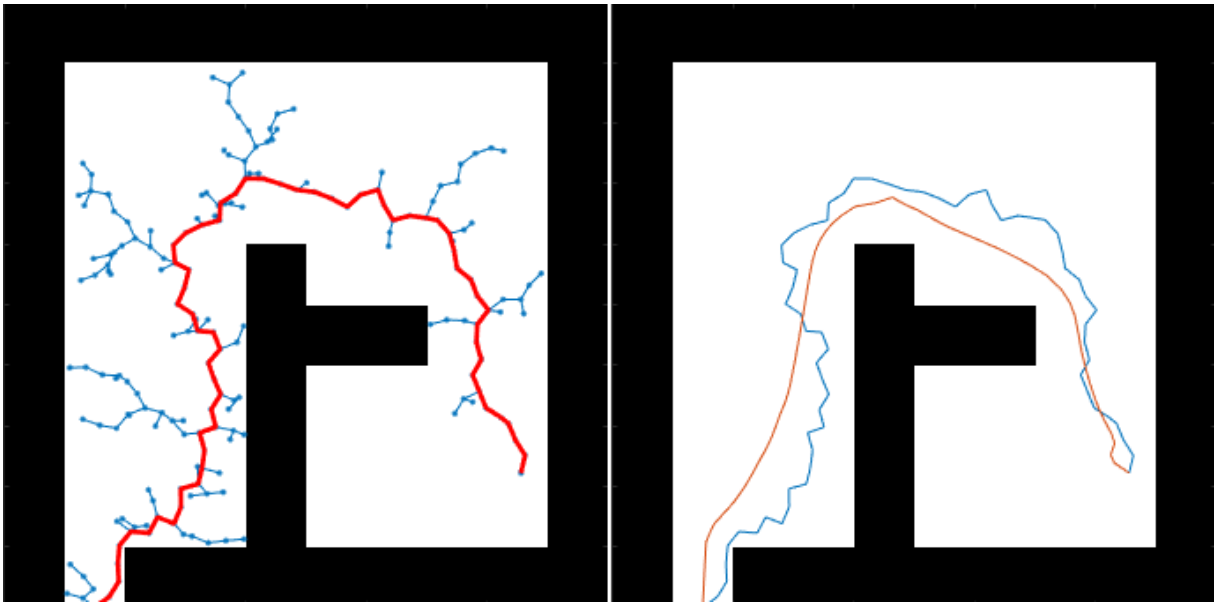
Nepřímá metoda využívá modelu systému a navrhuje řídicí signály nebo řídicí strategie, díky čemuž nepřímo optimalizuje parametry trajektorie. Je nutné definovat matematický model systému, který může být ve formě: diferenciálních rovnic, stavového popisu, přenosu atd. Opět je určena cílová funkce, která vyjadřuje kritéria pro optimalizaci. Opět je využit některý z algoritmů pro optimalizaci parametrů:

- metoda PMP (princip Pontryaginova maxima) – princip Pontryaginova maxima určuje podmínky pro optimální řízení dynamických systémů. Je nutné formulovat Hamiltonian dále maximalizovat Hamiltonian pomocí řídicích signálů.
- dynamické programování – Dynamické programování řeší problematiku pomocí rozdělení na podproblémy, které se řeší použitím principu optimality. V případě dynamického programování se řeší optimalizace od konce časového úseku.
- využití predikce řízení – Využívá se modelu systému k zjištění predikcí budoucího stavu systému na základě řídicích signálů. Díky tomu pak můžeme řešit optimalizaci s predikovaným stavem a v průběhu tak upravovat parametry a tím minimalizovat cílovou funkci.
- genetické algoritmy – U genetických algoritmů je sekce trajektorie, případně celá trajektorie, kterou lze vygenerovat pomocí modelu systému, jako jedinec generace. Dále je aplikována selekce, křížení a mutace, k zjištění optimální trajektorie.

(Gul, 2021)

Nepřímá metoda optimalizace trajektorie najde využití hlavně v případech, kdy jde o řízení komplexních systémů, které mají mnoho parametrů pro optimalizaci, a tak je výhodné využít model těchto systému k optimalizaci.

V levé polovině Obr. 3.3 je vidět využití algoritmu RRT k vytvoření prvotního odhadu cesty v testovacím prostředí a na pravé polovině je vidět optimalizace této cesty.



Obr. 3.3 – Návrh trajektorie pomocí RRT a optimalizace

3.4. MACHINE LEARNING

Nejspíš nejpopulárnější metodou pro návrh cesty robota, případně jakékoliv akce robota, je machine learning. Jednou z výhod machine learningu je možnost použití v reálném čase. Výhodou je také možnost spolehlivého využití v komplexních prostředích a není nutnost znát celé prostředí ani pozici a rychlost pohybu překážek předem, algoritmy jsou schopné reagovat za běhu a přizpůsobovat se. Machine learning lze rozdělit na:

- Reinforcement learning,
- Supervised learning.

3.4.1. Reinforcement learning

Reinforcement learning metody jsou založené na tzv. agentovy, který provádí možné akce a na základě provedené akce v určitém stavu prostředí a v určité pozici dostává odměnu. Na základě odměny se pak učí na stejném principu jako metoda pokus omyl. Mezi reinforcement learning patří:

- Q-learning,
- DQN,
- DDPG,
- DDQN,

- A3C.

V případě algoritmu Q-learning dochází k vybírání akce robota podle tabulky Q hodnot. Po tom, co agent provede akci je mu přidělena odměna, která určuje, zda akce byla „správná“ nebo „špatná“, což je určeno heuristikou. Následně je tabulka upravena v závislosti na odměně ve stavu prostředí, ve kterém byla akce provedena. Dochází ke zvýšení Q hodnot akcí, které vedou v určitých stavech na chování robot, které požadujeme (Chanda, 2024).

Deep Q-learning network (DQN) nevyužívá tabulky, ale neuronové sítě k určení Q hodnoty akcí v aktuálním stavu prostředí. Agent opět provádí akce v prostředí a na základě toho je odměněn. Tyto akce a jejich odezva, co se týče odměny ale i stavu, jsou zaznamenány do paměti. Z paměti je pak použita dávka pro učení neuronové sítě, takže nedochází k úpravě hodnot v tabulce, ale úpravě vah neuronové sítě. Neuronová síť pak určuje Q hodnoty pro akce v daném stavu prostředí (Yu, 2020).

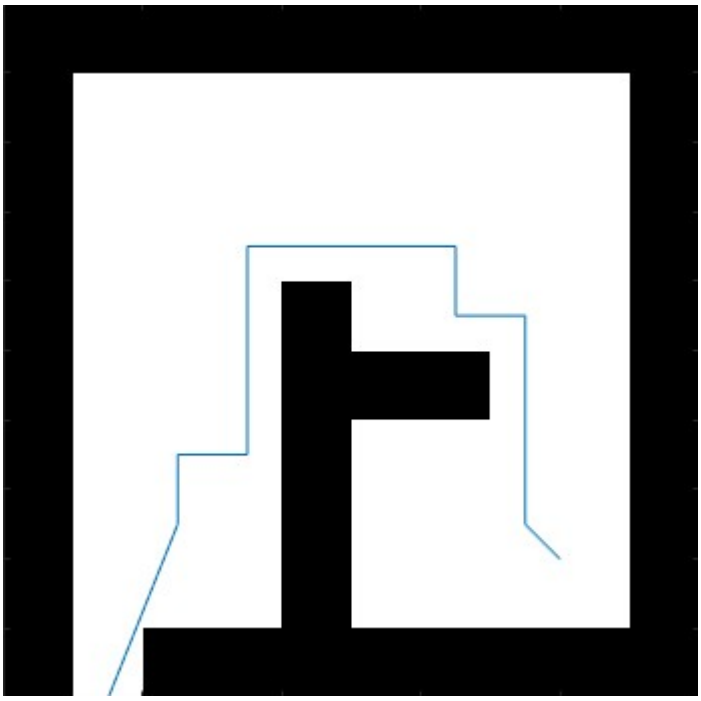
Actor-Critic (A3C) využívá jak agenta, tak kritika. Využívá dvou neuronových sítí, agenta, který dělá akce na základě určité strategie a tím prozkoumává prostředí a kritika, který určuje hodnotu akce, kterou agent provedl. Dochází tak zároveň k prozkoumávání prostředí a zároveň k využívání nabytých znalostí. K učení opět dochází prostřednictvím dávek z paměti (Yu, 2020).

Deep deterministic policy gradient (DDPG) je ve své podstatě DQN algoritmus s tím, že byl přidán A3C algoritmus. Nedochází k určení akcí pomocí dané strategie, ale pomocí DQN algoritmu, jehož akce jsou vyhodnoceny kritikem. Opět dochází k ukládání akcí, stavů, následujících stavů a odměny a následnému trénování na dávce z paměti (Yu, 2020).

Double DQN (DDQN) používá dvou neuronových sítí a opět zaznamenává akce, stavy atd. do paměti. Když pak dojde k úpravě vah neuronových sítí, používají se hodnoty druhé sítě (Yoon, 2019).

Ve všech případech agent nejdříve dělá náhodné akce a s každou provedenou akcí se zvětšuje pravděpodobnost udělat akci s největší Q hodnotou. Všechny metody využívají neuronové sítě s výjimkou Q-learning. Všechny algoritmy využívající neuronovou síť, využívají k učení také heuristické funkce, která přiděluje odměnu závislosti na výsledku provedené akce. Algoritmy, využívající neuronovou síť, mají velkou výhodu v tom, že nepotřebují dataset, ale vytvářejí si ho během učení. Jelikož na začátku agent dělá akce zcela náhodně, je vhodné využít model prostředí a model robota pro trénování.

Na Obr. 3.4 je vidět návrh cesty v testovacím prostředí pomocí algoritmu Q-learning.



Obr. 3.4 – Návrh trajektorie pomocí Q-learning

3.4.2. Supervised learning

Na rozdíl od reinforcement learning, supervised learning nevyužívá agenta, ale supervisora, který dohlíží na algoritmus. K využití této metody je třeba datasetu, který se skládá z dat, která jsou použita jako vstup k určení predikce, správné predikce a validačních dat. Tato metoda má ve své podstatě dva hlavní účely:

- klasifikace,
- predikce (IBM, 2024).

Metoda je schopná klasifikovat data do daných kategorií, případně lze využít dat s již známým výsledkem pro vytrénování algoritmu k predikci budoucího výsledku. Ačkoliv není metoda primárně používaná na plánování cesty, lze ji k tomuto účelu použít.

V případě využití této metody pro plánování trasy, nebo spíše další akce, by byla metoda použita k predikci akce robota, v tomto případě, jakým směrem se bude robot pohybovat. K trénování je tak potřeba dataset, který by obsahoval stavy prostředí, pozici robota a správnou cestu, kterou se má robot v dané situaci pohybovat. Je však potřeba znát optimální cestu v daném prostředí. Není však potřeba znát celé prostředí, ale je dostačující znát lokální okolí robota a na základě toho se pak algoritmus rozhoduje pro další akci.

K plánování cesty lze využít:

- random forest,
- support vector machine (SVM),
- decision trees,
- neuronová síť.

Decision tree je algoritmus, který se skládá ze vstupního uzle, rozhodujících se uzlů a výstupních uzlů. Vstupní uzel a následně vnitřní rozhodující uzle mají výstup rozdělen do dalších uzlů a rozhodují se jakou cestou se vydají v závislosti na vstupních datech. Tímto způsobem dojde k rozhodnutí se pro jeden z výstupních uzlů, které představují možné výsledky. Takto lze data klasifikovat, nebo předpovědět budoucí výsledek. V případě plánování cesty, tak dochází k předpovědi akce robota, směru nebo lokálního bodu v okolí robota, do kterého se má robot pohnout (IBM, 2024).

Radnom forest využívá několik decision tree modelů k předpovědi. Každý decision tree je trénován na podmnožině celkového datasetu, ale k predikci se využívá všech modelů a dochází k tzv. volení, kdy se jako platná predikce považuje ta s největším počtem predikcí všech decision tree modelů (Free Learning Platform For Better Future 2021).

SVM rozděluje data rovnou ve stavovém prostoru, kde jsou data reprezentována jako vektor. Metoda tak může na základě stavu prostředí a pozice robota rozdělit body v prostoru v okolí robota na body, které jsou součástí optimální trajektorie nebo trajektorie, která splňuje dané podmínky a na body, které jsou irelevantní pro robot, jelikož SVM rozděluje data do kategorií pomocí roviny (IBM, 2024).

V případě využití neuronové sítě, je princip funkce podobný jako u reinforcement learningu. Jediný rozdíl je v trénování sítě, k čemuž je potřeba vytvořit dataset. Výstupem neuronové sítě je opět predikce akce robota (IBM, 2024).

4. KONSTRUKCE ROBOTICKÉHO RAMENA

Konstrukcí robotického ramena je myšleno výběr správného materiálu, ze kterého bude vyroben tzv. link, který spojuje robotický řetězec dohromady a výběr typu kloubů a materiálů pro jejich konstrukci. Robotický řetězec se skládá ze základny robota, z kloubků robota a z koncového efektoru robota. Pokud je řetězec spojen od základny přes klouby až po koncový efektor pouze, jedná se o otevřený řetězec, zatímco pokud je koncový efektor spojen několika takovými otevřenými řetězci jedná se o uzavřený řetězec. Otevřený řetězec je mnohem volnější, co se týče pohybu, ale je více nepřesný, čím dál se koncový efektor nachází od základny, zatímco uzavřený řetězec je mnohem přesnější ale je velmi omezený, co se týče pohybu koncového efektoru. Při výběru materiálu je důležité zohlednit pevnost materiálu, ale zároveň hmotnost. Podle použití kombinací a druhu kloubu lze rozdělit robotická ramena podle architektury na:

- kartézský,
- cylindrický,
- sférický,
- skara,
- articulated,
- paralel.

Podle použité architektury se také odvíjí geometrie pracovního prostoru robotického ramena a přesnost koncového efektoru, případně převod souřadnic reálného světa na souřadnice robotického ramena a řešení inverzní úlohy. Nejčastěji používané klouby robotického ramena jsou: translační, rotační.

Translační kloub se využívá v případech, kdy je vyžadována přesnost pohybu a zároveň není nutná velká flexibilita pohybu. Rotační kloub je flexibilnější než translační, nicméně je nutné se vypořádat s tím, že jde o rotační pohyb, což komplikuje pohyb po trajektorii v podobě přímky. Výhodou rotačních kloubů je jejich kompaktnost a využití robotického řetězce složeného z rotačních kloubů jsou výhodné, jelikož dosahují relativně velkého pracovního prostoru a jsou poměrně kompaktní.

Robotický řetězec je pak možné řídit dvěma způsoby:

- decentralizované řízení,
- centralizované řízení.

Decentralizované řízení řídí jednotlivé osy a řízením ve stejný čas je dosažena součinnost celého robotického řetězce. Na druhou stranu centralizované řízení řídí všechny osy

najednou. Centralizované řízení dosahuje větší přesnosti, neboť uvažuje dynamické vlastnosti mezi jednotlivými osami, nicméně pokud dojde k selhání jednoho z kloubů, přestává být možné řídit robota. Decentralizované řízení může být složitější, kvůli nutnosti řízení jednotlivých os, nicméně pokud dojde k výpadku kloubu, robotické rameno je stále funkční a zároveň může dosahovat větší flexibility než centralizovaný způsob řízení.

4.1. VOLBA MATERIÁLU

Materiál použitý ke konstrukci robotického ramena není jednotný. Pro základnu robotického ramena je vhodné použít pevný materiál i za cenu toho, že bude těžší, neboť se základna robotického ramena nijak nepohybuje a je nutné, aby snesla váhu celého ramena a zároveň objektů se kterými má robotické rameno manipulovat. Na tzv. link mezi klouby je vhodné použít správnou kombinaci pevnosti a lehkosti. Čím těžší budou pohyblivé části robotického ramena, o to menší bude maximální nosnost robotického ramena. Pro koncový manipulátor se pak volí co nejlehčí materiály, aby zároveň nedocházelo k nepřesnostem a robotické rameno mělo dostatečně velkou nosnost. Nejčastěji používané materiály pro konstrukci robotického ramena jsou:

- ocel,
- hliník,
- acetal/POM,
- guma/elastické materiály,
- kevlar,
- inovativní/chytré materiály.

Ocel je jedním z nejpoužívanější materiálů při konstrukci robotického ramena. Jde o materiál, který je velmi tvrdý a silný a zároveň je možné jednoduše zvýšit tvrdost oceli. Bod tavení oceli je 1539 °C, takže je velmi odolná vůči teplu a zároveň odolná vůči korozi. Ocel je tak ideálním materiálem pro drsné podmínky. Komponenty vyrobené z oceli zahrnují: ozubená kola, koncový efektor, komponenty motorů, rámy atd. Jednou z nevýhod oceli může být její obrobitelnost, která není triviální v případě komplikovanějších tvarů (Matthews, 2019).

Hlavní vlastností hliníku, pro kterou je tak oblíbeným materiálem je jeho lehkost. Jde opět o materiál odolný vůči teplu, jeho bod tání je 660 °C a zároveň některé slitiny hliníku mají velkou odolnost vůči korozi. Hliník je možné následně zpracovávat, a tak dodat slitinám, které nejsou tak odolné vůči korozi, lepší odolnost, nebo aplikovat povrchovou úpravu, díky které bude mít materiál nízké tření. Další výhodou hliníku je lepší obrobitelnost oproti oceli na

druhou stranu jde o materiál dražší než ocel. Z hliníku se vyrábí: rámy, kryty, kola, ložiska, konce robotických ramen atd. (Matthews, 2019).

V případě acetal/POM jde o materiál plastový. Díky tomu je velmi lehký a zároveň jde o materiál s nízkým třením. Bod POM je okolo 162 °C tím pádem není tak odolný vůči teplu jako ocel nebo hliník, ale jde o materiál levný v porovnání s ocelí nebo hliníkem. Nejběžněji se používá pro: kryty, rámy, pouzdra atd. (Protolabs, 2023).

Elastické materiály se používají hlavně v případě, kdy není zapotřebí chránit robotické rameno, ale jeho okolí, jako jsou lidi pohybující se v jeho okolí. Zároveň je výhoda využití tohoto materiálu v případě, kdy robot manipuluje s jemnými předměty jako je např. ovoce, pak se jedná o tzv soft robotics. Je možné je využít pro: rám, pouzdra, v případech, kdy je nutná roztažnost materiálu (Protolabs, 2023).

Kevlar se používá k vytvoření „obleku“ pro robota. Kevlar má velice dobrou tepelnou odolnost, což se používá pro ochranu robota ve vysokých teplotách. Jde o lehký materiál, který pokryje celé robotické rameno a nijak neovlivňuje pohyblivost nebo nosnost robotického ramena (Matthews, 2019).

Mezi inovativní materiály patří např.: kolagen, cellulose nebo polycaprolactone. Jde o chytré materiály, které dokáží měnit tvar v závislosti na podněty. Jde také o materiály schopné tzv. regenerace a tím jsou schopny prodloužit životnost robotického ramena. Existují materiály, u kterých dojde po splnění účelu k rozkladu. Biodegradable chytré materiály byli použity pro kůži robota a je možné, že budou dostatečně tvrdé na tvorbu vnitřních komponent robota (Matthews, 2019; Protolabs, 2023).

4.2. KLOUBY ROBOTICKÉHO RAMENA

Klouby robotického ramena se skládají z mechanických prvků, elektrických prvků a akčních členů. Mezi mechanické prvky se řadí:

- ložiska – Umožňují rotační pohyb, snižují tření, zvyšují tak životnost mechanických částí kloubu, které se pohybují a zároveň zvyšují přesnost.
- Ozubená kola – Umožňují převod, díky tomu lze zvýšit přesnost a případně převést pohyb z rotačního na lineární.
- Hřídel – Používá se k převodu rotačního pohybu na lineární.
- Kulatina – V případě lineárních kloubů se může používat jako dráha, po které se kloub pohybuje.

(Beyatli, 2024)

Mezi elektrické prvky patří:

- Motor – Slouží jako akční člen kloubu, zprostředkovává pohyb.
- Senzor – Slouží pro určení buď úhlu natočení kloubu, nebo určení pozice kloubu.

(Beyatli, 2024)

Motor je součástí kloubu v případě, kdy se ho rozhodneme použít jako akční člen, nicméně je možné využít:

- Hydraulické akční členy,
- Pneumatické akční členy.

Jednou z nejpopulárnějších voleb pro akční členy jsou elektrické akční členy, které využívají buď krokového nebo stejnosměrného motoru. Jsou tiché, přesné a zároveň je možné ovládat přesně jejich pohyb tzn. rychlost a směr pohybu a jsou lehce použitelné. Nevýhodou je možno přehřátí a větší cena než hydraulické nebo pneumatické akční členy (Beyatli, 2024).

Hydraulické akční členy používají kapaliny, která je pod tlakem. Hydraulické akční členy jsou využívány hlavně v technice, která se používá pro velké zátěže, neboť dokážou vyvinout velkou sílu. Jsou však prostornější a komplikovanější na údržbu, jelikož je nutné zajistit, aby se nenacházeli žádné nečistoty v kapalině a zároveň může dojít k úniku kapaliny a tyto úniky není jednoduché odhalit (Beyatli, 2024).

Pneumatické akční členy využívají stlačeného vzduchu. Je tedy zapotřebí kompresor, který bude dodávat stlačený vzduch, pro správnou funkčnost těchto akčních členů. Nevýhodou je menší síla v porovnání s elektrickými nebo hydraulickými akčními členy, zato dosahují velké rychlosti a přesnosti (Beyatli, 2024).

V případě elektrických akčních členů jde většinou o motory, které zprostředkovávají rotační pohyb, který je podle potřeby převeden na lineární, případně zredukován. V případě hydraulických a pneumatických se využívá stlačeného média většinou k zprostředkování lineárního pohybu. Lze využít hydraulických motorů, ale jejich účinnost je malá v porovnání s elektrickými motory.

U pneumatických a hydraulických akčních členů lze přesně regulovat pozici a rychlost kloubů nastavením tlaku, zatímco u elektrických členů se většinou používá redukce. Je to z toho důvodu, aby se dosáhlo větší přesnosti a zároveň převodem větších otáček na menší je dosaženo větší síly kloubu. Nejčastěji používané převodové poměry jsou 1:20 a 1:200.

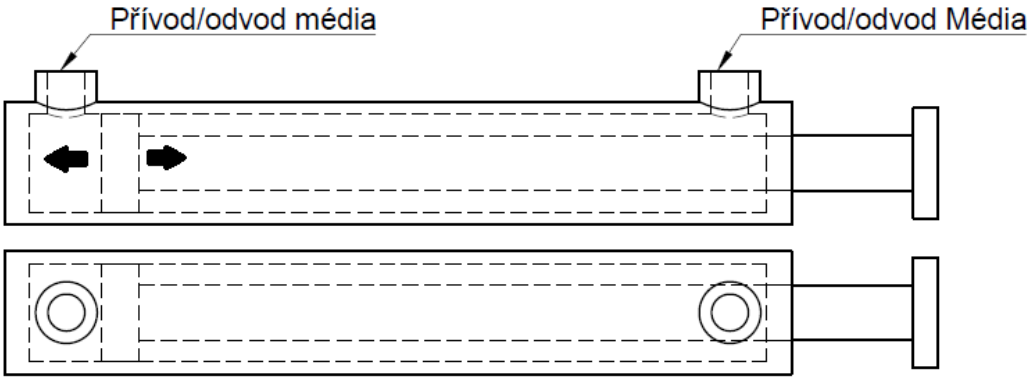
Klouby lze dělit pode typu pohybu na:

- Lineární,
- Rotační,

- Univerzální,
- Cylindrické,
- Planární.

4.2.1. Lineární kloub

Lineární kloub zprostředkovává lineární pohyb po ose. Skládá se ze dráhy, ve které se pohybuje pohyblivá část lineárního kloubu, zatímco dráha zůstává stacionární. V případě využití hydraulických nebo pneumatických akčních členů, dráha lineárního kloubu se skládá ze dvou komor, které jsou rozděleny pohyblivou částí. Podle připouštění stlačeného média do jedné nebo druhé komory, případně odpouštěním, dochází k snižování nebo zvyšování tlaku v komorách, a tak dochází k vyrovnání tlaku v komorách pohybem pohyblivé části. V tomto případě komory slouží zároveň jako dráha, ve které se pohybuje pohyblivá část kloubu. Zjednodušené principiální schéma lineárního akčního členu využívající pneumatiky/hydrauliky je vidět na Obr. 4.1 (Rao, 2023).

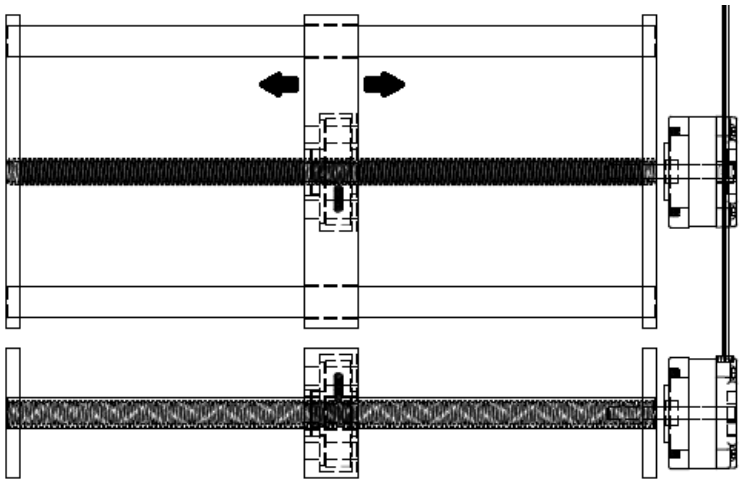


Obr. 4.1 – Pneumatického lineárního aktuátoru

V případě využití elektrického akčního členu je zapotřebí převést rotační pohyb elektrického motoru na lineární. To se provádí pomocí:

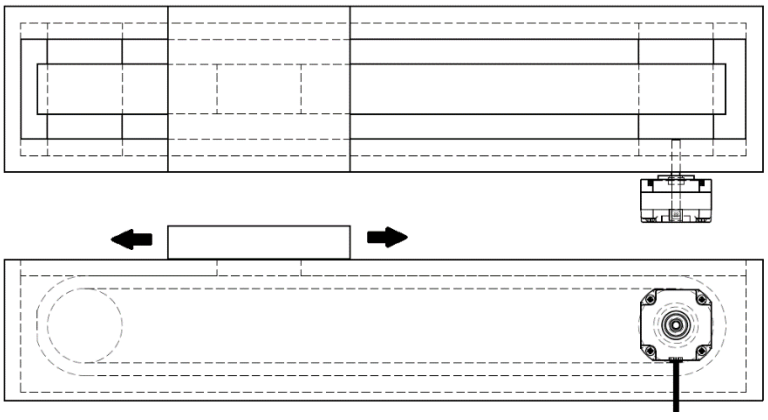
- vodicí šroub,
- kuličkový šroub,
- pás,
- hřeben a pastorek.

V případech vodicího a kuličkového šroubu je nutný šroub, který je upevněn k motoru a zprostředkovává rotační pohyb. Na šroub je připevněna pohyblivá část kloubu a zároveň ke dvou drahám na stranách tak aby nedocházelo k rotačnímu pohybu pohyblivé části. Tyto dráhy jsou většinou kulatiny, které potřebují mít malé tření, jelikož je s nimi pohyblivá část v kontaktu. Pohyblivá část pak buď obsahuje závit, nebo kuličky, které zajistí pohyb pohyblivé části vlivem rotačního pohybu šroubu.



Obr. 4.2 – Elektrický lineární aktuátoru (vodicí šroub)

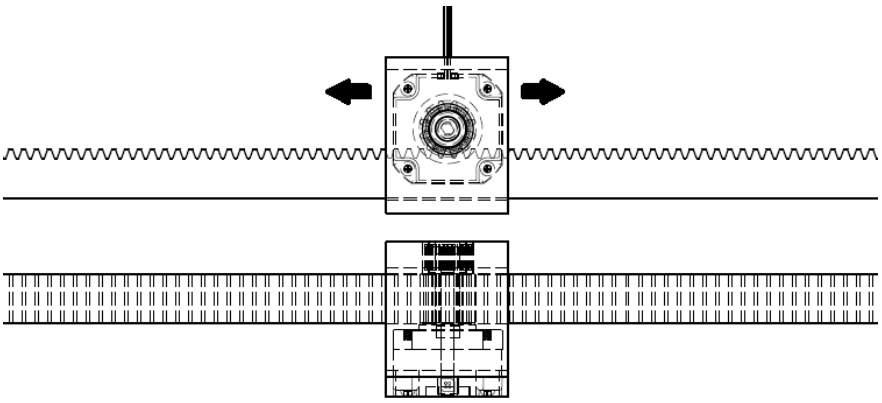
Lineárního pohybu je možné dosáhnout také pomocí pásu, který je upevněn na jedné straně pomocí ložiska a na druhé straně k motoru. Rotační pohyb motoru tak způsobí lineární pohyb pásu, ke kterému je připevněna pohyblivá část kloubu. Pohyblivá část je zároveň připevněna k dráze.



Obr. 4.3 – Elektrický lineární aktuátor (pás)

Při využití hřebene a pastorku je motor výjimečně součástí pohyblivé části kloubu. K motoru je připevněn pastorek (ozubené kolečko), které odvalováním se na hřebenu (ozubené

dráze) zajišťuje lineární pohyb, jelikož pohyblivá část je zároveň spojena s drahou, jejíž součástí je hřeben.



Obr. 4.4 – Elektrický lineární aktuátor (hřeben a pastorek)

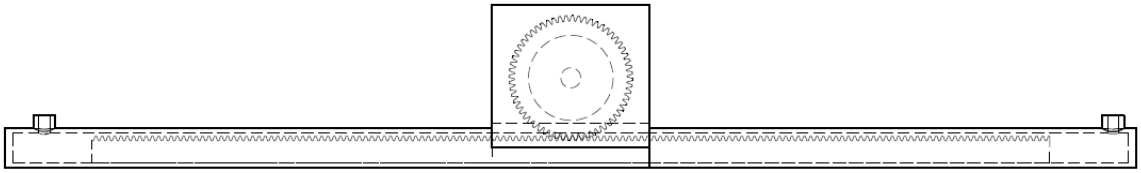
U vodicího a kuličkového šroubu je převod zajištěn sklonem závitu šroubu. Při menším sklonu závitu pak dochází k menší uražené dráze při jedné otáčce motoru a naopak. V případě použití pásu je nutné využít převodovky k převodu vysokých otáček. Redukovat otáčky v případě použití hřebene a pastorku lze velikostí pastoru. Čím menší pastorek, tím menší uražená dráha při jedné otáčce a naopak.

4.2.2. Rotační kloub

Pro rotační pohyb se převážně používá elektrických akčních členů, případně hydraulických, pneumatické se využívají převážně pro lineární akční členy. V případě využití hydraulického pohonu je nejvýhodnější využít lineární pohyb, který je převeden na rotační, neboť využití hydraulických motorů je sice možnost, ale jde o motory s malou účinností. Lineární pohyb hydraulického akčního členu je možné převést na rotační pomocí hřebenu a pastorku. V tomto případě je pastorek stacionární a dochází k pohybu hřebene, který se pohybuje v dráze, a tak zprostředkovává rotační pohyb pastorku. Pohyb hřebene v dráze funguje na stejném principu jako princip lineárního akčního členu zobrazeného na Obr. 4.4 Rotační pohyb je sice omezen, ale je možné dosáhnou velké síly a přesnosti (Rao, 2023).

Pro druhý způsob převodu, předpokládejme dvě části ramena spojeny spolu na koncích způsobem, umožňující jim rotační pohyb vůči sobě. Lineární hydraulický akční člen je upevněn stacionární částí k jedné části ramena a pohyblivou částí pomocí kulatiny vyvedené jednou

z komor k druhé části ramena. Při lineárním pohybu akčního členu pak dochází k rotačnímu pohybu dvou částí ramene vůči sobě. Pohyb je omezený ale je opět možné dosáhnou velké síly.

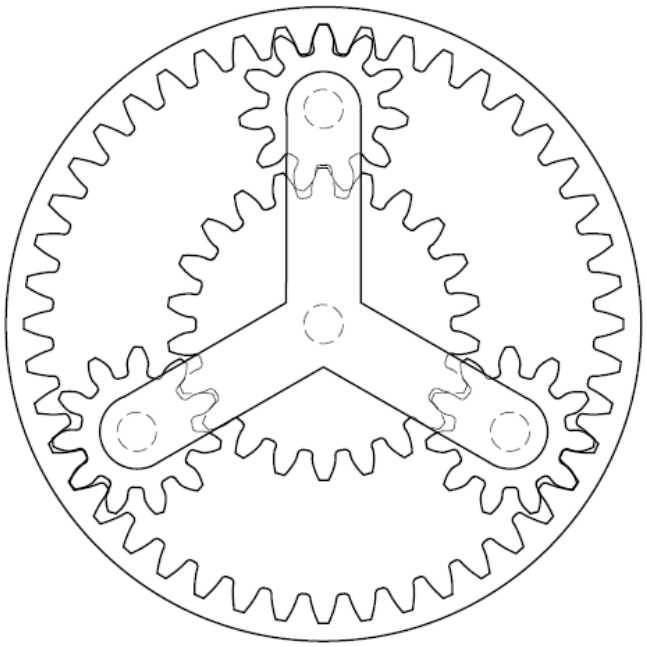


Obr. 4.5 – Pneumatický rotační aktuátor (hřeben a pastorek)

Elektrické akční členy zprostředkovávají výhradně pohyb rotační, takže není nutno jej dále převádět, nicméně se používají převodovky, které zajišťují převod z vysokých otáček na malé. Nejčastěji používané převodovky pro rotační kloub:

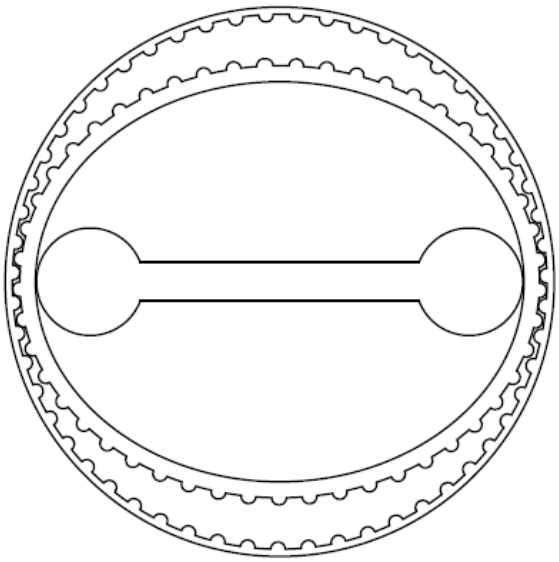
- planetové,
- harmonické,
- cykloidní.

Planetová převodovka se skládá z vnějšího ozubeného kola se zuby na vnitřní straně kola, ozubeného menšího kola, které je hnané motorem, jehož otáčky redukujeme a z ozubených koleček umístěných mezi hnaným kolem a vnějším kolem. Výstupní otáčky jsou otáčky ozubených koleček umístěných mezi velkým a hnacím kolem, které jsou spolu spojeny a tento moment je vyveden na hřídel.



Obr. 4.6 – Planetová převodovka

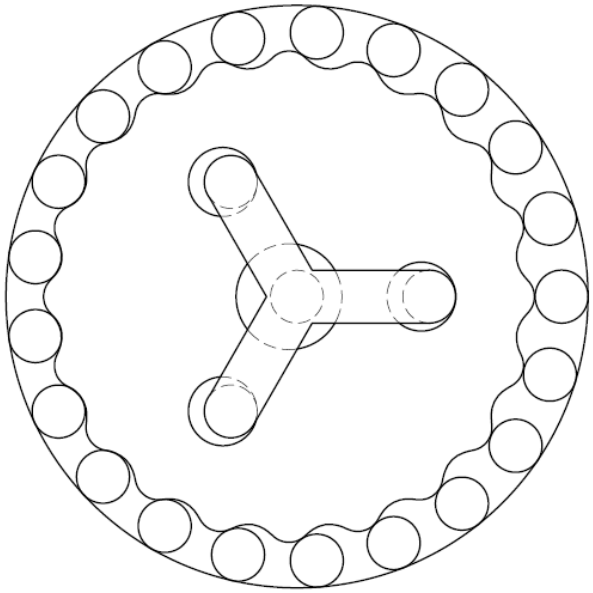
Harmonická převodovka se skládá z vnějšího ozubeného kola opět se zuby na vnitřní straně a z pružného prstence. Pružný prstenec je ozubený se zuby na vnější straně a s menším poloměrem, než je poloměr vnějšího kola a je přitlačován k vnějšímu kolu ve dvou bodech pomocí ložisek připevněných k motoru, jehož otáčky jsou převáděny. Dochází k deformaci pružného prstence a zároveň rotačním pohybem bodů, ve kterých je prstenec přitlačován dochází k rotaci prstence. Prstenec je pak spojen s hřídelí.



Obr. 4.7 – Harmonická převodovka

Cykloidní převodovka využívá cykloidního disku, tvar, který vytvoří pevný bod na malém kruhu, který se valí po velkém kole. Cykloidní disk je umístěn uvnitř kola se sloupky a je navržen tak, že umístěním do většího kola se buď zubem dotýká stěny, nebo se sloupek dotýká prostoru mezi zuby cykloidního disku. Uprostřed disku je kolečko spojeno s diskem pomocí ložiska a kolečko je spojeno s motorem, jehož otáčky jsou redukovány, ale hřídel motoru není uprostřed kolečka, nýbrž je posunutá o poloměr sloupků. Otáčením kolečka uprostřed cykloidního disku, vlivem otáček motoru, tak dochází k odvalování disku uprostřed

velkého kola. Disk obsahuje otvory pro sloupky, které jsou spojeny a vlivem valení cykloidního disku dochází k rotačnímu pohybu hřídele, která je připevněna k spojeným sloupků.



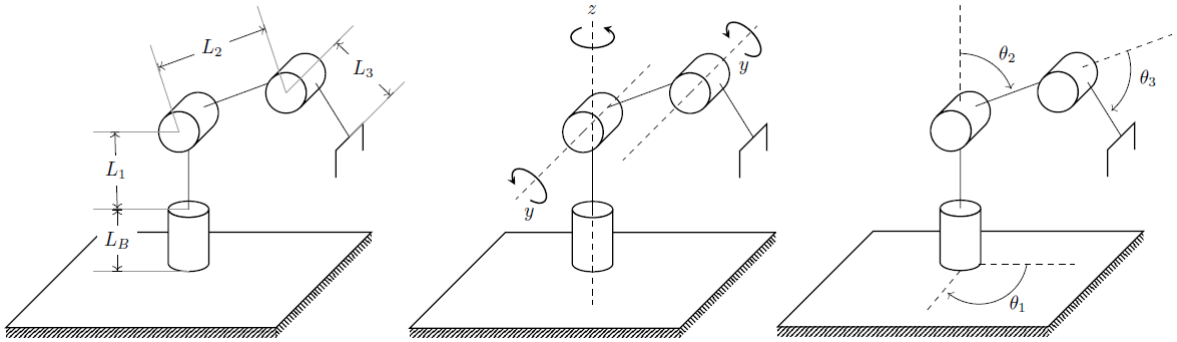
Obr. 4.8 – Cykloidní převodovka

Výhodou těchto převodovek je možnost kompaktního provedení, ve všech případech je možné využít vnější kolo, uprostřed kterého dochází k převodu, jako jedna část kloubu a rotační pohyb, který je zredukován je upevněn k druhé části kloubu, zatímco motor je připevněn k vnějšímu kolu.

5. ŘEŠENÍ KONSTRUKCE ROBOTICKÉHO RAMENA

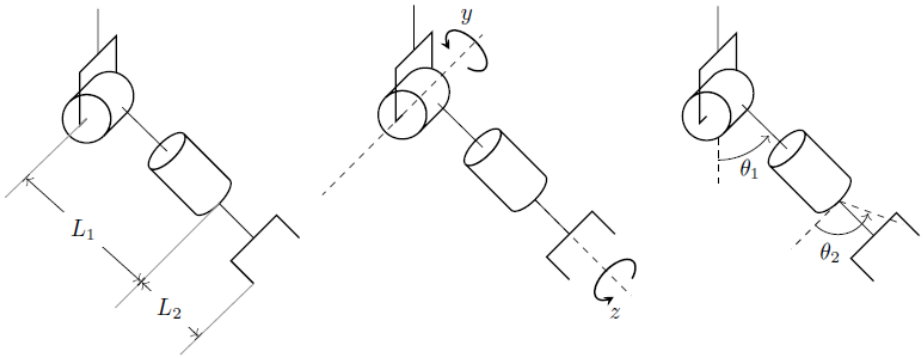
Zvolená architektura robotické ruky je vidět na Obr. 5.1 Jde o SCARA architekturu, která se nejvíce podobá lidskému rameni. Robotické rameno se skládá ze tří rotačních kloubů. Jeden rotující okolo osy Z a dva rotující okolo osy Y. Byla zvolena, jelikož nabízí poměrně velkou flexibilitu a pro vyhýbání se překážkám se zdá být vhodná. Robotické rameno disponuje třemi stupni volnosti.

Architektura zápěstí robotického ramena je vidět na Obr. 5.2. Jde o zápěstí se dvěma stupni volnosti. Skládá se ze dvou rotačních kloubů. Jeden rotující okolo osy Z a jeden rotující okolo osy Y.



Obr. 5.1 – Diagram robotického ramena

- L_B – 600 mm (výška základny)
- L1 – 180 mm
- L2 – 230 mm
- L3 – 140 mm



Obr. 5.2 – Diagram zápěstí robotického ramena

- L1 – 40 mm
- L2 – 70 mm

Celá tato struktura tak poskytuje 5 stupňů volnosti, které sice nepostačují na dosažení jakékoliv orientace koncového bodu, nicméně byl robot primárně navržen na to, aby byl schopen uchopit objekt, přenést ho, zatímco se pohybuje v jedné rovině, aby byl schopen vyhnout se překážkám a opět objekt položit, k čemuž je 5 stupňů volnosti naprosto dostačujících.

Tab. 5.1 - Seznam využitých prvků a materiálů

Materiál/prvek	Typ	Množství
NEMA 17 krokový motor	17H3401	3x
NEMA 17 krokový motor	17HS4401	1x
NEMA 17 krokový motor	17HS8401S	1x
Servomotor	SG90	1x
Lano z polyamidu	2 mm	~4000 mm
Teflonová trubička	4x1mm	~3200 mm
Vytištěné prvky	-	-
Šroub	M3	~73x
Šroub	M4	~20x
Matice	M3	~32x
Matice	M4	~20x
Ozubené kolečko	d = 25 mm	2x
Ložisko	6810	5x
Ložisko	6804	9x
Ložisko	6700	1x
Ložisko	6820	1x
Kulatina	4 mm	~35mm
Kulatina	2 mm	~10mm
Hliníková extruze	20 mm	380 mm

5.1. NÁVRH CYKLOIDNÍHO DISKU

Při konstrukci bylo využito cykloidní převodovky, nebo její modifikace, pro rotační klouby robotického ramena. Pro návrh cykloidního disku bylo využito skriptu v jazyce python, který byl implementován v programu Fusion 360, viz Obr. 5.3. Tento kousek pak lze zkopírovat okolo osy ve středu disku počtem redukčního poměru a výsledkem je cykloidní disk. Ve skriptu je vidět hodnota offset, která umožňuje posunout pevný bod, umístěný na malém kolečku, opisující dráhu, blíže ke středu kolečka. To zajistí, že sloupky, umístěné na vnějším kole, zapadnou do drážek disku. Na Obr. 5.3 je zvolen moc malý offset, a tak by nebylo možné cykloidní disk pro účely převodovky použít. Moc velký offset ovšem snižuje tuhost celé převodovky.

Obr. 5.3 – Segment cykloidního disku

```
sketches = rootComp.sketches

        xyRovina = rootComp.xYConstructionPlane
        sketch = sketches.add(xyRovina)

        polomer_sloupku = 0.8
        polomer_vnejsiho_kola = 8
        pocet_sloupku = 21
        offset = 0.05

        polomer_valiciho_kolecka = polomer_vnejsiho_kola / pocet_sloupku
        redukcni_pomer = pocet_sloupku - 1
        polomer_cykloidniho_disku = redukcni_pomer * polomer_valiciho_kolecka

        posledni_bod = None
        usecka = None

        usecky = []

        for angle in drange(0, 360 / redukcni_pomer, 0.1):
            x = (polomer_cykloidniho_disku + polomer_valiciho_kolecka) * cos(angle)
            y = (polomer_cykloidniho_disku + polomer_valiciho_kolecka) * sin(angle)

            bod_x = x + (polomer_valiciho_kolecka - offset) * cos(pocet_sloupku * angle)
            bod_y = y + (polomer_valiciho_kolecka - offset) * sin(pocet_sloupku * angle)

            if angle == 0:
                # the first bod
                posledni_bod = adsk.core.bod3D.create(bod_x, bod_y, 0)
            else:
                usecka = sketch.sketchCurves.sketchuseckas.addByTwobods(
                    posledni_bod, adsk.core.bod3D.create(bod_x, bod_y, 0)
                )
                posledni_bod = usecka.endSketchbod
                usecky.append(usecka)

            app.activeViewport.refresh()
```

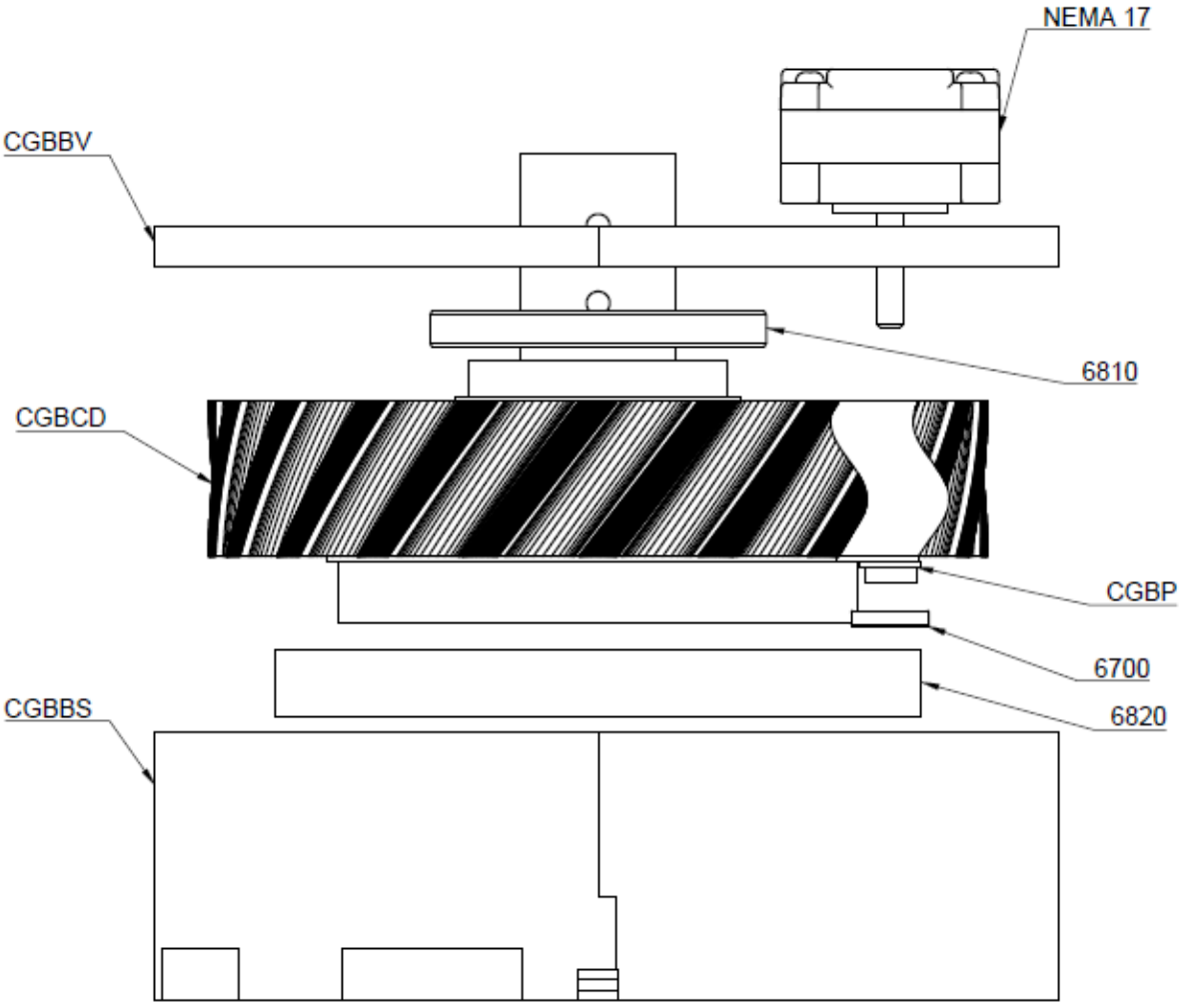
5.2. ZÁKLADNA RAMENA

Pro základnu ramene byla zvolena cykloidní převodovka, ale nejde o klasickou cykloidní převodovku. Je zhotoven cykloidní disk, který je pak extrudován kolem osy uprostřed disku, kolmé k němu. Úhel rotace extruze okolo osy je stanoven podle rovnice (5.1).

$$\frac{360}{n}, \tag{5.1}$$

Kde n – převodový poměr.

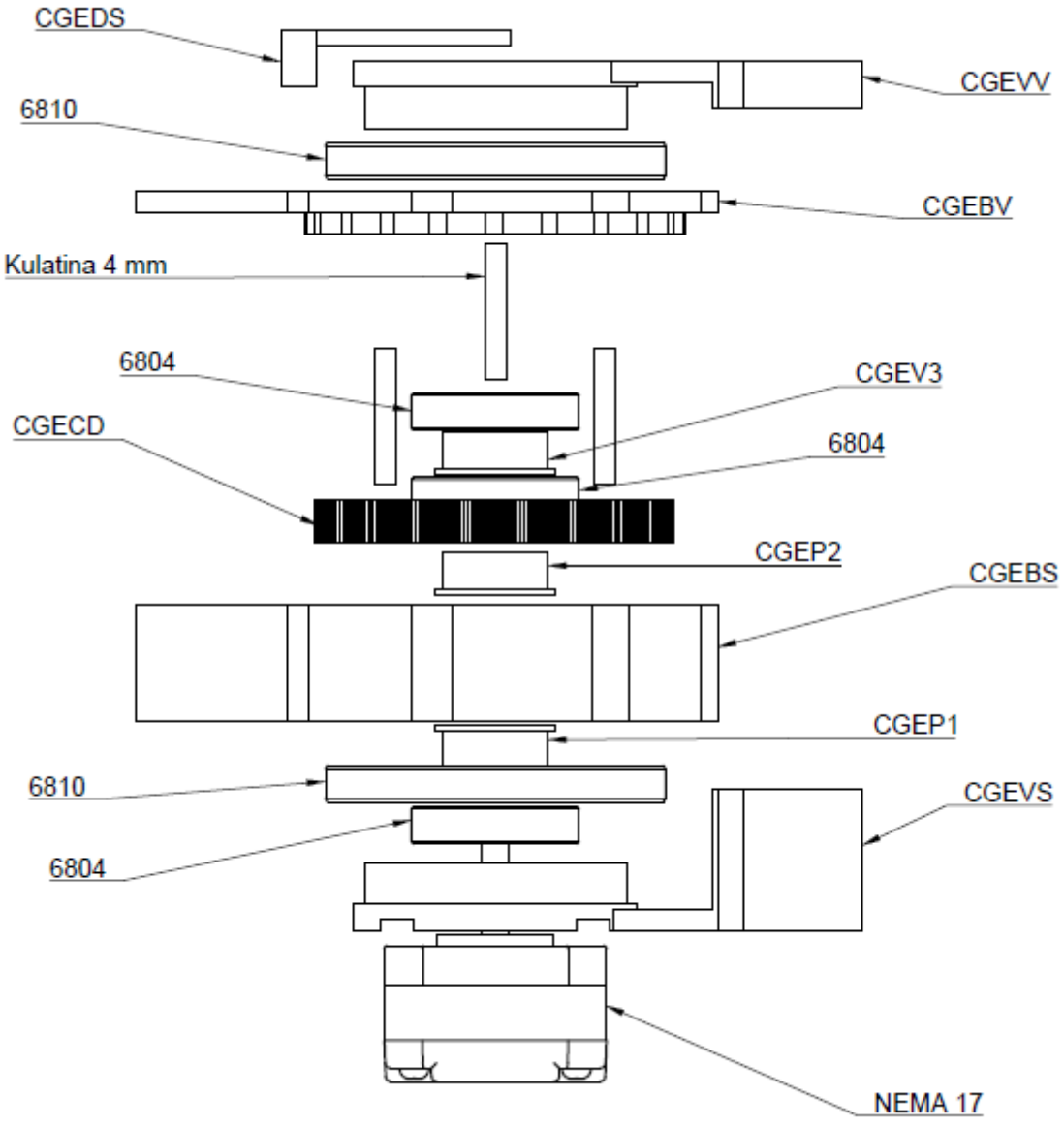
Převodovka pro základnu je dimenzována na převodový poměr 1:20. Tímto diskem je pak otáčeno z vně pomocí kolečka, které je v kontaktu s diskem. Poloměr je stanoven při návrhu cykloidního disku. V běžné cykloidní převodovce by plnilo funkci sloupku vnějšího kola. Kolečko je extrudováno okolo osy kolmé k němu stejně jako disk, jen s tím rozdílem, že osa je posunutá o půlku poloměru kolečka od středu. V případě, kdy je při návrhu cykloidního disku bod na kolečku, opisující větší kolečku, odsazen od okraje kolečka, je nutné tuto vzdálenost vzít k úvahu a osu o tuto vzdálenost posunout blíže ke středu. Kolečko je extrudováno okolo osy o úhel 360 °. Toto řešení bylo zvoleno, jelikož je možné navrhnou disk s dostatečně velkým průměrem, aby poskytoval robotickému ramenu dostatečně velkou stabilitu. Pro konstrukci základny byla zvolena 3D tiskárna. Rozměry základny museli být omezeny na maximální plochu tisku tiskárny a zároveň jsou omezeny volbou ložiska, jelikož ložiska větších průměrů jsou neobvyklé zboží, a proto je jejich cena znatelně větší. Základna využívá motoru typu 17H3401 a je připevněna na dřevotřískovou desku o rozměrech 630x550x180 mm doprostřed desky. Na základnu jsou připevněny čtyři sloupky z hliníkového profilu o výšce 50 mm, na kterých je položena dřevotřísková deska o rozměrech 630x550x120 mm.



Obr. 5.4 – Sestava základny

5.3. KLOUBY RAMENA

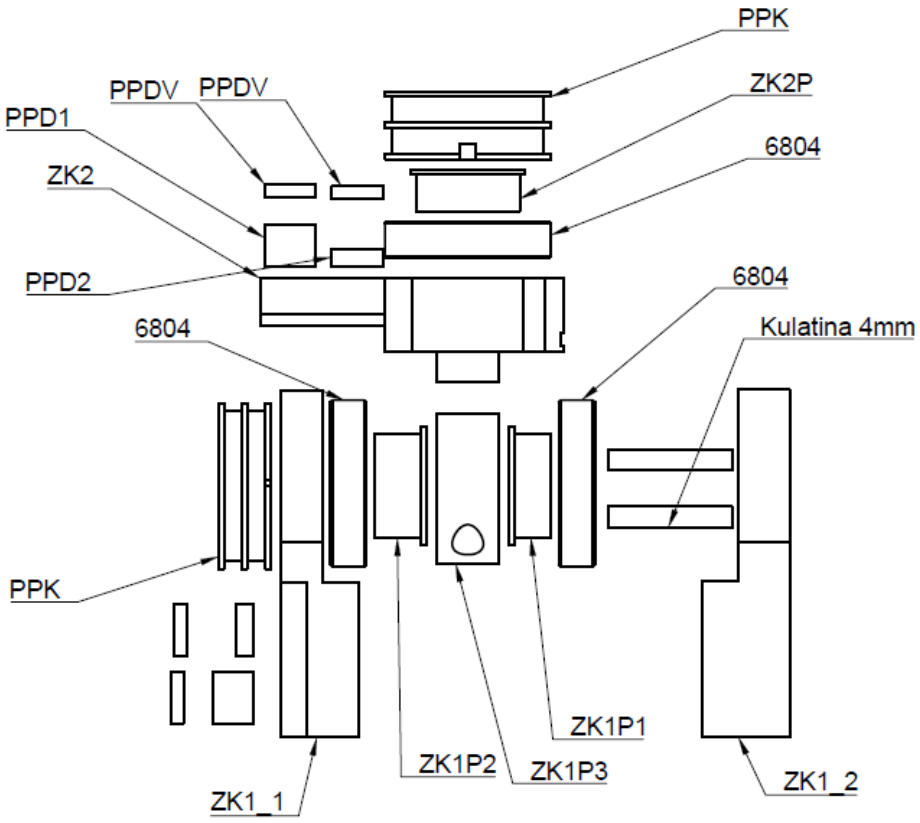
Pro klouby ramena byla využitá klasická cykloidní převodovka s převodovým poměrem 1:20. K návrhu byl opět využit skript v pythonu zobrazen v předchozí kapitole. V případě prvního kloubu, který je spojen se základnou, bylo nutno využít motor typu 17HS4401, aby byl zajištěn dostatečný moment, zatímco u druhého kloubu stačil motor typu 17H3401.



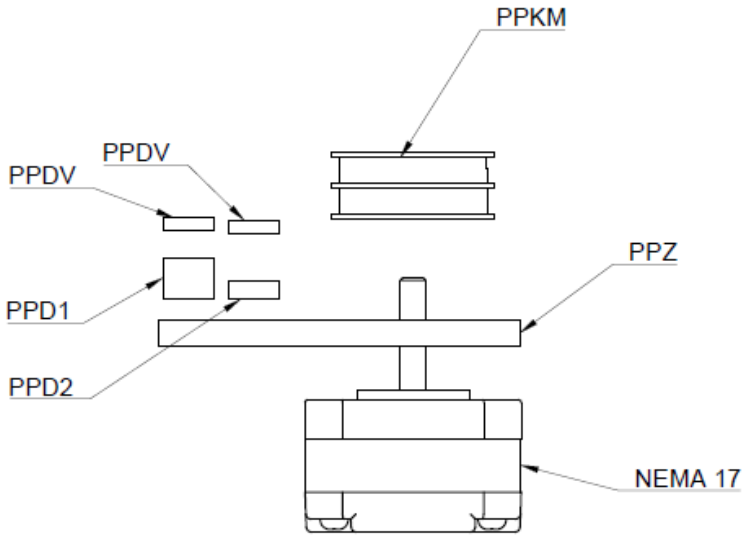
Obr. 5.5 – Sestava převodovky kloubu

5.4. ZÁPĚSTÍ ROBOTA

Aby bylo zápěstí robota co nejvíce odlehčeno, nebyli umístěny motory přímo na zápěstí, ale bylo využito podobného principu, na kterém pracuje bowden u jízdního kola. Rotační pohyb motoru je převeden pomocí lanka, prostrčeného trubičkou na zápěstí. Lanko je navinuto na kolečko umístěné na hřídeli motoru a na druhém konci na zápěstí. Hadičky jsou zasunuty do dílů PPD1, PPD2 a PPDV jak bude vidět ve výkresech. Tyto díly jsou umístěny a upevněny k dráze z důvodu montáže a potřebného napnutí lanka pro správnou funkčnost. Pro kloub, který hýbe zápěstím kolem osy Y byl použit motor typu 17HS8401S pro druhý kloub postačil motor typu 17H3401.



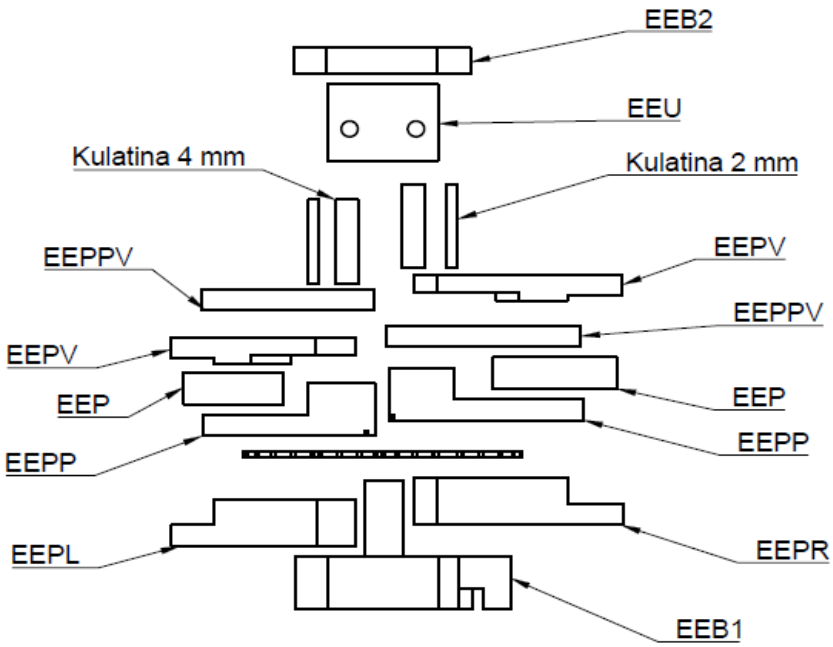
Obr. 5.6 – Sestava zápěstí robota



Obr. 5.7 – Sestava systému pro převod pohybu motoru

5.5. END-EFEKTOR

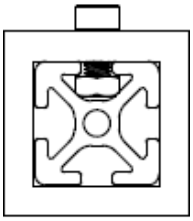
End-efektor je k zápěstí upevněn dvěma šrouby a využívá servomotoru SG90 k otevření nebo sevření. Pohyb jedné poloviny end-efektoru, která je ovládána servomotorem je převeden přes ozubená kolečka symetricky na druhou polovinu end-efektoru. Tímto způsobem dochází k otevírání/zavírání dvou pacek end-efektoru.



Obr. 5.8 – Sestava end-efektoru

5.6. SPOJENÍ JEDNOTLIVÝCH ČÁSTÍ

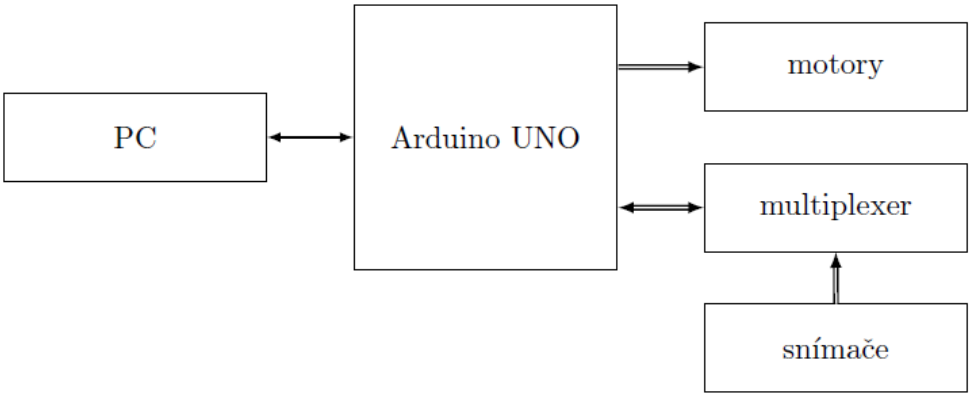
Jednotlivé částí, s výjimkou end-efektoru jsou spojeny pomocí hliníkového profilu, který je zasunut do obdélníkových otvorů na základně nebo kloubech.



Obr. 5.9 – Uchycení hliníkového profilu

6. ŘÍDICÍ SYSTÉM ROBOTICKÉHO RAMENA

Řídicím systémem je myšleno systém robotického ramena, který přijme požadovanou konfiguraci od PC a nastaví robotické rameno do požadované konfigurace. Tento systém je možné rozdělit na dvě části: hardwarovou část a softwarovou část.



Obr. 6.1 – Blokové schéma řídicího systému robota

6.1. HARDWAROVÁ ČÁST

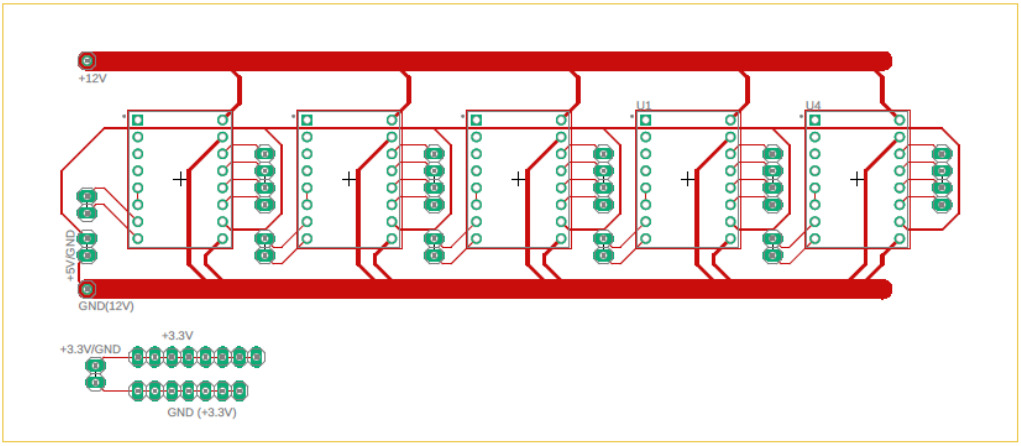
Pro řídicí část je nutné využít digitální I/O a I^2C sběrnici a sériovou komunikaci, a tak je Arduino UNO naprosto dostačující. Navíc poskytuje napětí +5 V a +3,3V, které jsou použity pro napájení použitých komponent. Nevýhodou tohoto řešení je potenciální rozšíření o 6. stupeň volnosti případně přidání redundantních kloubů, pro což by už Arduino UNO nestačilo. Arduino UNO je připevněno na stejné desce jako základna robota.

Robot má 5 stupňů volnosti, což znamená 5 motorů, v tomto případě krokových, a proto byla zvolena varianta řízení jednotlivých motorů pomocí integrovaného obvodu A4988. Toto řešení umožňuje ovládat motor pomocí pouze dvou pinů. Jeden pin určuje směr otáčení a na druhý pin je přiváděn impuls, což způsobí otočení motoru o jeden krok. K napájení motorů je použit zdroj napětí +12 V s maximálním proudem 5 A.

A4988 umožňuje jak řízení, tak pomocí tří pinů dále přizpůsobit krok motoru a pomocí potenciometru omezit maximální proud motorem. Přizpůsobení kroku využito není, jelikož je již využito převodovky, která poskytuje převodový poměr 1:20, nicméně omezit proud je nezbytné, aby nedošlo k přehřátí motoru. Podle použitého typu motoru se bude maximální proud lišit. Integrovaný obvod je umístěn na desce plošného spoje a je napájen 5 V dodávaných Arduinem.

Rotační enkodér byl zvolen AS5600. S enkodérem je možné komunikovat pomocí I^2C sběrnice. Problém je však to, že enkodér má jednu fixně danou adresu, což znemožňuje komunikaci s více enkodéry. To je vyřešeno pomocí multiplexeru TCA9548A, který umožňuje komunikaci s více enkodéry přepínáním jednotlivých senzorů na sběrnici. Enkodéry jsou pak adresované číslem vstupu, na který jsou připojeny, na multiplexeru. Enkodér umožňuje změnit směr snímání pohybu, což znamená směr, na kterou stranu se úhel zmenšuje nebo zvětšuje. Enkodéry jsou umístěny přímo na kloubech, nebo na zápěstí a na pohyblivých částech jsou umístěny magnety přímo nad enkodérem. Enkodér je napájen 3.3 V dodávaných Arduinem.

Na Obr. 6.2 je vidět návrh desky plošného spoje. Na levé straně desky je přívod napětí +12 V, jelikož integrovaný obvod bude používat napětí k napájení cívek motoru a přívod +5 V pro napájení samotného integrovaného obvodu. Na levé straně všech integrovaných obvodu je dvojice děr na připojení dvou pinů. Po pravé straně integrovaných obvodů je čtveřice děr, pro připojení konektoru pro kabel, kterým jsou připojeny motory k integrovaným obvodům. Ve spodní části desky je přivedeno napětí +3.3 V pro rozvod na jednotlivé enkodéry. Neodpovídá počtu enkodérů, jelikož některé jsou podle potřeby připojeny buď na GND nebo na +3.3 V pro případnou změnu směru snímání.



Obr. 6.2 – Deska plošného spoje řídicího systému roobota

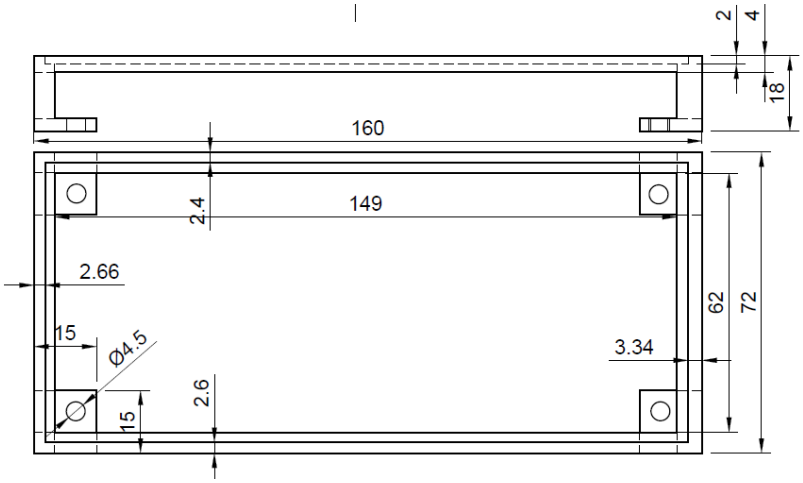
Tab. 6.1 - Seznam všech použitých komponent

Komponenta	Množství
Arduino UNO	1x
A4988	5x
AS5600	5x

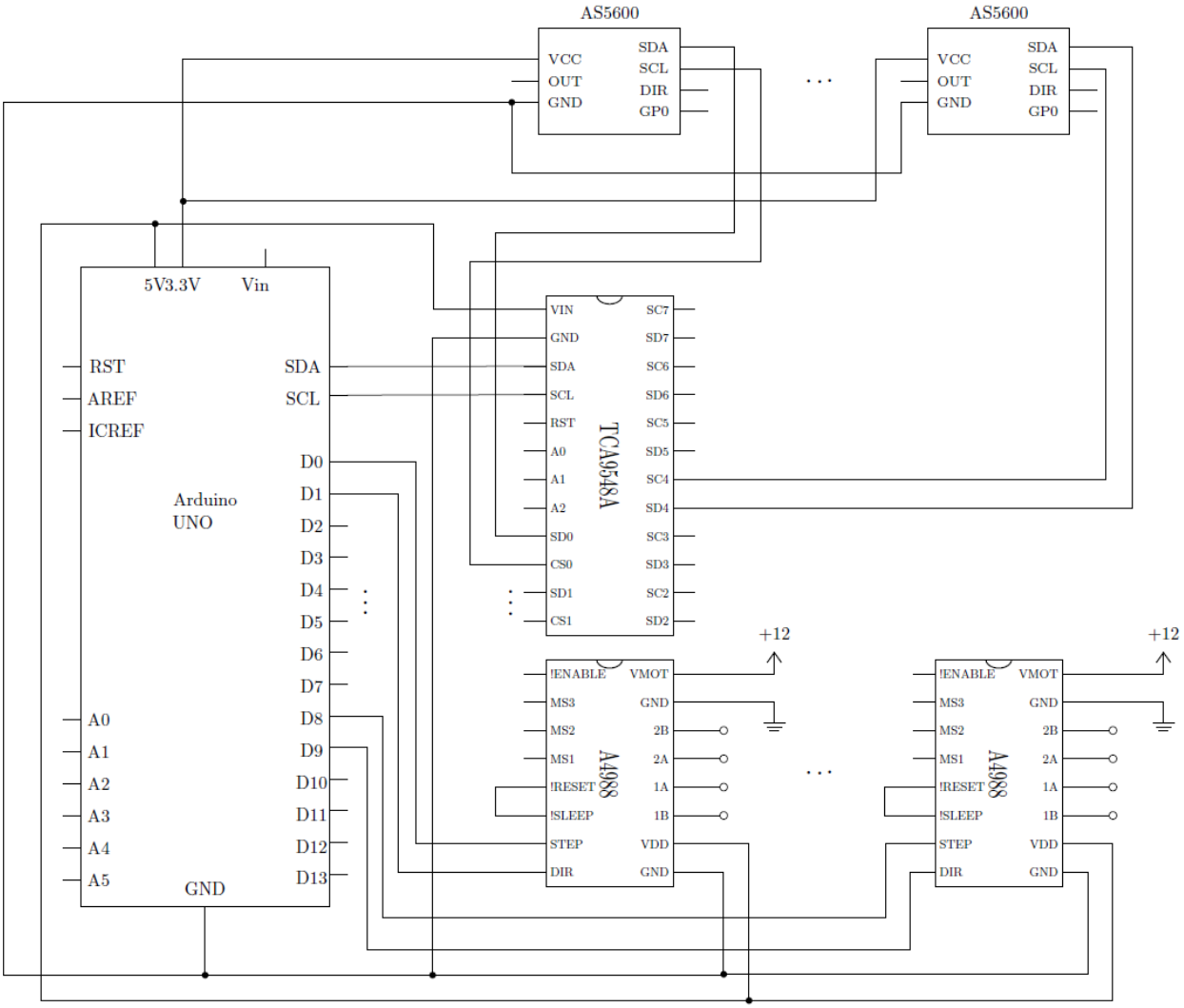
Tab. 6.1 - Seznam všech použitých komponent

TCA9548A	1x
JST-XH 2,5mm konektor	5x
Kolíková lišta	10 pinů
Kabel s konektory JST-XH-4 JST-XH -6	5x
Propojovací kabely	-
Rychlosvorky	-

Deska plošného spoje je položena na jednoduchém držáku připevněném na dřevotřískové desce, stejné jako základna robota.



Obr. 6.3 – Výkres držáku desky plošného spoje



Obr. 6.4 – Schéma zapojení řídicího systému robota

6.2. SOFTWAROVÁ ČÁST

Softwarová část se skládá z kódu Arduina. Je využito knihoven EEPROM.h, Wire.h a AS5600.h. Je využito dvou struktur:

- data,
- angles.

```
struct data {
  unsigned long timeMicros = 0;
  bool movingFlag;
};
```

Tato struktura je určená k uchovávání dat jednotlivých kloub. Uchovává informace o času, kdy došlo k provedení impulsu na příslušném pinu, aby bylo možné nastavit čas mezi impulzy. Dále uchovává informace o tom, zda se příslušný kloub hýbe.

```
struct angles {
  float angle1;
  float angle2;
  float angle3;
  float angle4;
  float angle5;
};
```

Struktura angles je vytvořena pro ukládání dat do EEPROM paměti. Je možné uložit celou strukturu, do které byli uloženy postupně všechny úhly robota a zároveň načíst celou strukturu a přímo přistupovat k jednotlivým úhlům. Ke čtení a zapisování jsou využity funkce put() a get() z knihovny EEPROM.

Dále je vytvořeno několik metod:

- clamp,
- PCAChanel,
- parseData,
- setDir,
- setWristDir,
- moveJoint,
- moveWrist.

Metoda clamp jednoduše omezí vstupní číslo n podle stanoveného minima nebo maxima a vrátí buď hodnotu n pokud je v rozmezí minima a maxima, maximum pokud je větší nebo minimum pokud je menší než stanovené meze.

Metoda PCAChanel je určená k přepínání enkodéru, se který je právě komunikováno. Vstupem metody je číslo od nuly do sedmi, neboť je možno přepínat mezi až 8 sériovými zařízeními připojenými na multiplexer.

```
void PCAChanel(uint8_t i) {
  Wire.beginTransmission(0x70);
  Wire.write(1 << i);
  Wire.endTransmission();
}
```

Metoda parseData je použita pro zpracování vstupních dat z počítače. Arduino přijme data ve formátu řetězce, který je dále potřeba zpracovat: převést řetězec na datový typ float, přiřadit hodnoty do správných proměnných. Metoda využívá metody clamp pro omezení hodnot, tak aby nedošlo k nastavení úhlů, které nelze dosáhnout, nebo by zapříčinili kolizi. Tímto způsobem jsou nastaveny požadované úhly jednotlivých kloubů a rychlosti pohybu pro jednotlivé klouby ve formě rychlosti otáčení motorů. Metoda pomocí funkce strtok rozdělí řetězec na řetězec po čárku a zbytek řetězce. První řetězec odpovídá číslu úhlu nebo rychlosti kloubu, v závislosti na pořadí, které je převedeno na datový typ float a uloženo do proměnné a zbytek řetězce je dále zpracováván, dokud nedojde metoda ke koncovému znaku, který je: \n.

```
char * strtokIndx;
strtokIndx = strtok(tempChars, ",");
desiredAngleBase = clamp(atof(strtokIndx), angleBaseMin, angleBaseMax);
```

```
96.06,131.83,119.99,236,173,2500,2500,2500,20000,20000,
```

Metoda setDir nastavuje směr pohybu kloubu pro klouby ramena, nikoliv zápěstí. Metoda nastaví příslušný pin pro daný kloub robotického ramena podle požadovaného úhlu a aktuálního úhlu.

Metoda setWristDir je použita pro nastavování směru zápěstí z toho důvodu, že došlo ke změně směru měření enkodérů. Jediný rozdíl je ten, že nastavuje opačný směr než metoda setDir.

Metoda moveJoint rozhoduje, zda se má příslušný kloub pohybovat a případně zprostředkovává impulz na příslušný pin, dokud nedojde k nastavení úhlu kloubu na žádaný úhel. Opět jde pouze o nastavení kloubů robotického ramene, ne kloubů zápěstí. Pro klouby robotického ramena je využita struktura data, aby bylo možné monitorovat, zda se rameno hýbe nebo ne.

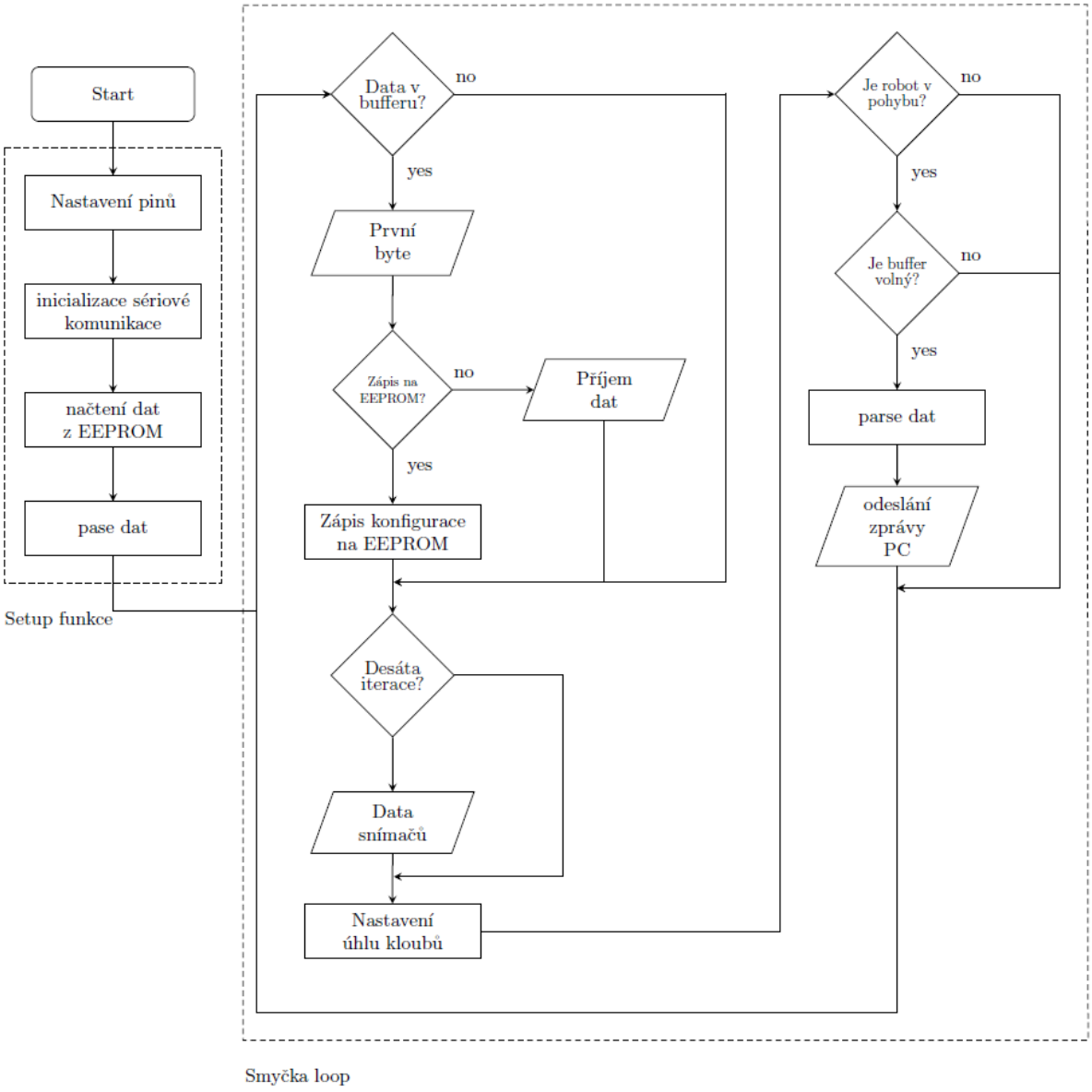
Metoda moveWrist zajišťuje pohyb pro zápěstí. Principiálně je metoda stejná jako moveJoin, jen s tím rozdílem že vrací pouze čas, kdy došlo k provedení kroku.

V setup funkci pak dojde k inicializaci pinů jako výstupní piny a k inicializaci sériové komunikace. Dojde k načtení hodnot z paměti EEPROM, pokud byli nějaké hodnoty uloženy v EEPROM paměti a dojde k přiřazení těchto hodnot do proměnných.

Loop smyčka nejdříve zkontroluje, zda jsou na vstupu (v bufferu) nějaká data a pokud jsou zkontroluje, zda se jedná o znak '#', který je použit pro signalizaci Arduinu pro zápis konfigurace robota do EEPROM paměti. Pokud se o znak nejedná a jsou data k dispozici, začne přijímat data ve vstupním buffer do pole deklarovaném v kódu. Při každé iteraci loop smyčky dojde k načtení jednoho bytu ze vstupního bufferu. Následně se provádí čtení hodnot ze senzorů, nicméně každou 10 iteraci, jelikož čtení zabírá relativně dlouhou dobu a znemožňuje to přesné nastavení rychlosti motoru. To není možné i tak, ale je možné nastavení větší rychlosti. Dojde k zaznamenání aktuálního času, aby bylo možné porovnat čas s časem kdy byl proveden poslední impulz a bylo možné nastavovat dobu mezi jednotlivými impulzy, a tak nastavovat rychlost motoru. Následně jsou nastaveny směry otáčení a jsou případně vyslány impulzy na jednotlivé piny motorům. Nakonec je zkontrolováno, zda se robot hýbe a zda je pole pro příjem dat volné, pokud ano dojde k zpracování dat, je umožněno načíst další data do pole, po sériové komunikaci je odeslán znak '1'. Příchozí data jsou tak uchovávána buď v poli, pokud je volné, nebo ve vstupním bufferu.

6.3. ŘÍZENÍ ZÁPĚSTÍ ROBOTA

Řízení zápěstí robota je prováděno pomocí Raspbery pi. Je to z toho důvodu, aby bylo možné případně připojit k Arduinu v budoucnu případně více kloubů a zároveň se vyskytla chyba při pokusu řídit zápěstí Arduinem, kterou se nepodařilo odstranit. Motor zápěstí je připojen na piny Raspbery pi poskytující napájení a na GIO pin pro ovládání zavírání a otevírání zápěstí. Raspbery pi přijímá řetězec, který odpovídá jedné ze dvou možností a Raspbery pi reaguje buď otevřením nebo zavřením zápěstí.



Obr. 6.5 – Vývojový diagram programu řídicího systému robota

7. DETEKCE PŘEKÁŽEK

7.1. YOLOv8

Pro detekci překážek byl zvolen algoritmus YOLOv8. Algoritmus byl zvolen, jelikož je možné jej vytrénovat a tím pádem přizpůsobit překážkám, které se mohou vyskytovat v pracovním prostoru robota. Jde zároveň o algoritmus velmi rychlý, díky čemuž může být využit pro detekování objektů v reálném čase a je taky testován pro toto použití. Algoritmus se neustále vyvíjí a existuje už verze YOLOv9.

K využití YOLO algoritmu byl použit programovací jazyk python a knihovna ultralytics. Algoritmus je načten pomocí funkce YOLO. Jelikož využívá algoritmus předtrénovaného modelu, je nutné disponovat tímto modelem, nicméně pokud je použita funkce YOLO, aniž by byl model v pracovní složce programu, je automaticky stažen.

```
model = YOLO("yolov8n.pt", "v8")
```

Pro samotnou detekci je využito funkce predict, která umožňuje nastavení parametrů detekce jako je např. zdrojový obrázek, nebo zvolit si hranici jistoty, což znamená že funkce vrátí jen ty boxy, které ohraničují objekty, u kterých je pravděpodobnost výskytu podle predikce modelu větší než stanovená hranice. Tato funkce navrátí datový typ zvaný result, který má několik objektů, ke kterým lze přistupovat tečkovou konvencí. Z těchto objektů je nejdůležitější boxes, neboť uchovává informace o boxech jednotlivých detekovaných objektů.

```
detect_params = model.predict(source=[color_image], conf=0.45, save=False)
```

K jednotlivým proměnným objektu boxes je opět možné přistupovat tečkovou konvencí a jde o vlastnosti jako: souřadnice, ID, jistota, typ objektu. V případě souřadnic je možnost přistoupit k souřadnicím ve dvou formátech: xyxy, xywh. První formát navrátí souřadnice rohů boxu ve formátu x a y souřadnice horního pravého rohu a x a y souřadnice spodního levého rohu, zatímco druhý formát navrátí souřadnice horního pravého rohu a výšku a šířku boxu. Pro následné zpracování je využit formát xyxy.

7.2. KAMERA

Detekce využívá kamery Intel Realsense D456. Jde o hloubkovou kameru používající stereovidění pro určení hloubky. Je možné využít přímo software od Intel pro práci s kamerou,

nicméně není možné extrahovat informace, tak spíše slouží jako test a ověření funkčnosti a případně správnosti extrakce/výpočtu dat. Pro použití kamery byl zvolen opět programovací jazyk python s využitím knihovny pyrealsense.

Je nejprve nutné nastavit kameru a zahájit tok, následně spustit rouru, přes kterou obdržíme jak hloubkovou mapu, tak i barevný obrázek.

```
config = rs.config()

config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
pipe.start(config)
```

Knihovna pyrealsense umožňuje získat informace o parametrech kamery tzv. intrinsics pomocí funkce get_intrinsics, které jsou dále využity ke zpracování hloubkové mapy k získání reálných souřadnic.

```
intr = profile.get_stream(rs.stream.color).as_video_stream_profile().get_intrinsics()
```

Knihovna pyrealsense má zabudované funkce pro filtrování hloubkové mapy. Je nutné nejdříve vytvořit filtr, nastavit jeho možnosti a nakonec použít na zvolenou hloubkovou mapu.

```
spatial = rs.spatial_filter()

spatial.set_option(rs.option.filter_magnitude, 5)
depth_frame_filtered = spatial.process(depth_frame)
```

Knihovna obsahuje:

- Decimation filter,
- Spatial edge-preserving filter,
- Temporal filter,
- Holes filling filter.

Pro pořízení obrázku z kamery:

7.3. DETEKCE A ZPRACOVÁNÍ

Samotná detekce probíhá pomocí YOLO algoritmu na barevném obrázku a hloubková mapa je využita pro určení přesné polohy objektu. Poloha objektu je vztažena ke středu kamery, s mírným posunem, jelikož jde o střed RGB kamery, která není přímo ve středu. Detekce má nastaven maximální čas deseti sekund detekce a pokud do té doby není detekován žádný objekt algoritmus je ukončen bez detekce objektu. Pokud objekt detekován je, případně více objektů, jsou zjištěny boxy ohraničující tyto objekty a je použita hodnota hloubky v boxu, která

odpovídá největší výšce objektu, pro stanovení souřadnic v reálném světě. Informace jsou dále převedeny a zpracovány strukturou prostředí o které bude zmíněno dále.

```
frames = pipe.wait_for_frames()

color_frame = aligned_frames.get_color_frame()
depth_frame = aligned_frames.get_depth_frame()
```

K převodu do reálných souřadnic je použito rovnic (7.1) a (7.2):

$$x = d * \frac{(px - ppx)}{fx}, \tag{7.1}$$

$$y = -d * \frac{(py - ppy)}{fy}, \tag{7.2}$$

Kde d – je hloubka pixelu,

px/pz – jsou x a y souřadnice pixelu,

ppx/ppy – je x a y souřadnice středu promítání obrazu,

fx/fy – ohnisková vzdálenost,

x/y – reálné souřadnice.

8. NÁVRH TRAJEKTORIE

Pro návrh trajektorie byl použit algoritmus DQN. Je to z toho důvodu, že je možné použít algoritmus jak pro naplánování celé trajektorie a pak začít s pohybem robota, nebo pomocí algoritmu reagovat na vnější vlivy. Byla zvolena varianta návrhu celé trajektorie a robot tak není schopen reagovat na vnější vlivy. Je to tak hlavně z důvodu umístění kamery. Jelikož je umístěna nad robotem, tak aby bylo vidět celá pracovní plocha robota. Pokud by bylo vyžadována reakce na vnější vlivy, bylo by nutné snímat pracovní prostor více kamerami nebo jiným způsobem.

8.1. TRÉNOVÁNÍ DQN

DQN vyžaduje k správné funkčnosti vytrénovanou neuronovou síť, kterou pak používá k rozhodování o akcích. K vytrénování neuronové sítě bylo vytvořeno prostředí a tzv. agent, opět v programovacím jazyce python. Prostředí i agent byli vytvořeny jako struktury, které jsou následně použity pro trénovaní neuronové sítě a taky pro návrh trajektorie.

Struktura prostředí reprezentuje pracovní prostředí robota, nebo spíše jednu plochu pracovního prostředí robota, jako mřížku, kde každá buňka je buď okupovaná end-efektorem robota, překážkou nebo je volná. Buňky mřížky reprezentují délku 2 cm v reálných souřadnicích. To znamená, že když se pohne end-efektor z jedné buňky do druhé pohne se o 2 cm jedním ze čtyř směrů: nahoru, dolu, doprava, doleva. Struktura prostředí uchovává informace o počáteční pozici, koncové pozici (cíl), aktuální pozici, pozici překážek a další pomocné informace a definuje metody:

- Reset,
- Done,
- Reward_move,
- Get_state,

mimo jiných pomocných metod. Tyto metody jsou hlavními metodami pro správnou funkčnost a interakci agenta s prostředím.

Metoda reset jednoduše navrátí stav prostředí do výchozího stavu. Většinou to znamená že pouze navrátí pozici do počáteční pozice, ale vygeneruje novou počáteční a koncovou pozici. Pro lepší trénování byla metoda upravena tak, aby vygenerovala buď počáteční a koncovou pozici, nebo vygenerovala i pozici předdefinované překážky. V případě, kdy dojde ke generování prostředí s překážkou, jsou tři možnosti, jak může dojít ke generaci počátečního

a koncového bodu. Buď jsou body vygenerovány horizontálně nebo vertikálně naproti sobě odděleny překážkou, nebo jsou vygenerovány diagonálně s překážkou umístěnou náhodně v prostoru. K této modifikaci došlo pro zlepšení kvality trénování sítě.

Prostředí má většinou nějaký koncový stav, při kterém dochází k resetu. To může být i stav, který je nežádoucí, jako je např. kolize s překážkou. O tom, zda takový stav nastal právě informuje metoda done, která vrací proměnnou typu bool, která buď odpovídá stavu, který není koncovým stavem, nebo stavu, který je. V tomto případě dojde ke koncovému stavu, pokud dojde ke kolizi s překážkou nebo se robot dostane přes hranice stanovené prostředím. Koncovým stavem je ovšem i stav, kdy došlo k dosažení cíle, takže k dosažení koncového bodu.

Metoda reward_move provede jednu z akcí, kterou lze provést, a vrátí hodnotu, kterou byla tato akce odměněna. Akce jsou odměněny podle toho, co je naším cílem. To znamená, že pokud se nějakým způsobem přibližujeme k cíli, odměna je větší, pokud se oddalujeme odměna je menší. Odměna byla zvolena za přibližování se k cíli jako dva body, za oddalování se od cíle jako mínus dva bod, za to že překročil hranice nebo došlo ke kolizi mínus deset bodů a za dostavení se do koncového bodu deset bodů. Dále pro ospravedlnění obcházení překážky kdy dochází ke vzdalování od koncového bodu, pokud nedochází k pohybu do bodů, kde se nacházel v předchozích šesti krocích je odměněn jedním bodem.

Aby se mohl agent správně rozhodnout o další akce, potřebuje informace o daném prostředí a stavu, ve kterém se zrovna nachází. Tento stav vrací metoda get_state. Informace o stavu, které jsou předávány metodou:

- Aktuální pozice,
- Koncová pozice,
- Vzdálenost od překážky/hranice prostředí,
- Zdali se nachází překážka vedle aktuální pozice.

Je žádoucí předávat stručné ale výstižné informace o stavu, aby nedošlo k předání redundantních informací, a zároveň dojde k zrychlení trénování, může ovšem s menším množstvím informací dojít k zhoršení kvality trénování, a tak je nutné informace o stavu správně vyvážit.

Struktura agent představuje ve své podstatě někoho, kdo rozhoduje o akcích v prostředí a v závislosti na jejich důsledku se učí a snaží se optimalizovat chování v prostředí pro efektivní dosažení cíle. Struktura agenta uchovává informace o aktuální hodnotě epsilon. Agent provádí akce buď naprosto náhodně, nebo využívá neuronové sítě, která je buď

vytvořena nová při začátku učení, nebo je použita již vytrénovaná pro zlepšení. Epsilon určuje pravděpodobnost zvolení akce buď náhodně, nebo neuronovou sítí. Metoda dále uchovává v paměti stav, akci, nadcházející stav, odměnu, hodnotu done. Tyto hodnoty z paměti pak používá pro trénování neuronové sítě. Struktura definuje metody:

- Choose_action,
- Remember,
- Replay,

a další pomocné metody.

Metoda Choose_action nejdříve rozhodne, zda bude akce vybraná náhodně nebo pomocí neuronové sítě. Šance vybrat náhodnou akci začíná na 100 % a postupně klesá. Je to z toho důvodu, aby došlo k prozkoumávání prostředí a zjištění jaká je reakce na příslušné akce v různých stavech. Postupně se šance snižuje, dokud nedojde na přednastavenou hodnotu, na které se zastaví. Tato hodnot není nulová, aby docházelo neustále k učení agenta.

Jak již bylo zmíněno agent uchovává v paměti informace o akcích odměnách a stavech a k ukládání těchto informací slouží metoda remember.

Metoda replay nejdříve vybere dávku informací z paměti, kterou použije pro učení neuronové sítě, následně zpracuje informace, a to tím způsobem že vytvoří dataset, který se skládá ze stavů a správné akce v daném stavu tzv. target.

```
targets = rewards + self.gamma np.amax(self.model.predict(next_states), axis=1))
```

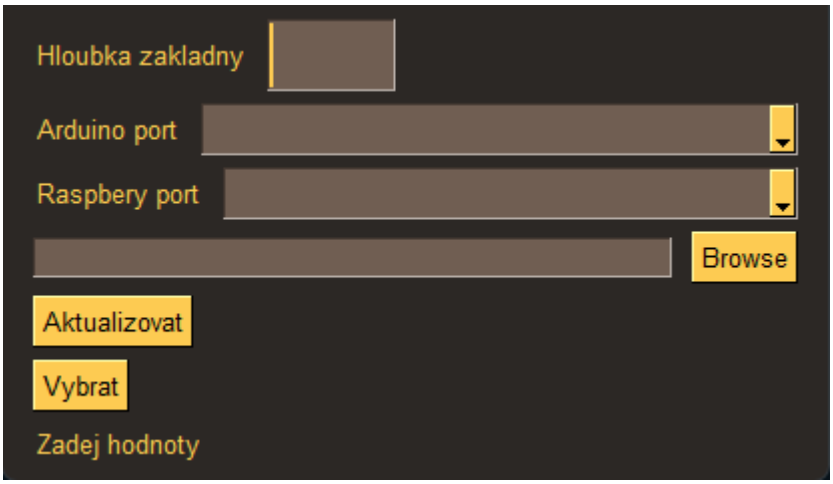
Následně je agent umístěn do prostředí a v cyklu se stanoveným počtem epoch provádí akce v prostředí a pokud se dostane do koncového stavu je zavolána metoda replay, prostředí je resetováno a začne nová epocha.

Pro samotný návrh trajektorie je nezbytné zvolit epsilon nulové, aby nedocházelo k náhodným akcím. Principiálně jde o jednu epochu, kde se agent, který již využívá vytrénovanou neuronovou síť, snaží najít optimální řešení k cíli. Je využito prostředí, které uchovává informace o detekovaných překážkách.

9. KONTROLNÍ SYSTÉM ROBOTICKÉHO RAMENA

Kontrolním systémem je v tomto případě myšleno systém, který poskytuje možnost uživateli kontrolovat robota. To znamená zadávat požadované souřadnice, detekovat objekty, navrhnout trajektorii, ovládat zápěstí atd. Toto prostředí je opět implementováno pomocí programovacího jazyka python a je využito knihovny pysimplegui k vytvoření jednoduchého grafického prostředí.

Po spuštění programu je otevřeno okno pro nastavení hloubky základny, která je potřeba ke správnému zobrazování a počítání polohy překážek, dále volba portů Arduina a Raspbery a cesta k modelu, který je použit pro plánování trajektorie. Toto nastavení lze otevřít při běhu programu a změnit tak hloubku, pokud se změnila poloha kamery, porty nebo zvolit jiný model pro plánování trajektorie při běhu programu. Nastavení je při ukončení programu uloženo v souboru a při následném spuštění programu je automaticky vyplněno, ne však použito. Po výběru je okno zavřeno a následně dojde ke spuštění samotného programu.



Obr. 9.1 – Grafické rozhraní nastavení konstant

Grafické prostředí se skládá z několika oken. První okno obsahuje prvky pro kontrolu robota. Dále z okna, kde je zobrazován barevný obraz pracovní plochy a okno, které zobrazuje hloubkovou mapu pracovní plochy, která není filtrovaná. Při zvolení jednoho z režimů dojde ke skrytí prvků pro ovládání robota v ostatních režimech.

Obr. 9.2 – Grafické rozhraní kontrolního systému robota

Program pracuje v jednom ze čtyř režimů:

- Měření hloubky,
- Manuální nastavení,
- Plánování trajektorie,
- Sken.

Měření hloubky zobrazuje v okně barevného obrazu pracovní plochy na místě kurzoru hloubku. Program ukazuje hloubku na souřadnicích kurzoru vyfiltrované hloubkové mapy.

Manuální nastavení umožňuje nastavit robot na požadovanou pozici, a to dvěma způsoby. Jedním z nich je nastavení dvojitým kliknutím na barevný obraz tam kam chceme robot nastavit. Na pozici kurzoru jsou zobrazovány reálné souřadnice x a y a výška, do které je

robot nastaven je daná, nicméně je možné ji změnit pomocí set height. Nastavit robota na požadované souřadnice je možné přímo zadáním souřadit do textového pole. Všechny souřadnice jsou zadávány v metrech. Je možné, pomocí clean view, nastavit do přednastavené polohy, která zaručuje pohled na celou plochu, aniž by došlo k zastínění robotem. Nakonec je možné nastavit robot na požadovanou výšku na souřadnicích, na který se právě nachází a následně ovládat zápěstí (otevřít/zavřít). Je to z toho důvodu, aby nedošlo omylem k otevření zápěstí a upuštění objektů, s kterými robot manipuluje. Při nastavování pomocí zadání souřadnic nebo kliknutím je nastavena vzdálenost do které pokud se nachází robot od aktuální pozice dojde k nastavení přímo, pokud je však vzdálenost větší, robot je nastaven nedříve tak, aby s co nejmenší pravděpodobností narazil do objektů v pracovním prostoru, pokud se nějaké nachází v pracovním prostoru, následně dojde k nastavení základny, a nakonec celého robota.

Režim plánování trajektorie zobrazí mřížku, která zobrazuje možné body, kde se může robot pohybovat a umožní pomocí dvojitého kliknutí pravého nebo levého tlačítka zvolit buď počáteční nebo koncový bod. Po zvolení obou bodů pomocí plan trajectory začne plánování trajektorie a je zobrazována trajektorie v mřížce jako pohyblivý bod. Pomocí GO! tlačítka dojde k nastavení robota do počátečního bodu a následně dojde k pohybu po naplánované trajektorii, pokud byla nějaká trajektorie naplánovaná. Při nastavování do počátečního bodu je opět nastavena vzdálenost stejně jako při manuálním ovládání a dojde k nastavení stejným způsobem. Mřížka je nastavena pomocí tlačítka Set border, přičemž se odvíjí od buňky, která má střed v místě obrazu, kde je střed RGB kamery.

Režim sken nastaví robota tak, aby bylo vidět pracovní prostředí, aniž by robot zacláněl a dojde k detekci objektů, jak bylo popsáno. Následně jsou tyto objekty zobrazeny v mřížce v režimu plánování trajektorie, nikoliv v manuálním režimu. Tyto objekty jsou součástí prostředí, s kterým pracuje algoritmus DQN při navrhování trajektorie, a tak dojde k navržení trajektorie, která překážky obejde. Hranice překážky jsou přizpůsobeny v závislosti na nastavení mřížky, a taky velikosti pracovního prostoru umělé inteligence. Pokud hranice překážky přesahují pracovní prostor umělé inteligence, jsou zarovnány po hranice pracovního prostoru.

Pro komunikaci s Arduinem je použito vlákno. Vláknu jsou předány nastavení robota, která jsou postupně odeslány robotu, je však nutné mezi jednotlivým odesláním počkat na odpověď robota. Program mezitím neumožňuje uživateli udělat nic dokud nedojde k dokončení pohybu robota. Níže vizte část kódu kontrolní části pro komunikaci s Arduinem. Funkce sentArduinoInfo pouze naformátuje data pro odeslání.

```
def arduinoSent(data):
    global BUSY
    for coordinate in data:
        sentArduinoInfo(coordinate)
    BUSY = False


thread = threading.Thread(target=arduinoSent, args=(arduinoData,))
thread.start()
```

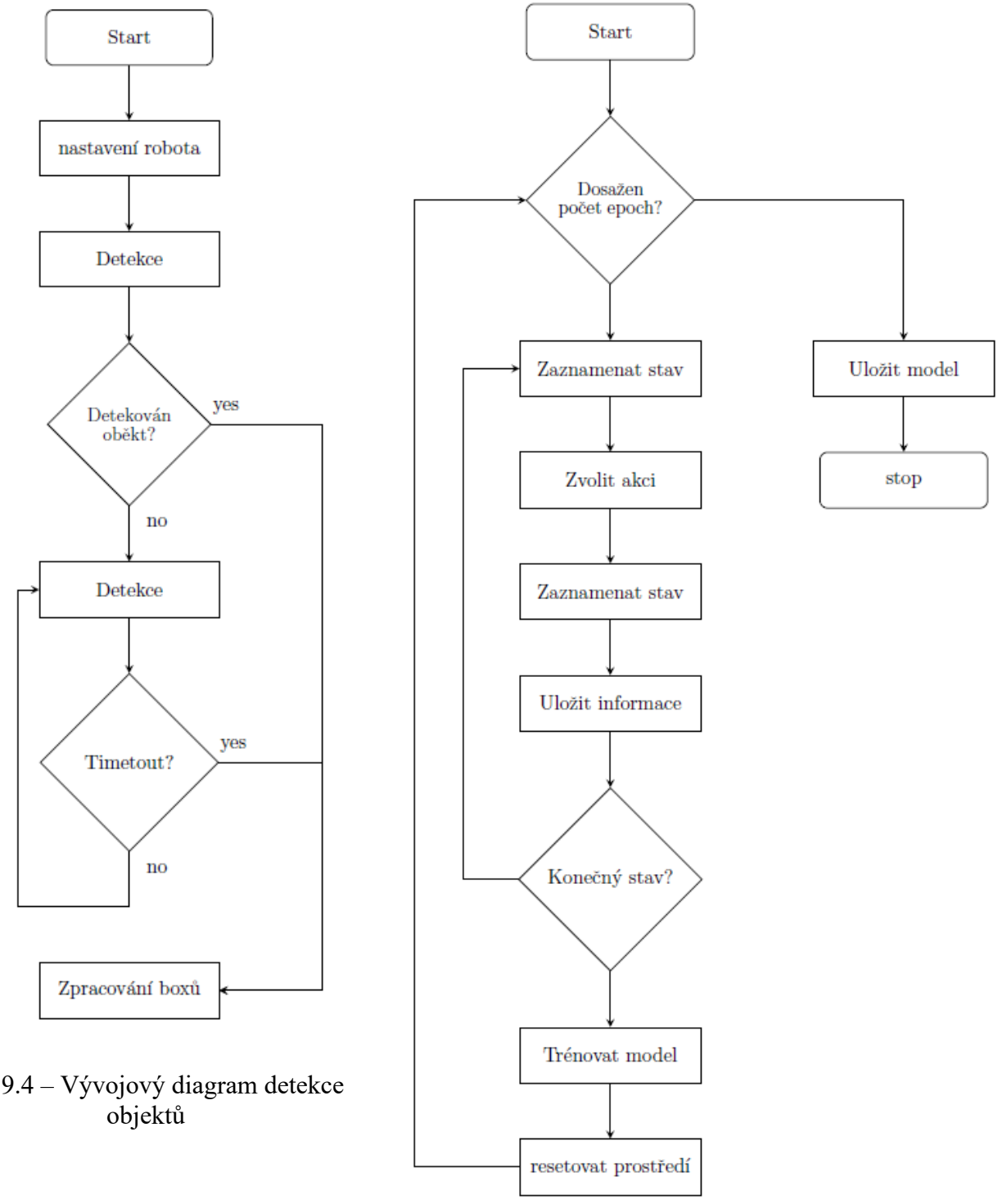
Pro výpočet inverzní kinematiky je využit programovací jazyk Matlab a generalizedInverseKinematics, který vytvoří tzv. solver. Solver je následně nastaven, je mu přiřazen model robota, parametry, které bude zohledňovat při řešení inverzní kinematiky a následně jsou nastaveny souřadnice koncového manipulátoru a orientace. Orientace je vždy směrem k zemi. Následně je vypočítaná inverzní kinematika a nastavení předáno prostředí v pythonu. Python spustí session v prostředí Matlab při spuštění kódu a pokaždé, když je nutno, zavolá funkci v prostředí Matlab, která vrátí nastavení robota pro požadované souřadnice. Model robota byl vytvořen ve formátu URDF.

```
gik = generalizedInverseKinematics;

gik.RigidBodyTree = robot;
gik.ConstraintInputs = {"position","aiming"};
q0 = homeConfiguration(robot);
posTgt = constraintPositionTarget("Arm_link5");
aimCon = constraintAiming("Arm_link5");
[qi,~] = gik(q0,posTgt,aimCon);
```

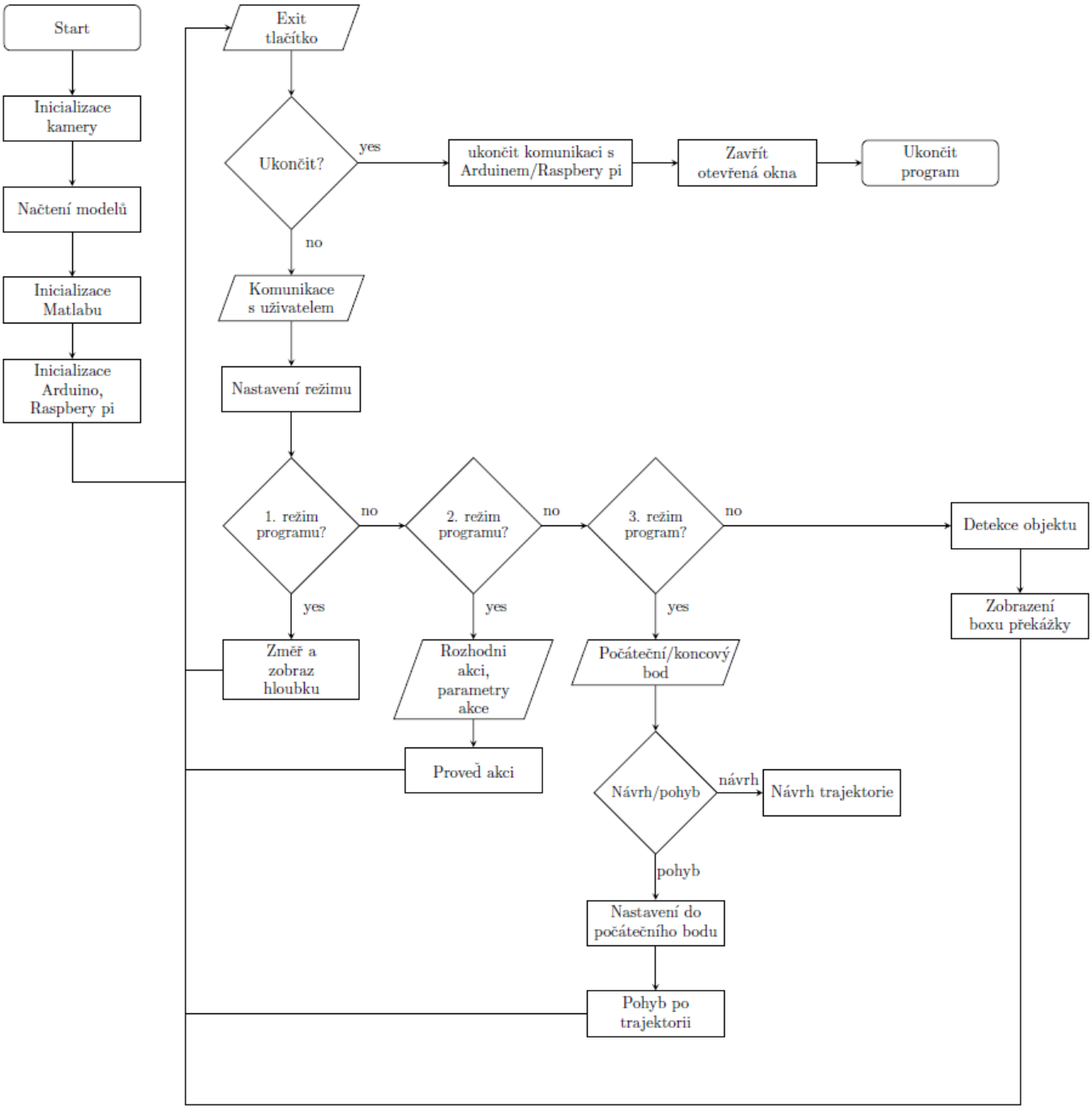
Po vypnutí programu dojde k zápisu do souboru nastavení konstant, souřadnice koncového bodu robota a konfigurace robota a je odeslán příkaz Arduinu pro zápis konfigurace do paměti EEPROM a robot je nastaven na výšku těsně nad pracovní plochou, aby nedošlo při odpojení napětí k pádu robotického ramena a k poškození.

9.1. VÝVOJOVÉ DIAGRAMY KONTROLNÍ ČÁSTI



Obr. 9.3 – Vývojový diagram trénování neuronové sítě

Obr. 9.4 – Vývojový diagram detekce objektů



Obr. 9.5 – Vývojový diagram programu kontrolní části robota

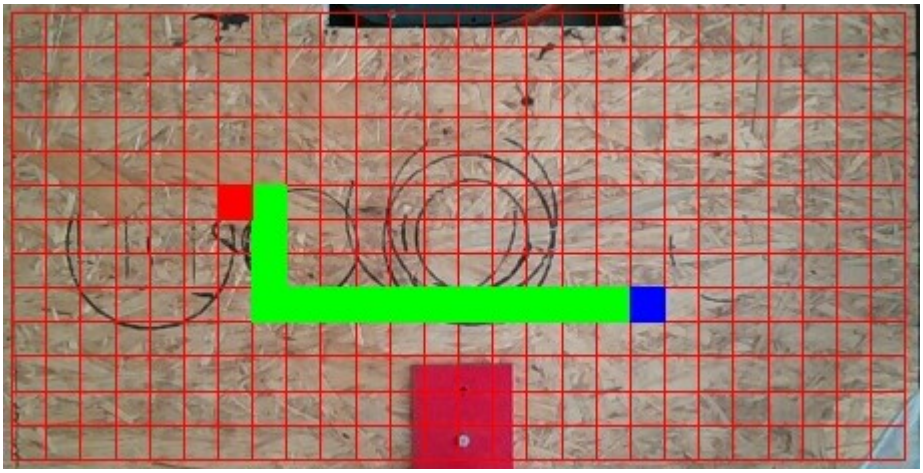
10. VÝSLEDKY

Konstrukce základy se projevila dosti stabilní, jen s menší možnou úpravou, a to části, ve které je upevněn hliníkový profil. Bylo by vhodné vyztužit stěny.

Jednotlivé klouby byly schopny zvednout zhruba 1,2 kg na délce paže 20 cm s nejslabším motorem NEMA 17 s momentem 0,14 N/m. Celé rameno bylo pak zatíženo maximální zátěží o váze 0,3 kg. Toto zatížení bylo ovšem provedeno, zatímco bylo dosaženo nejmenšího možného rozpětí ramene, což znamená druhý kloub ramene byl vždy namířen k zemi. Reálně tak robotické rameno dokáže manipulovat s menší zátěží, případně je nutno upravit pohyb robota při manipulaci s maximální zátěží.

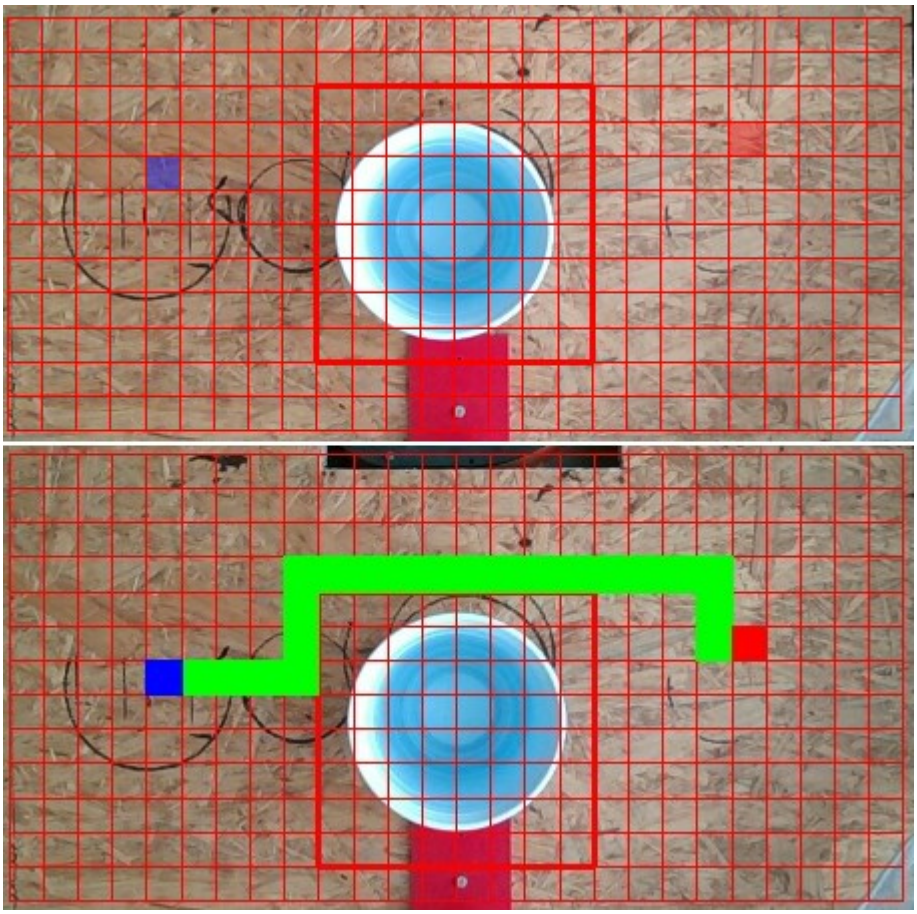
Rychlost kloubů byla stanovena jednotná kvůli komplikacím s komunikací. Největší možná rychlost kloubů byla dosažena 0,628 rad/s. S touto rychlostí bylo možné robot ovládat ale nebylo možné manipulovat s věcmi, resp. maximální zátěž se zmenšila výrazně, a proto byla nastavena pracovní rychlost 0,318 rad/s. Maximální rychlost odpovídá délce mezi pulzy 2,5 ms a pracovní rychlost odpovídá délce mezi pulzy 5 ms. Pracovní rychlost kloubů odpovídá rychlosti 0,04 m/s při pohybu v rovině po vypočítané trajektorii.

Na Obr. 10.1 je vidět návrh trajektorie ve volném prostoru, kde modře je vyznačen počáteční bod a červeně je označen koncový bod.



Obr. 10.1 – Ukázka návrhu trajektorie ve volném pracovním prostoru

V horní polovině Obr. 10.2 je vidět ukázka prostoru s překážkou, kde překážkou byl použit kelímek a byli zvoleny body po stranách kelímku tak, aby bylo nutné překážku obejít. Počáteční bod je opět vyznačen modře a červeně koncový. V dolní polovině je vidět navržená trajektorie.



Obr. 10.2 – Ukázka návrhu trajektorie v pracovním prostoru s překážkou

Ačkoliv došlo k úspěšnému navržení trajektorie, docházelo taky k situacím, kdy došlo k zacyklení a umělá inteligence opakovala pohyb z jednoho bodu do druhého. V tomto případě bylo nutné restartovat návrh trajektorie a zkusit zvolit jiný počáteční a koncový bod. Šlo převážně o situace, kdy se umělá inteligence při návrhu trajektorie nacházela vedle koncového bodu a nedošlo tak ke kompletnímu návrhu, ale došlo k úspěšnému obejití překážky.

11. ZÁVĚR

Konstrukce robotického ramena, konkrétně kloubů, pomocí částí vytisknutých na 3D tiskárně je sice dostačující, ale je dosažena menší přesnost při výrobě částí kloubů, která se odráží na tuhosti převodovky kloubu a tím i přesnosti kloubu, nicméně je stále dosažena přesnost <1° nastavení úhle kloubu. Hliníkový výlisek se osvědčil jako velmi dobrý materiál pro daný účel.

Velkou nevýhodou řídicí části robota jsou zvolené magnetické rotametry. Komunikace s nimi je velmi dlouhá a znemožňuje tak nastavení přesné rychlosti kloubu, a tak je dosažena menší přesnost koncového efektoru co se týče pohybu. Přesnost polohy je tímto řešením neovlivněna.

Řešení převod pohybu motoru pomocí lanek a trubiček na principu bowdenu se prokázalo jako velmi užitečné. Řešení je však vhodné doplnit o převodový poměr, aby bylo možné využít slabších motorů pro zmenšení celkového odebíraného proudu.

Zvolené řešení koncového efektoru není ideální, jelikož dochází k zahřívání servomotoru při manipulaci s objekty relativně brzy, a tak ztrácí sílu.

Detekce objektů pomocí YOLO algoritmu je velmi rychlá a relativně spolehlivá, záleží však na modelu. Verze YOLOv8 se osvědčila jako rychlá a použitelná v reálném čase a zároveň velmi stabilní.

Reinforcement learning se jeví jako velmi dobré řešení pro návrh trajektorie v dynamickém prostředí. Nevýhodou tohoto řešení je nutnost volby správného stavu prostředí a způsob odměny agenta. Jde však o velmi flexibilní řešení.

LITERATURA

ZIVID. *Structured Light* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://www.zivid.com/3d-structured

KUMAR, Prabu. *How Time-of-Flight (ToF) compares with other 3D depth mapping technologies* [online]. 2023, aktualizace 17. 10. 2023 [cit. 2024-05-05]. Dostupné z: https://www.e-consystems.com/blog/camera/technology/how-time-of-flight-tof-compares-with-other-3d-depth-mapping-technologies/

KUMAR, Prabu. *What is a ToF sensor? What are the key components of a ToF camera?* [online]. 2023, aktualizace 17. 10. 2023 [cit. 2024-05-05]. Dostupné z: https://www.e-consystems.com/blog/camera/technology/what-is-a-time-of-flight-sensor-what-are-the-key-components-of-a-time-of-flight-camera/

REGION GROWING METHOD FOR DEPTH MAPFCOLOR IMAGE. Vynálezci Tao ZHANG, Yu-Pao TSAI. Spojené státy. 13/669,453. Přihlášeno 6. 11. 2012. Uděleno 10. 10. 2013. Dostupné také z: https://patentimages.storage.googleapis.com/2e/ff/2c/552e0746fff73f/US20130266223A1.pdf

MITTAL, Himanshu, Avinash Chandra PANDEY, Mukesh SARASWAT, Sumit KUMAR, Raju PAL a Garv MODWEL. SPRINGER LINK. *A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets* [online]. 2021 [cit. 2024-05-05]. Dostupné z: https://link.springer.com/article/10.1007/s11042-021-10594-9

BUHL, Nikolaj. ENCORD. *Image Thresholding in Image Processing* [online]. 2023, Aktualizace 5. 5. 2024 [cit. 2024-05-05]. Dostupné z: https://encord.com/blog/image-thresholding-image-processing/

BEBIS, Dr. George. *Region Growing* [online]. 2004 [cit. 2024-05-05]. Dostupné z: https://www.cse.unr.edu/~bebis/CS791E/Notes/RegionGrowing.pdf

MUTNEJA, Vikram. *Methods of Image Edge Detection: A Review* [online]. 2015 [cit. 2024-05-05]. Dostupné z: https://www.researchgate.net/profile/Vikram-Mutneja/publication/282775998_Methods_of_Image_Edge_Detection_A_Review/links/596087ba0f7e9b8194096704/Methods-of-Image-Edge-Detection-A-Review.pdf

ABUSHAALA, Ahmed. *Edge Detection Methods* [online]. 2020 [cit. 2024-05-05]. Dostupné z: https://www.researchgate.net/profile/Ahmed-Abushaala/publication/340514789_Edge_Detection_Methods/links/5e8e23ba92851c2f5288e203/Edge-Detection-Methods.pdf

WALCZYK, Robert a T.D. ARMITAGE. *Comparative Study on Connected Component Labeling Algorithmsfor Embedded Video Processing Systems* [online]. 2010 [cit. 2024-05-05]. Dostupné z: https://www.researchgate.net/publication/220808599_Comparative_Study_on_Connected_Component_Labeling_Algorithms_for_Embedded_Video_Processing_Systems

JORDAN, Jeremy. *An overview of object detection: one-stage methods.* [online]. 2018 [cit. 2024-05-05]. Dostupné z: https://www.jeremyjordan.me/object-detection-one-stage/

PARK, Sieun. *A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN* [online]. 2021 [cit. 2024-05-05]. Dostupné z: https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c

HOU, Saihui, Zilei WANG a Feng WU. *Deeply Exploit Depth Information for Object Detection* [online]. 2016 [cit. 2024-05-05]. Dostupné z: https://openaccess.thecvf.com/content_cvpr_2016_workshops/w24/papers/Hou_Deeply_Exploit_Depth_CVPR_2016_paper.pdf

JORDAN, Caleb. *Feature Extraction from Depth Maps for Object Recognition* [online]. 2013 [cit. 2024-05-05]. Dostupné z: https://stacks.stanford.edu/file/druid:np318ty6250/Jordan_Preprocessing_and_Feature_Extraction_from_Depth_Maps.pdf

NEUVITION, INC. *Clustering Algorithms* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://www.neuvition.com/technology-blog/clustering-algorithms.html

HRUTKA, B. P., Z. SIKI a B. TAKÁCS. *VOXEL-BASED POINT CLOUD SEGMENTATION AND BUILDING DETECTION* [online]. 2022 [cit. 2024-05-05]. Dostupné z: https://isprs-archives.copernicus.org/articles/XLVIII-4-W1-2022/209/2022/isprs-archives-XLVIII-4-W1-2022-209-2022.pdf

LOEB, Regis. *Deep learning on point clouds for 3D object detection* [online]. 2022 [cit. 2024-05-05]. Dostupné z: https://medium.com/@regis.loeb/playing-with-point-clouds-for-3d-object-detection-eff1d98e526a

ROBOTICS MP. *Reactive Motion Planning Algorithms* [online]. 2022 [cit. 2024-05-05]. Dostupné z: https://roboticsmp.medium.com/reactive-motion-planning-algorithms-14f29d0186e7

MEDIAVILLA, Margarita, José L. GONZÁLEZ, Juan C. FRAILE a José. R. PERÁN. *REACTIVE PATH PLANNING FOR ROBOTIC ARMS WITH MANY DEGREES OF FREEDOM IN DYNAMIC ENVIRONMENTS* [online]. 2002 [cit. 2024-05-05]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1474667015393071?ref=pdf_download&fr=RR-2&rr=86c8a76a2f68b36c

KHOKHAR, Arushi. *Probabilistic Roadmap (PRM) for Path Planning in Robotics* [online]. 2021 [cit. 2024-05-05]. Dostupné z: https://medium.com/acm-juit/probabilistic-roadmap-prm-for-path-planning-in-robotics-d4f4b69475ea

HUPPI, Matthias, Luca BARTOLOMEI, Ruben MASCARO a Margarita CHLI. *T-PRM: Temporal Probabilistic Roadmap for Path Planning in Dynamic Environments* [online]. 2022 [cit. 2024-05-05]. Dostupné z: https://lucabartolomei.github.io/publications/IROS_2022_TPRM.pdf

GOVIND, Aswath. *Rapidly exploring random Trees (RRT) and their much nicer properties.* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://medium.com/geekculture/rapidly-exploring-random-trees-rrt-and-their-much-nicer-properties-7b5d983e5b18

LAVALLE, Steven M. *Gradient-Based Trajectory Optimization* [online]. 2020 [cit. 2024-05-05]. Dostupné z: https://lavalle.pl/planning/node795.html

GUL, Faiza, Imran MIR, Laith ABUALIGAH, Putra SUMARI a Agostino FORESTIERO. *A Consolidated Review of Path Planning and Optimization Techniques: Technical Perspectives and Future Directions* [online]. 2021 [cit. 2024-05-05]. Dostupné z: https://www.mdpi.com/2079-9292/10/18/2250

YU, Jinglun, Yuancheng SU a Yifan LIAO. *The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning* [online]. 2020 [cit. 2024-05-05]. Dostupné z: https://www.frontiersin.org/articles/10.3389/fnbot.2020.00063/full#B6

CHANDA, Kaustav Kumar. *Q-Learning in Python* [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://www.geeksforgeeks.org/q-learning-in-python/?ref=lbp

YOON, Chris. *Double Deep Q Networks* [online]. 2019 [cit. 2024-05-05]. Dostupné z: https://towardsdatascience.com/double-deep-q-networks-905dd8325412

IBM. *What is supervised learning?* [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://www.ibm.com/topics/supervised-learning

IBM. *What is a decision tree?* [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://www.ibm.com/topics/decision-trees

FREE LEARNING PLATFORM FOR BETTER FUTURE. *Random Forest Algorithm* [online]. 2021 [cit. 2024-05-05]. Dostupné z: https://www.javatpoint.com/machine-learning-random-forest-algorithm

MATTHEWS, Kayla. *5 materials to evaluate for designing, building robust robots* [online]. 2019 [cit. 2024-05-05]. Dostupné z: https://www.therobotreport.com/materials-rugged-robot-design-building/

PROTOLABS. *Materials that command the Robotics Industry* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://www.protolabs.com/en-gb/resources/blog/materials-that-command-the-robotics-industry/

BEYATLI, Ahmed. *Beginners Guide to Actuators* [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://acrome.net/post/beginners-guide-to-actuators

RAO, Ravi. *Robot Joints: An In-Depth Guide to Anatomy, Physics and Challenges in Design* [online]. 2023 [cit. 2024-05-05]. Dostupné z: https://www.wevolver.com/article/robot-joint

CVPR. *Frustum PointNets for 3D Object Detection from RGB-D Data* [online]. 2018 [cit. 2024-05-08]. Dostupné z: https://stanford.edu/~rqi/frustum-pointnets/

Příloha A

Příloha k bakalářské práci

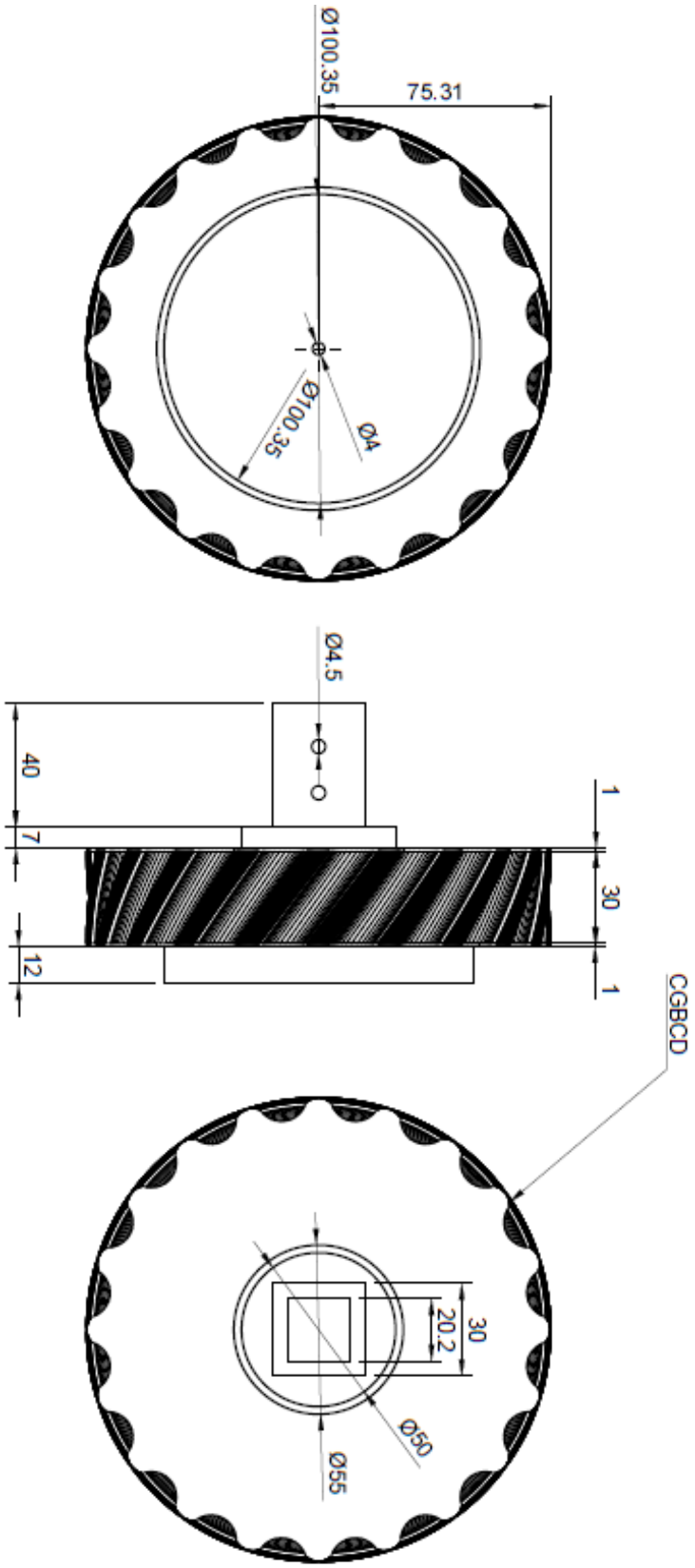
Návrh robotického manipulátoru s funkcí vyhýbání se překážkám

Matouš Volák

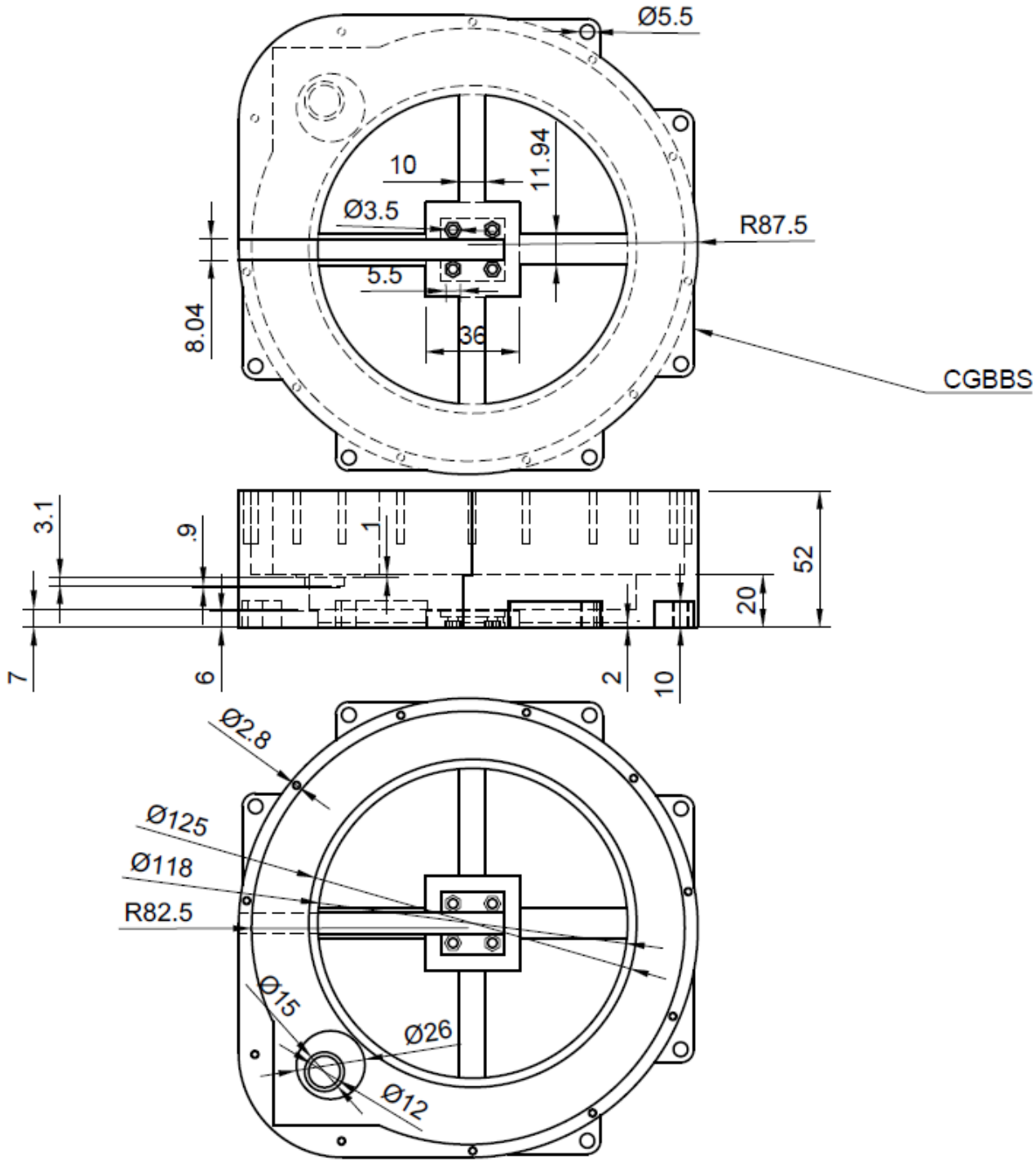
Výkresy

OBSAH

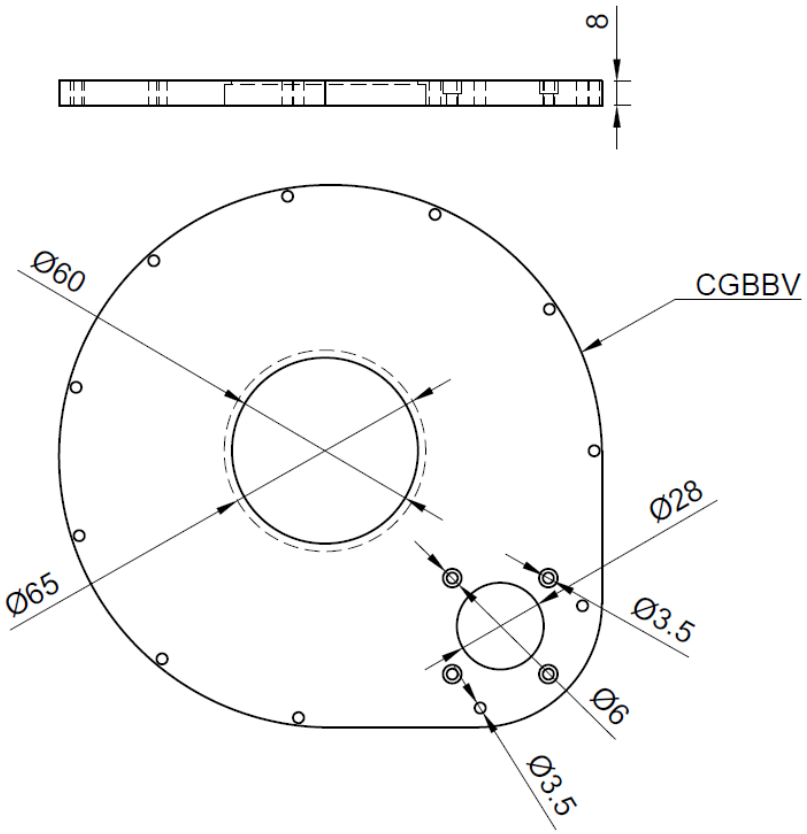
1. Výkres cykloidního disku základny

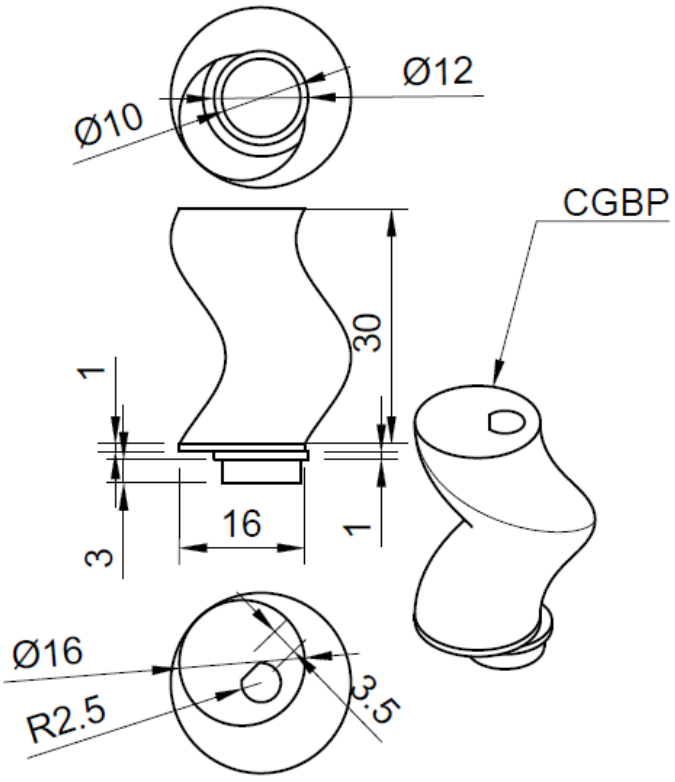


2. Výkres krytu převodovky základny

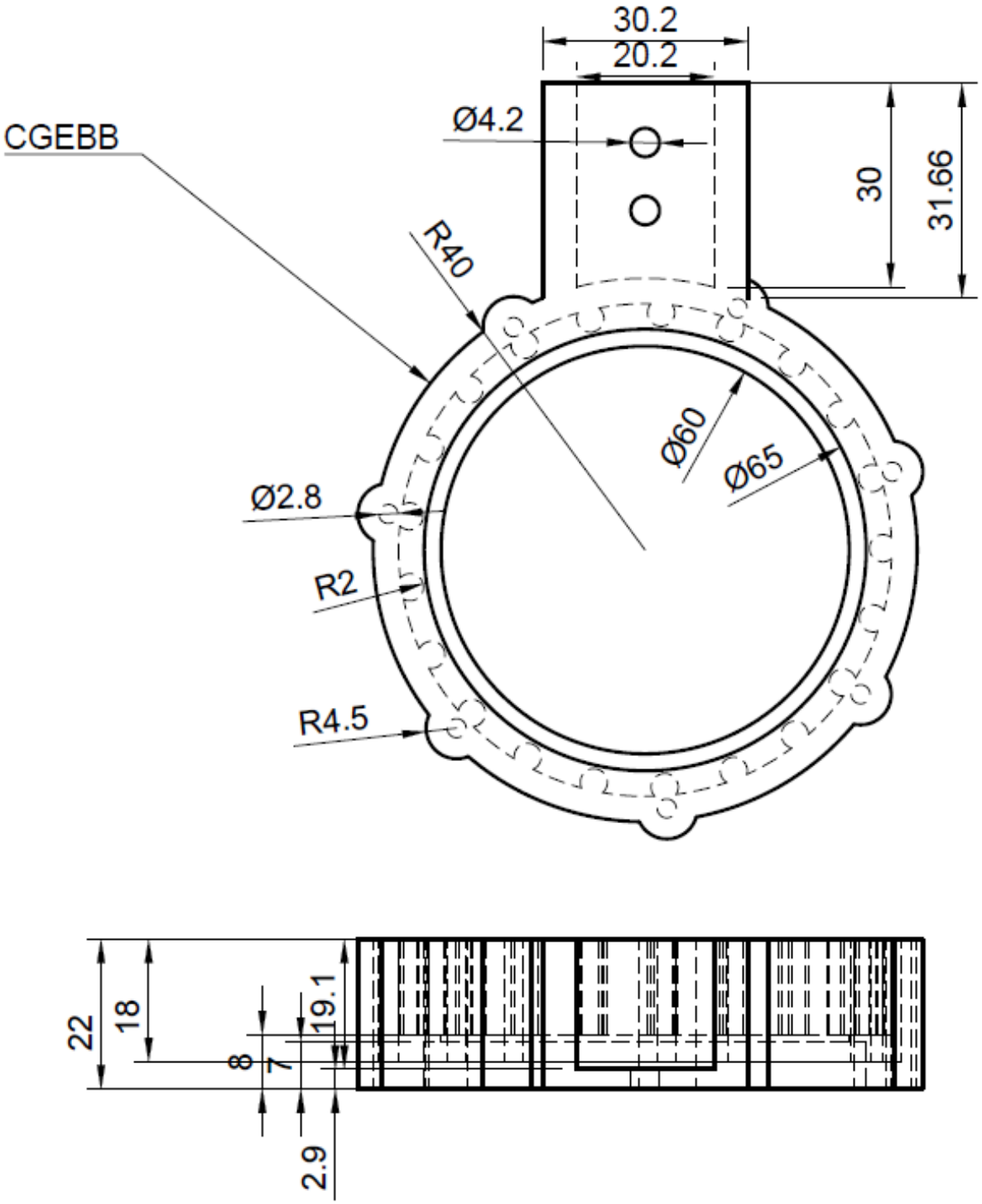


3. Výkres poklopu krytu základny, hnacího kolečka převodovky základny

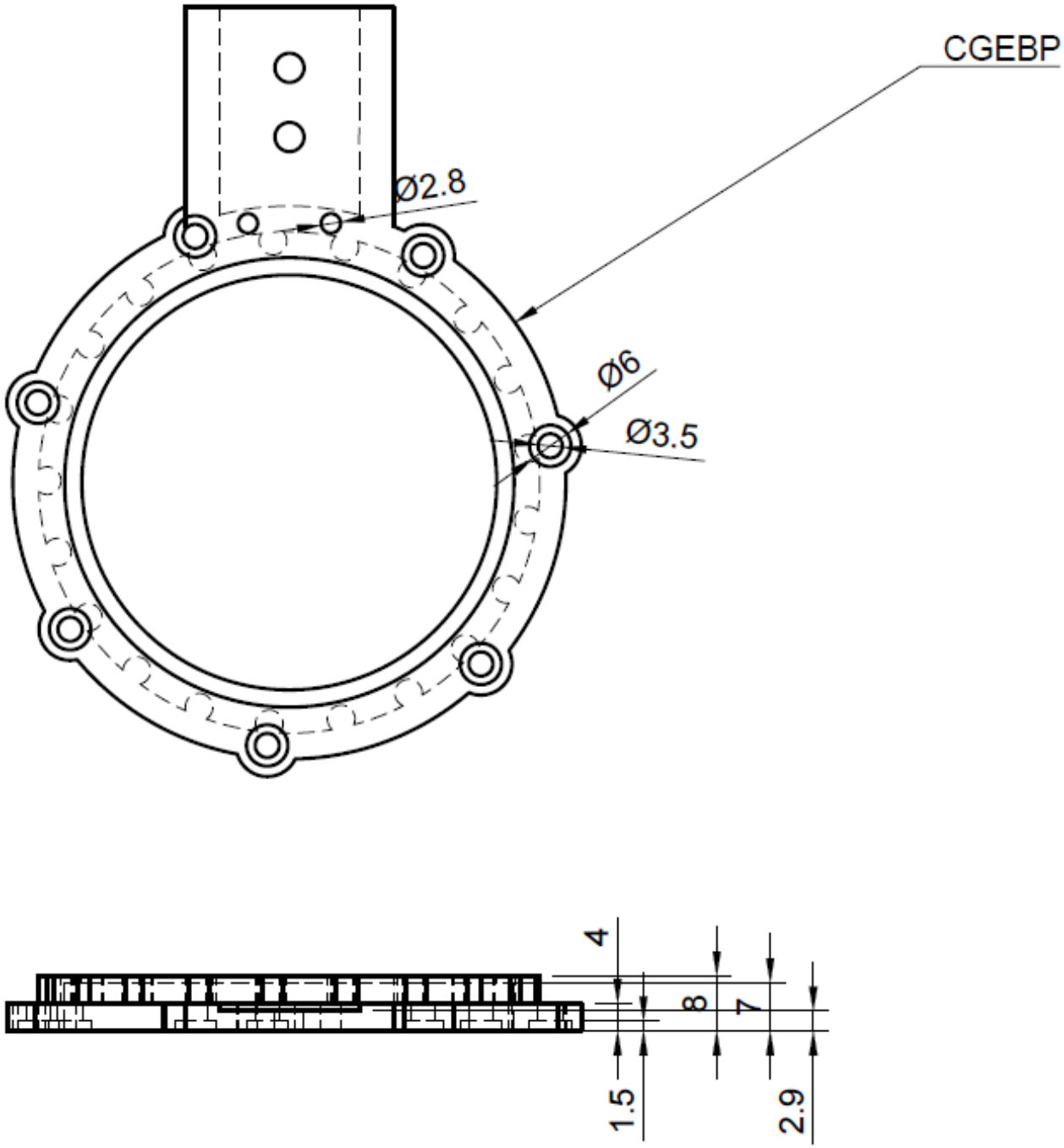




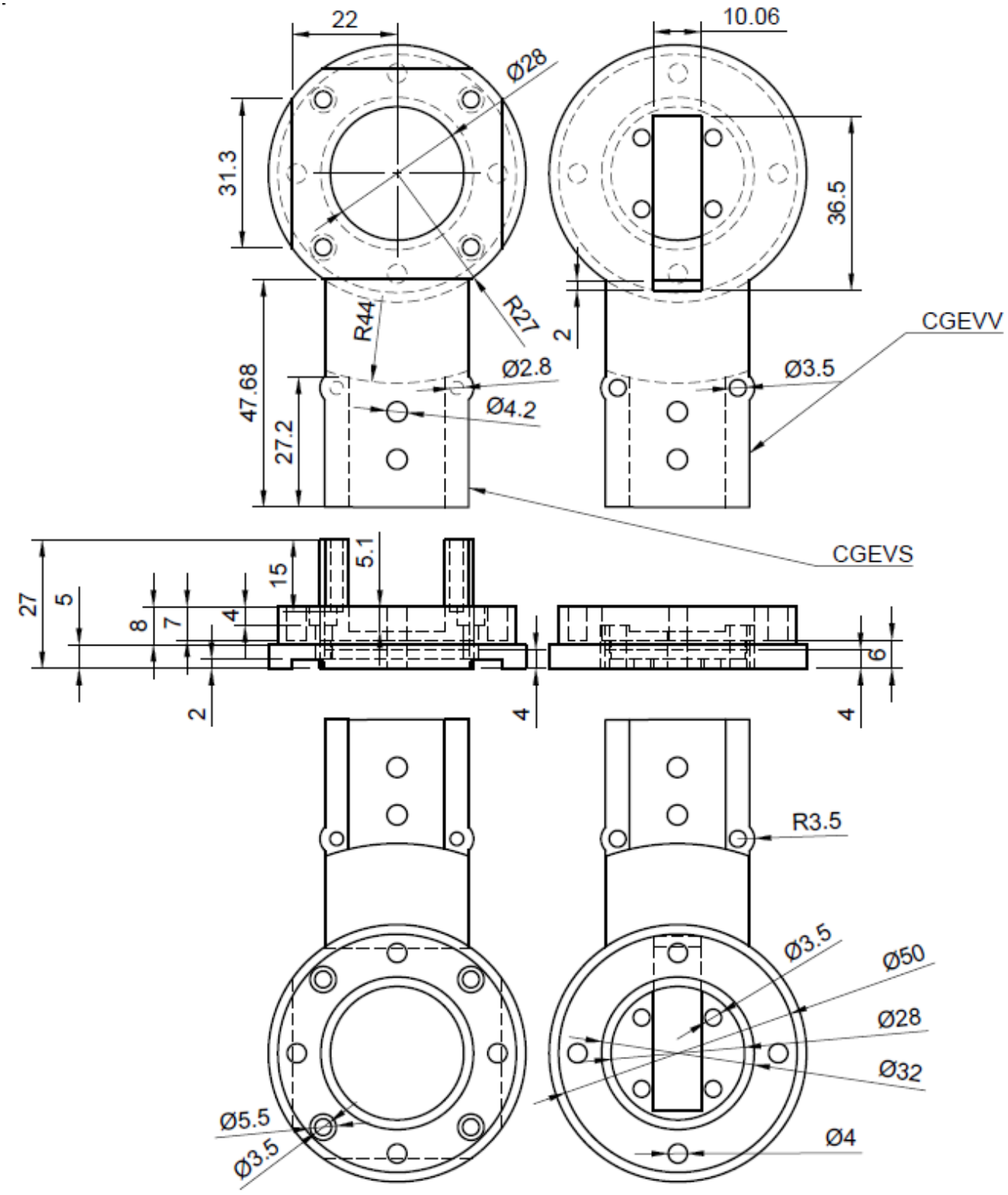
4. Výkres krytu převodovky kloubu



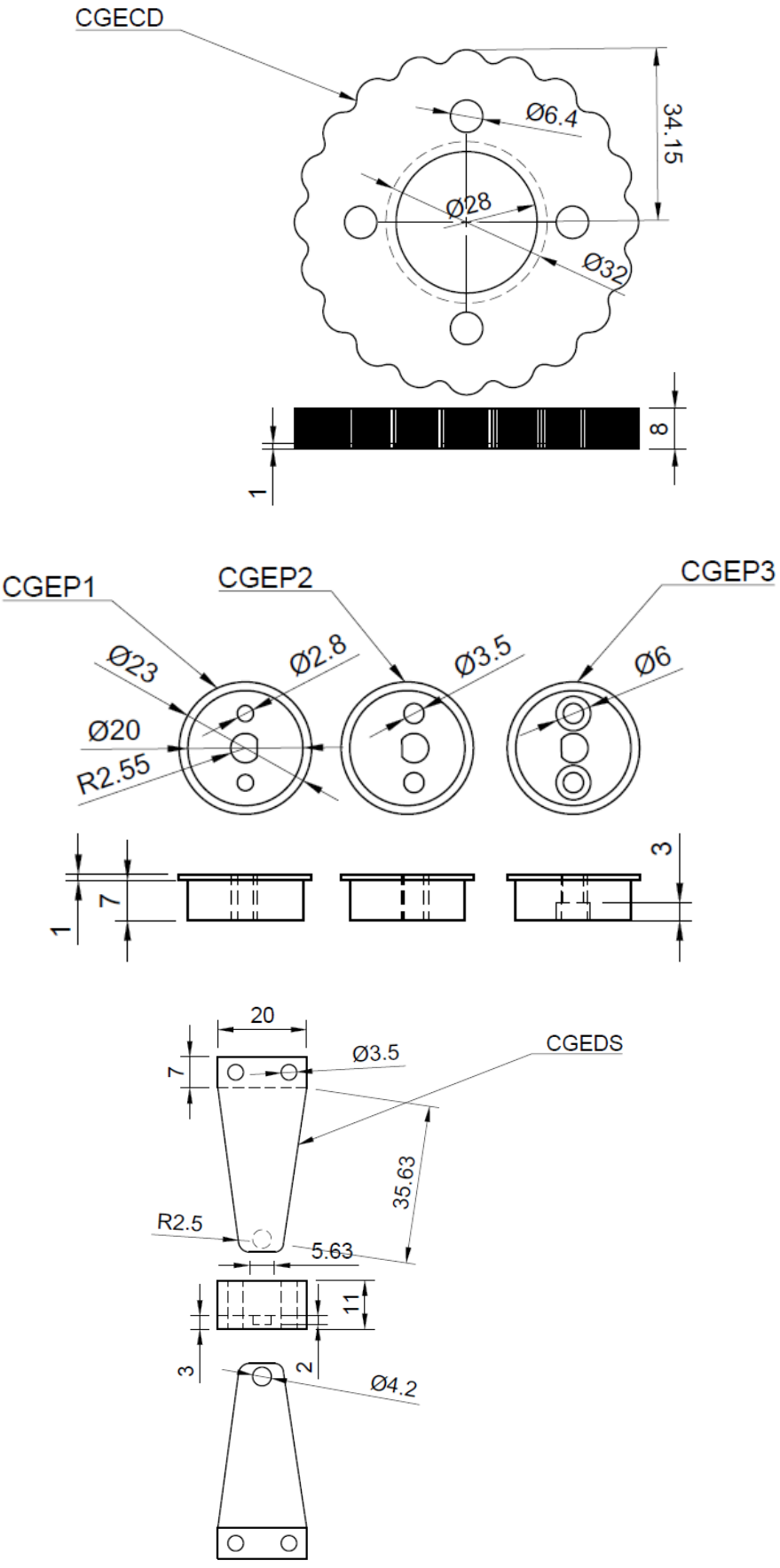
5. Výkres poklopu převodovky kloubu



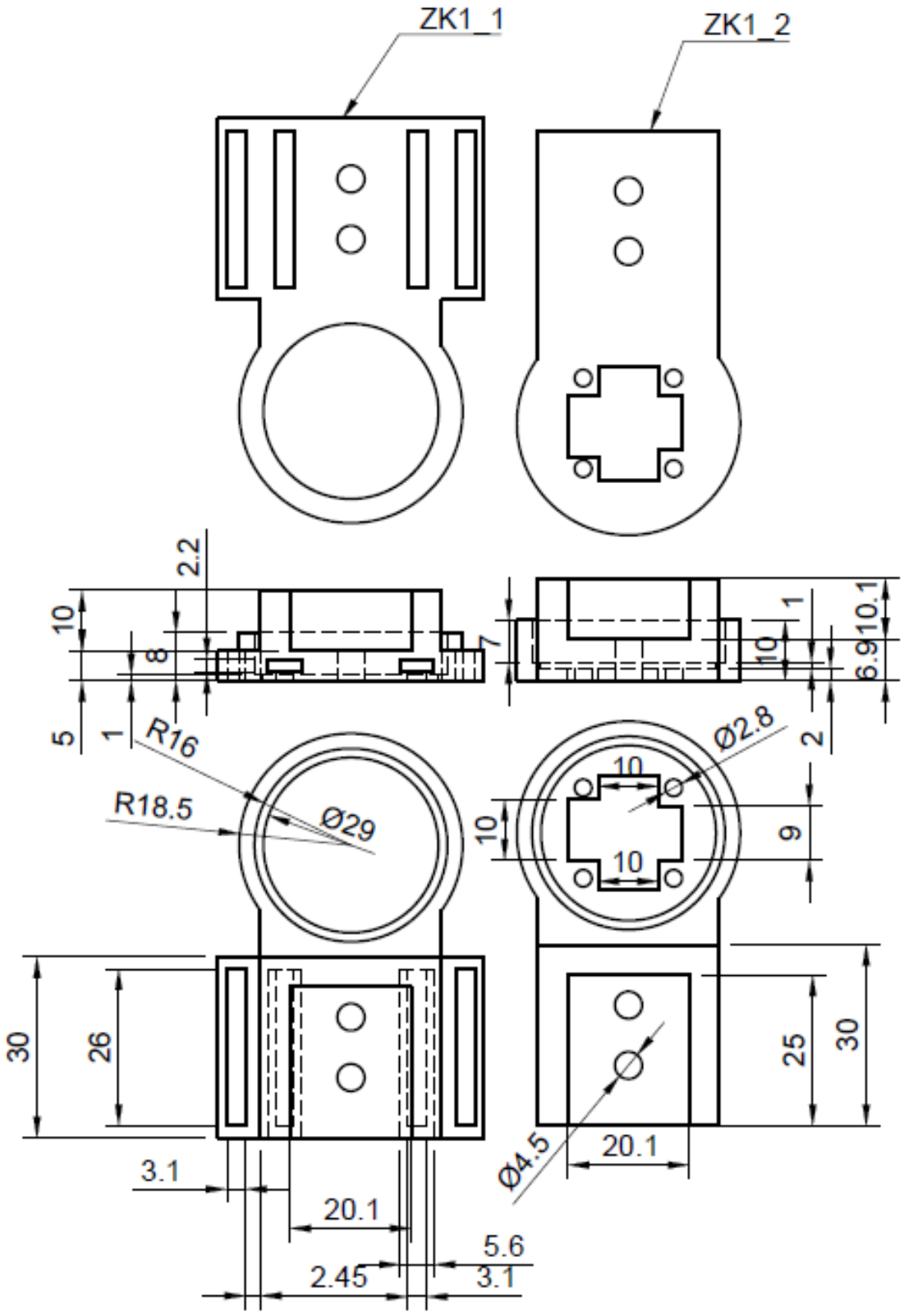
6. Výkres rotační části převodovky kloubu



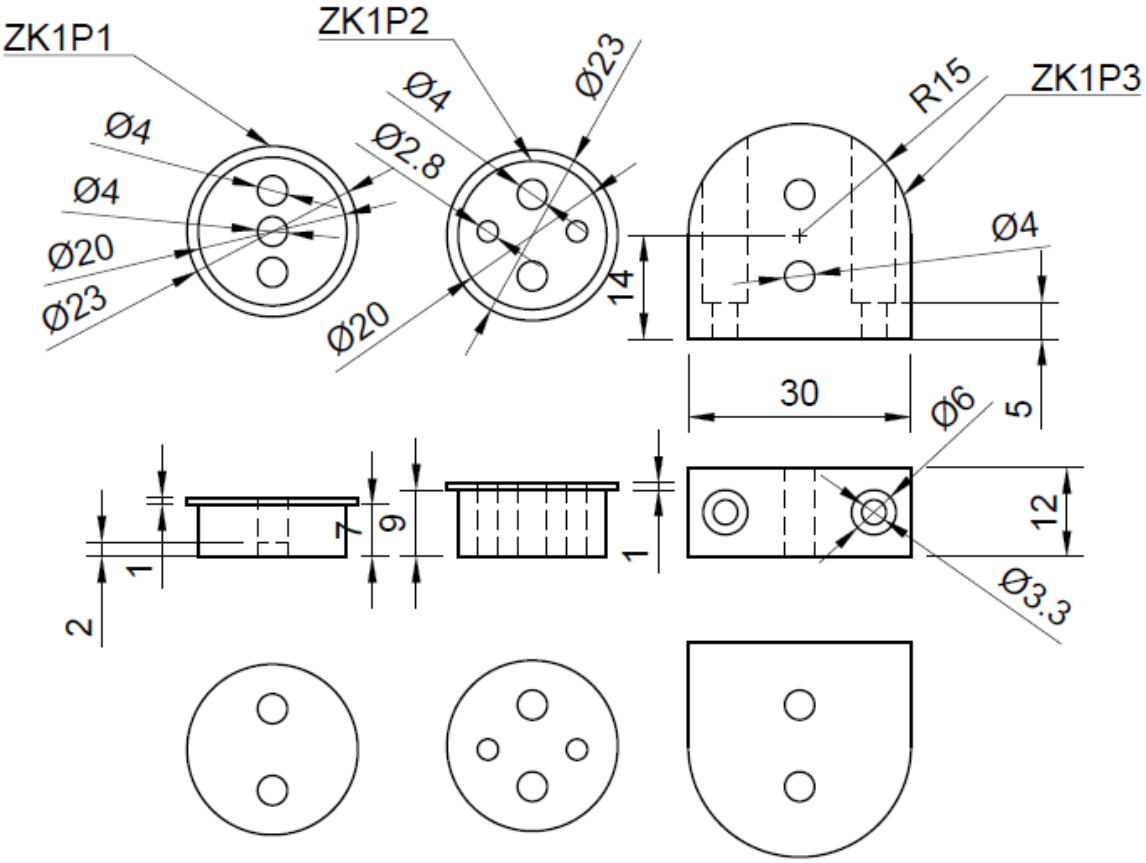
7. Výkres cykloidního disku, hnacího kolečka, držáku na magnet

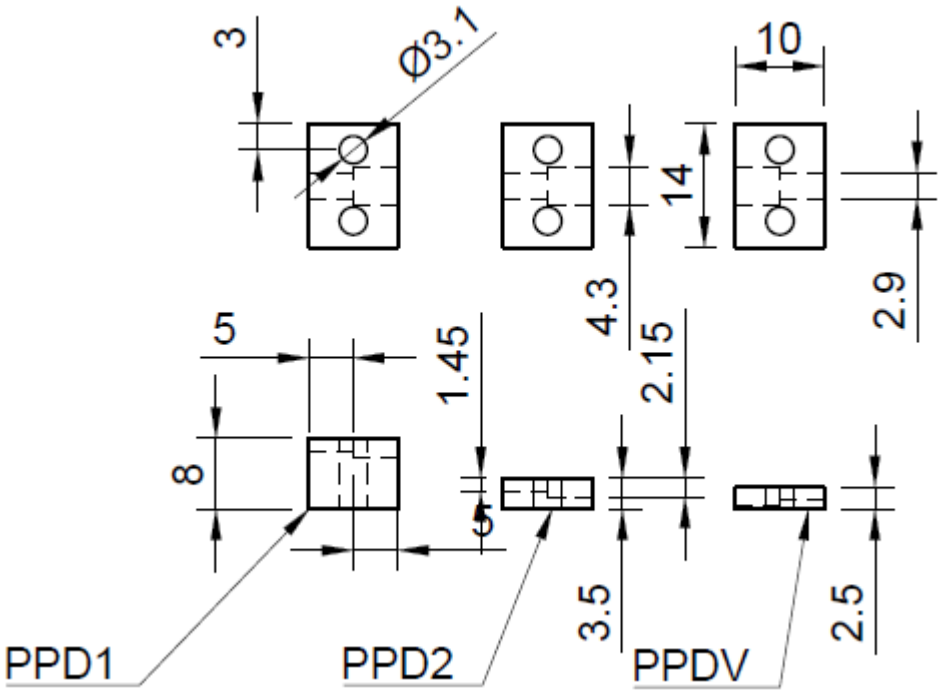


8. Výkres stacionární části kloubu zápěstí

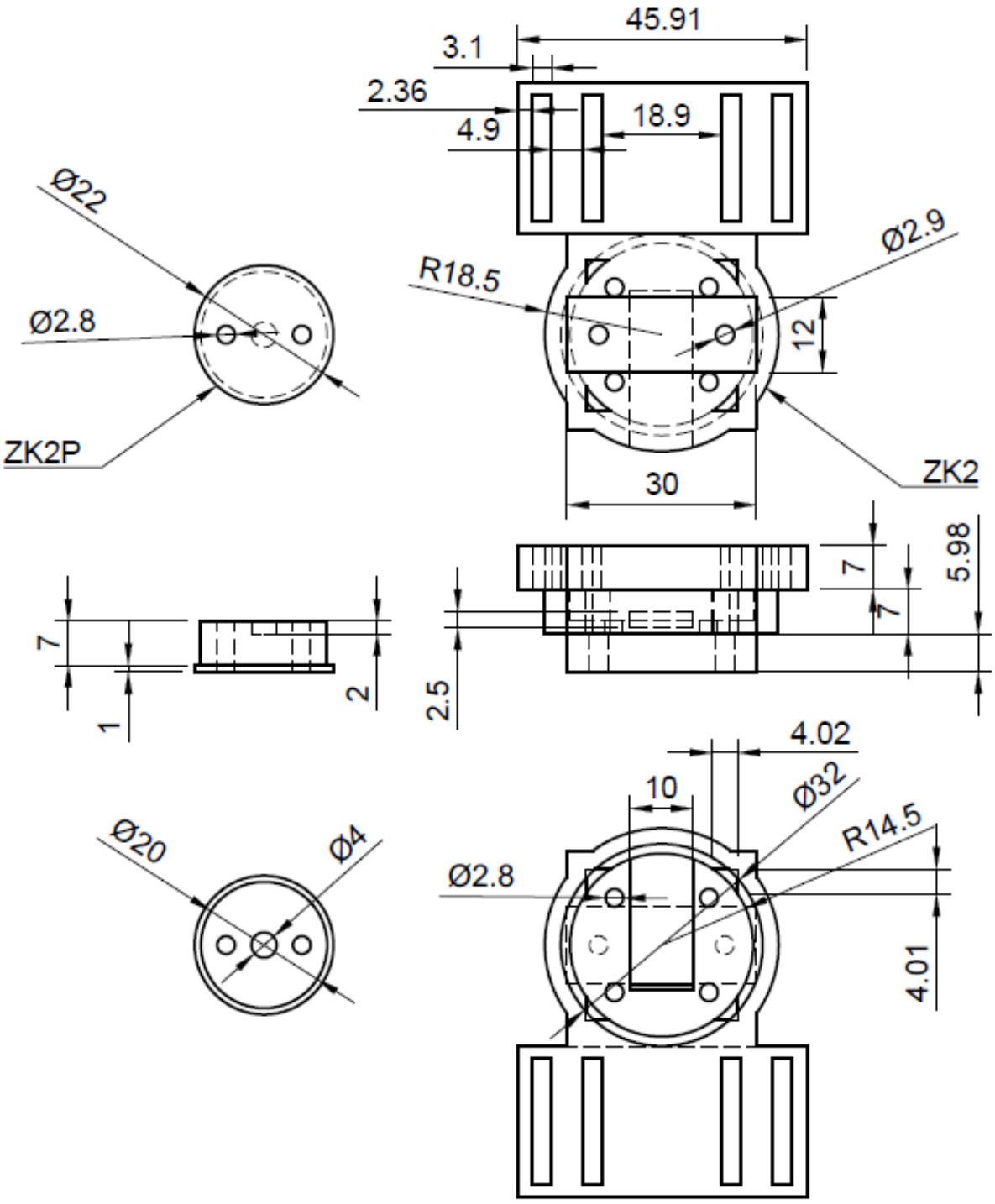


9. Výkres rotační části kloubu zápěstí, držáků na hadičky

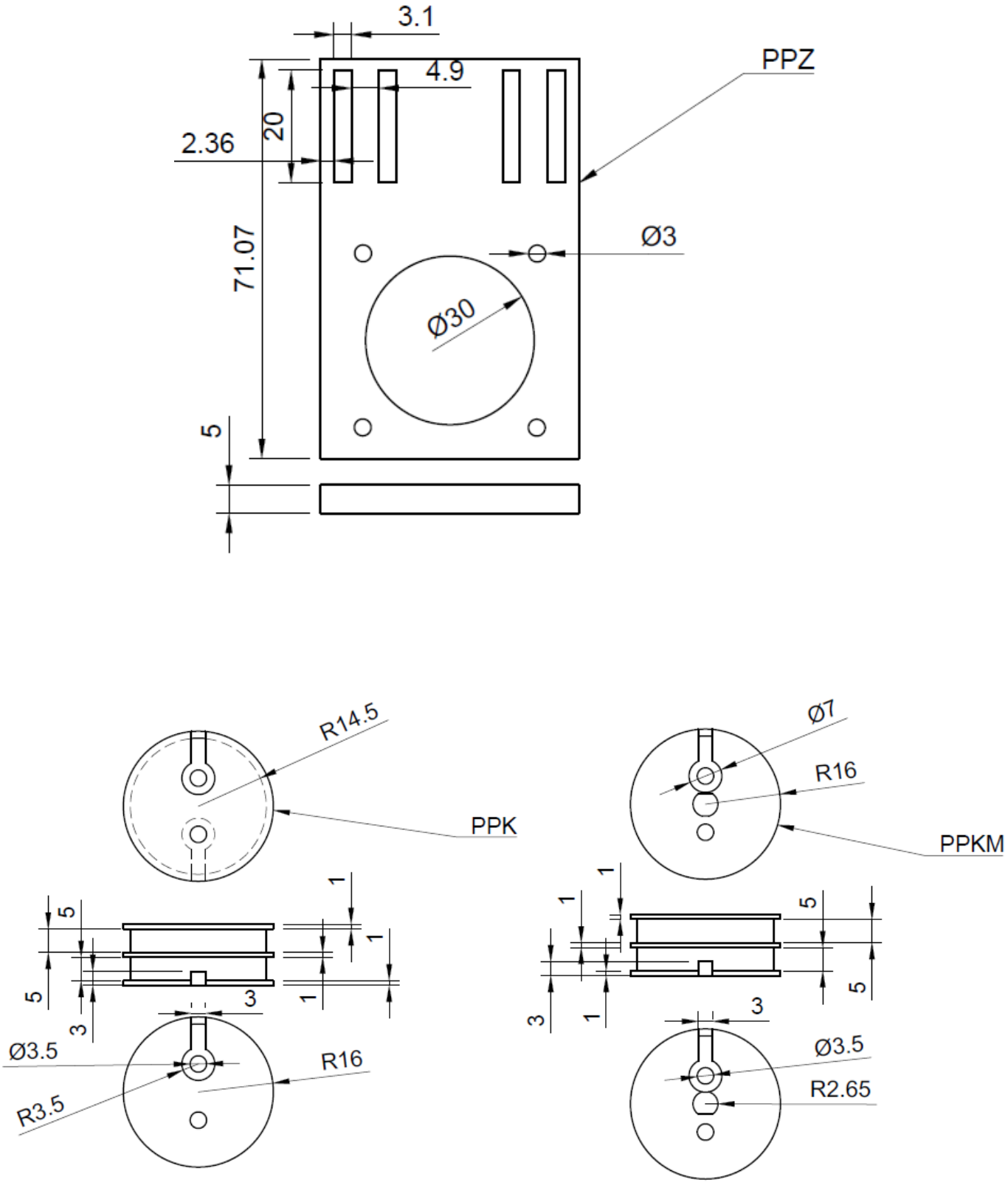




10. Výkres stacionární a pohyblivé části 2. kloubu zápěstí



11. Výkres držáku na hadičky, kolečka na lanka



12. Výkresy end-efektoru