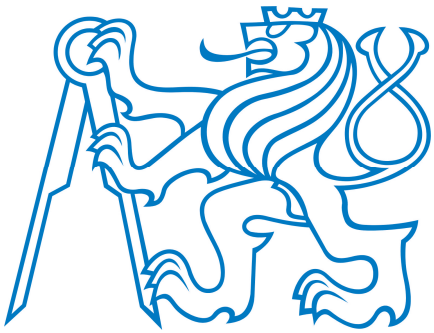
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta strojní – Ústav přístrojové a řídicí techniky



BAKALÁŘSKÁ PRÁCE

UŽIVATELSKÉ ROZHRANÍ PRO POLOHOVÁNÍ ROBOTICKÉ RUKY BCN3D MOVEO

User interface for positioning the robotic arm BCN3D MOVEO

Patrik Zach

2019/2020

Prohlašuji, že jsem tuto práci vypracoval(a) samostatně s použitím literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

Datum: ....

podpis

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zach** Jméno: **Patrik** Osobní číslo: **473565**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Uživatelské rozhraní pro polohování robotické ruky BCN3D MOVEO

Název bakalářské práce anglicky:

User interface for positioning the robotic arm BCN3D MOVEO

Pokyny pro vypracování:

1. Proveďte rešerši na téma „robotická ruka BCN3D MOVEO“ a zvolte potřebné prostředky pro její řízení. Vyberte vhodné softwarové prostředí pro vývoj programu s uživatelským prostředím, ve kterém bude možné zadávat zvolené operace robotické ruky a jejího pohybu dle uživatele. Navrhněte grafickou podobu a obsah uživatelského prostředí.
2. Na základě předchozí rešerše naprogramujte uživatelské prostředí.
3. Zprovozněte pohyby robotické ruky BCN3D MOVEO a otestujte její řízení ve Vámi navrženém uživatelském prostředí.

Seznam doporučené literatury:

[1] MEIER, Burkhard. Python GUI Programming Cookbook: Develop Functional and Responsive User Interfaces with Tkinter and PyQt5. 3rd edition. Birminghan, UK: Packt Publishing, Limited, 2019. ISBN 9781838828813.
[2] Dokumentace projektu BCN3D MOVEO na adrese https://www.bcn3d.com/bcn3d-moveo-the-future-of-learning/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Zdeněk Novák, Ph.D., U12110.1

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **30.04.2020** Termín odevzdání bakalářské práce: **27.08.2020**

Platnost zadání bakalářské práce: _____

Ing. Zdeněk Novák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Anotace

Bakalářská práce se zabývá rozpohybováním robotické ruky BCN3D Moveo a návrhem grafického rozhraní. V teoretické části jsou představeny základní komponenty pro sestavení robotické ruky a možnosti pro vytvoření grafického rozhraní. V části praktické je popsáno vytvoření grafického rozhraní pomocí knihovny PyQt5. Je zde také uvedena Denavit-Hartenbergova úmluva pro výpočet přímé kinematiky. V další části práce je vysvětlena inverzní kinematika manipulátoru. Výpočet přímé a inverzní kinematiky je poté implementován v programovacím jazyku Python.

Klíčová slova: BCN3D Moveo, Python, PyQt5, robotická ruka, GUI, přímá kinematika, inverzní kinematika, Denavit-Hartenbergova úmluva, Qt Designer, Arduino, sériový manipulátor

Abstract

The bachelor's thesis deals with movement of the BCN3D Moveo robotic arm and the design of a graphical interface. In the theoretical part the basic components for assembling a robotic arm and the possibilities for creating a graphical interface are introduced. In practical part is described a creation of a graphical interface using the PyQt5 library. Denavit-Hartenberg is also mentioned here for the calculation of forward kinematics. The next part of the work explains the inverse kinematics of the manipulator. Calculation of forward and inverse kinematics is then implemented in the Python programming language.

Keywords: BCN3D Moveo, Python, PyQt5, robotic arm, GUI, forward kinematics, inverse kinematics, Denavit-Hartenberg notation, Qt Designer, Arduino, Serial Manipulator

Poděkování

Děkuji svému vedoucímu bakalářské práce panu Ing. Zdeňkovi Novákovi, Ph.D. za konzultace, užitečné připomínky a rady při vypracování.

Obsah

1 Úvod

Historie průmyslové robotizace se začala psát na počátku druhé poloviny 20. století. George Devol se svou společností Unimation si nechal patentovat prvního průmyslového robota. Po 7 letech, roku 1961, se robotický manipulátor stal součástí výrobního procesu. Větší rozmach robotizace přišel až ke konci 70. let, kdy se do výroby pustilo několik dalších japonských korporací. Dnes Japonsko patří mezi absolutní špičku v odvětví robotizace, to možná i díky rychle stárnoucí populaci a jiným demografickým ukazatelům. [1], [2]

Pojem průmyslová robotizace není v dnešní době cizí. Nedostupnost levné pracovní síly nutí mnoho manažerů k nasazením nejen robotických manipulátorů. Výhod robotizace je nespočet, mezi ty hlavní patří zvýšení produktivity, úspora ve výrobě a přesnost provedení úkonů. Využívá se pak hlavně u opakovatelných procesů, například svařování automobilových karosérií či lakování vagónů. Manipulátory se používají například pro výměnu obrobků či jakoukoliv těžkou manipulaci. V České republice se trend využívání robotů neustále zvyšuje a nahrazuje zejména opakovatelnou manuální práci. [3]

Robot aplikovaný v průmyslových odvětví se od tohoto projektu principielně moc neliší. Konstrukce průmyslového robota tvoří pevnější a odolnější materiály a motory s větším kroutícím momentem. Ovládání jednotlivých os zajišťuje řídící systém PLC, který pomocí ethernetu zpracovává akce, které uživatel zvolil v HMI.

Cílem této práce je sestavení robotické ruky BCN3D Moveo a následně vytvoření grafického uživatelského rozhraní ve vhodně zvoleném programovacím jazyku. Výstupem bakalářské práce by mělo být navržené ovládání pro pohyb jednotlivých krokových motorů bez nutnosti použití jiné aplikace než samotně naprogramovaného souboru. Robotická ruka by měla umět uchopit a přemístit lehký předmět z bodu A do bodu B.

2 Teoretická část

2.1 BCN3D Moveo

Projekt BCN3D Moveo je open-source pětiosá robotická ruka, která byla vyvinuta společností BCN3D Technologies ve spolupráci s ministerstvem školství v Barceloně. Cílem tohoto projektu je přiblížení průmyslové automatizace nejen studentům, ale i zájemcům o robotiku. [4]

Robotická ruka je navržena tak, že veškeré její nosné části mohou být vytištěny na 3D tiskárně, což se nám výrazně promítne do pořizovací ceny. Mozkem celého systému je Arduino Mega 2560, který společně s motor shieldem ovládá 6 krokových motorů, konkrétně jejich stepper drivery, a servomotor. Krokové motory přenáší svůj točivý moment pomocí ozubených řemenů a otáčí nám tak jednotlivé osy. Servomotor slouží k ovládání kleští, pomocí nichž jsme schopni uchopit předmět. Napájení těchto komponent je zajištěno 12V zdrojem napětí. Arduino je řízeno upraveným programem Marlin, který je používán u 3D tiskáren. [5]

Celý projekt je volně ke stažení na webové stránce www.github.com, kde se dá nalézt CAD model robotické ruky, 3D modely ve formátu STL, BOM a firmware pro ovládání.

Obr. 1: BCN3D Moveo [4]

V následujících částech budu postupně přibližovat nejdůležitější komponenty robotické ruky.

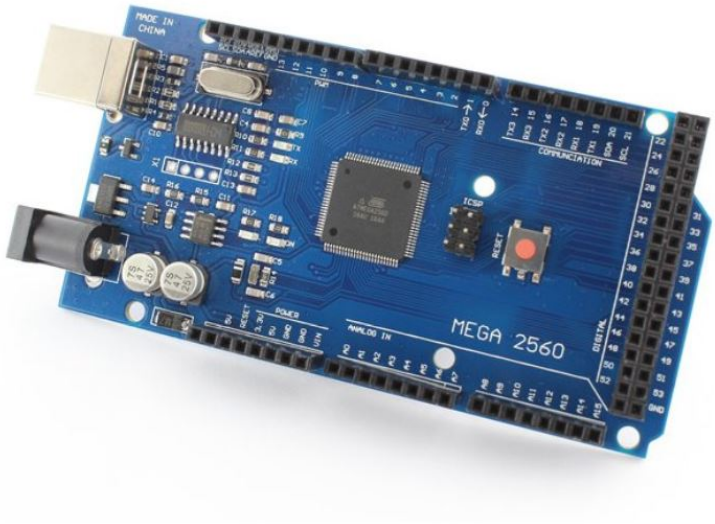
2.2 Řídící část

2.2.1 Arduino Mega 2560

Mikrokontrolér Arduino Mega 2560 od firmy Atmel patří mezi hojně užívané řídící jednotky, zejména v oblasti robotiky a 3D tiskáren. Vznikl zvětšením Arduina Uno, aby zde mohlo být umístěno více pinů a výkonější čip. Většinou se používá v kombinaci se shieldem, kterých je na trhu celá řada - WiFi shield, Ethernet shield a motor shield. [6]

Arduino Mega 2560 operuje na 5 V, které je zajištěno buď napájením z PC přes USB port nebo z baterie přes napájecí konektor. K dispozici máme 54 digitálních inputů/outputů, z nichž lze 15 použít jako PWM, 16 analogových inputů a 4 inputy UART, což je asynchronní přijímač/vysílač, který má vlastní generátor hodinového signálu. Na desce dále najdeme ICSP hlavici, sloužící pro externí programování, 16 MHz oscilátor, indikační LED diody, resetovací tlačítko a hlavní čip desky ATMEGA2560. [6], [7], [8]

Ukládání dat nám na Arduinu zařizují 3 druhy paměti. Flash paměť, která ukládá námi napsaný program. SRAM, která nahrává data při spuštění programu - jedná se o proměnné, které vkládáme do hlavičky programu. EEPROM, jež slouží k ukládání během chodu programu a data uchovává i během vypnutého napájení. [9]



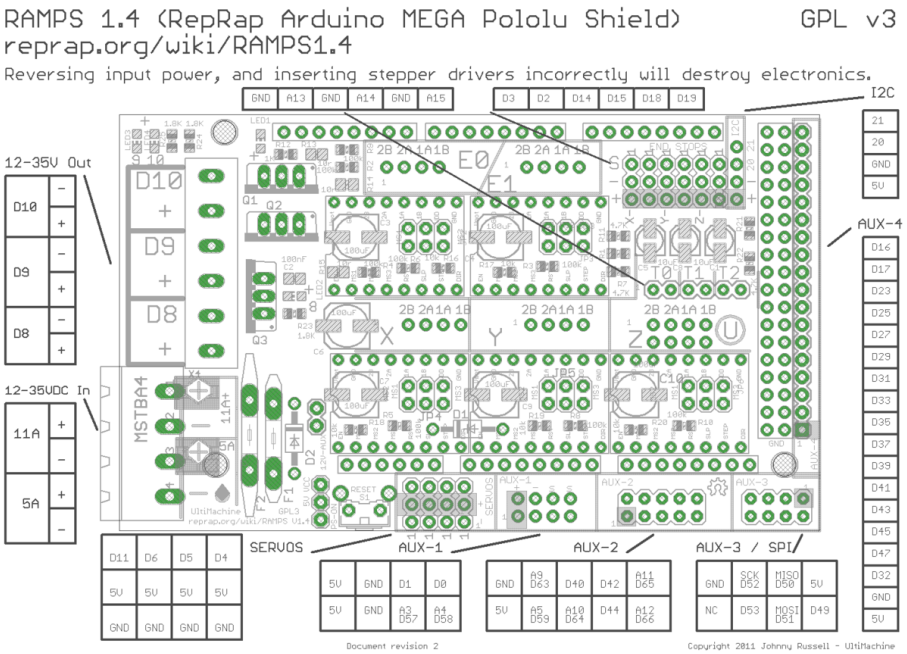
Obr. 2: Arduino Mega 2560
[https://www.vokolo.cz/arduino-mega-2560/]

2.2.2 Shield RAMPS 1.4

Shield RAMPS 1.4 je open-source plošná deska vyvinutá skupinou uživatelů Rep Rap, která obsahuje veškerou elektroniku potřebnou pro ovládání v kartézském souřadnicovém systému. Výhodou je úspora materiálu a místa. Deska plošných spojů je zapojena na externí napětí 12 V. [10]

Na obr. 5 je znázorněno schéma. Sekce X, Y, Z, E0, E1, konkrétně piny 2B, 2A, 1A, 1B, které slouží pro zapojení krokových motorů. Ve verzi 1.4 má osa Z k dispozici 8 pinů, takže zde můžeme připojit dva motory. U 3D tiskáren se piny E0 a E1 využívají pro připojení krokových motorů, které do extruderů dodávají materiál. Pod těmito oblastmi se nachází 6 pinů, konkrétně u X nese název JP4. Piny se dají spojit propojkami, kde nám jejich kombinace určí velikost kroku, nejmenší hodnota kroku je určena druhem stepper driveru, u A4988 je to 1/16 kroku a u DRV8825 1/32 kroku. Šestnáct pinů, které ohraničují piny s propojkami, slouží pro zapojení stepper driverů. [10]

Na desce dále lze nalézt piny pro připojení servomotorů, pro připojení koncových spínačů a T0, T1 a T2 sloužící pro připojení termistorů. Na řadu, jež se nachází na pravém kraji desky můžeme připojit LCD displej. Je zde také výstupní zdrojové napětí, které slouží u 3D tiskáren k výhřevu podložky, ohřevu tiskové hlavy a chlazení. Piny na spodní části desky se dají využít k zapojení SD karty a bluetooth modulu. [10]



Obr. 3: RAMPS 1.4 [10]

2.2.3 Stepper driver DRV8825

Driver DRV8825 slouží k ovládání bipolárních krokových motorů. Skládá se ze dvou H-můstků, které jsou tvořeny tranzistory typu MOSFET. Operuje s napětím 8,2 - 45 V. Při vstupním napětí 24 V a při teplotě 25 °C je výstupem na pinech AOUT1, AOUT2, BOUT1 a BOUT2 proud o maximální velikosti 2,5 A, který se dá regulovat potenciometrem, jenž je zabudovaný v driveru. Tento druh driveru je vytvořen opět komunitou Rep Rap a lze jej zapojit přímo do shieldu RAMPS 1.4. [11]

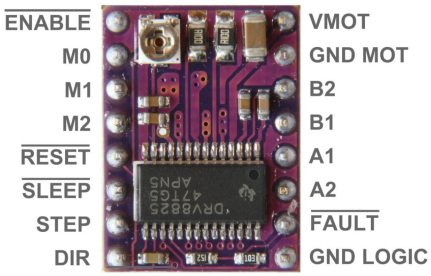
DRV8825 má v sobě zabudovaný tzv. Microstepping indexer, což umožňuje nastavení mikrokrokování až do 1/32 původního kroku. Ke zjemnění kroku poslouží piny Mode 1, Mode 2 a Mode 3. [11]

MODE2	MODE1	MODE0	STEP MODE
0	0	0	Full step (2-phase excitation) with 71% current
0	0	1	1/2 step (1-2 phase excitation)
0	1	0	1/4 step (W1-2 phase excitation)
0	1	1	8 microsteps/step
1	0	0	16 microsteps/step
1	0	1	32 microsteps/step
1	1	0	32 microsteps/step
1	1	1	32 microsteps/step

Obr. 4: Mikrokrokování
[https://aip.scitation.org/doi/abs/10.1063/1.5097484?journalCode=apc]

Pin STEP impulzy ovládá krok motoru. Pin DIR ovládá směr rotace hřídele motoru, který v případě rotace v jednom směru můžeme připojit pouze na VCC nebo GND. Driver má několik ochranných prvků, mezi ty nejdůležitější patří tepelná a proudová ochrana. [11]

Pokud i po nastavení mikrokrokování není chod motoru dostatečně plynulý, v kombinaci se stepper driverem se používá TL smoother. [12]



Obr. 5: DRV8825
[https://electrobes.com/product/drv8825-stepper-motor-driver-with-heat-sink/]

2.3 Aktuátory

2.3.1 Krokový motor

Krokový motor je vcelku přesný synchronní stroj, který nemá zpětnou vazbu. Využívá se zejména kvůli jeho přímé závislosti otáčecí rychlosti na frekvenci proudu a poloze na počtu impulzů proudu. Je pro něj typická hlučnost, a jelikož jeho funkčnost zajišťuje magnetické pole, podléhá magnetickým vlivům. [13]

Krokové motory dělíme z hlediska konstrukce na krokové motory s proměnnou reluktancí a krokové motory s aktivním rotorem. [13]

Krokové motory s proměnnou reluktancí

Tento typ motoru se využívá v aplikacích, kde si vystačíme s nižší přesností polohování. Konstrukce motoru je velmi jednoduchá. Rotor motoru je tvořen svazkem plechů s pólovými nástavci a zuby. Stator je tvořen z vyniklých pólů, na kterých jsou navinuté cívky. Na pólech statoru se také nachází pólové nástavce a zuby. Podle počtu fází motoru je určen počet vyniklých pólů. Protilehlé cívky jsou zapojeny do jedné fáze. Zuby na rotoru jsou souosé pouze s jednou dvojicí pólů na statoru, ostatní jsou posunuty. [13], [14]

Motor pracuje na principu změny reluktance. Pokud na jednu fázi připojíme napětí, rotor se natočí do polohy, kde je reluktance minimální. Rotor se i při vychýlení snaží stabilizovat do ustálené polohy. Motor je minimálně třífázový. [13]

Krokové motory s aktivním rotorem

Krokový motor s aktivním rotorem je konstrukčně dražší a méně přesný. Pólové nástavce rotoru jsou tvořeny permanentními magnetickými póly. Jejich polarita se po obvodu střídá. Na rozdíl od rotoru je počet polů na statoru dvojnásobný. Polaritu pólů na statoru měníme napětím. Motor je dvoufázový. [13]

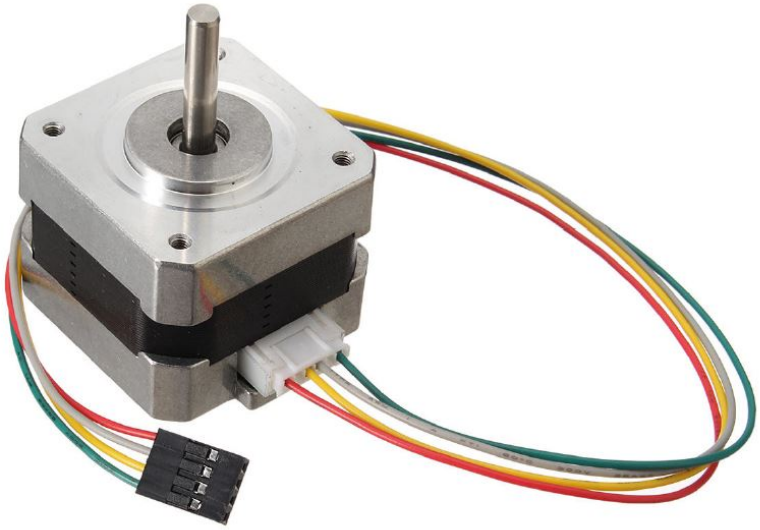
Hybridní motory

V dnešní praxi se nejčastěji používají hybridní motory, které z hlediska dělení spadají pod krokové motory s aktivním rotorem. Mají nejlepší momentové parametry s nejmenší velikostí kroku. [13]

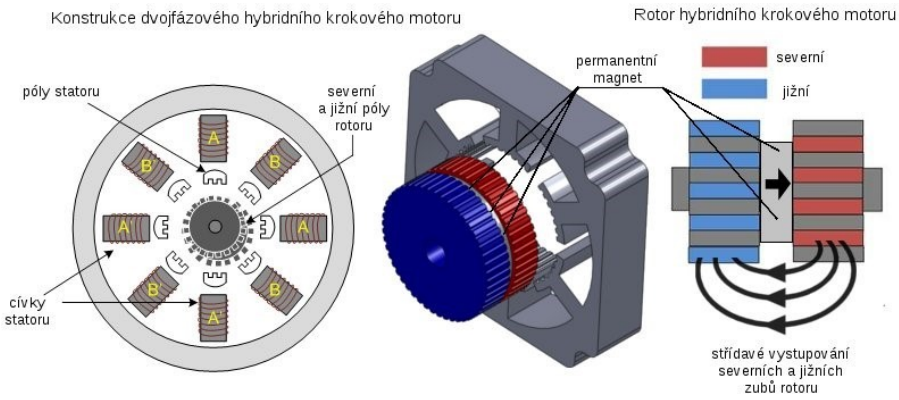
Rotor motoru se skládá z nemagnetické hřídele uložené v kuličkových ložiskách, kde vzniká jediné mechanické opotřebení. Na hřideli jsou nalisovány dva pólové nástavce, mezi nimiž je umístěn permanentní magnet. Pólové nástavce jsou vůči sobě posunuty o polovinu rozteče zubu. Stator tvoří pólové nástavce s cívkami, které jsou napájeny stejnosměrnými impulzy proudu a vytváří tak magnetické pole. Počty zubů na statoru a rotoru si nejsou rovné. Tak jako u motorů s proměnnou reluktancí je lícována pouze část zubů na statoru a rotoru. Při

napájení cívky se vytváří magnetická síla a rotor se nastaví do pozice nejmenší energetické náročnosti. Stepper driverem měníme polaritu. [13]

Krokové motory lze také dělit z hlediska řízení na unipolární a bipolární. V této práci využívám drivery A4988 a DRV8825, všechny motory jsou tedy řízeny bipolárně.



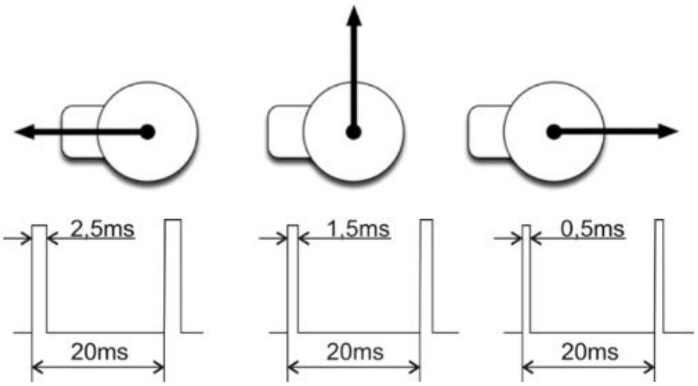
Obr. 6: Bipolární krokový motor
[https://www.banggood.com/42mm-12V-Nema-17-Two-Phase-Stepper-Motor-For-3D-Printer-p-1164619.html?cur_warehouse=CN]



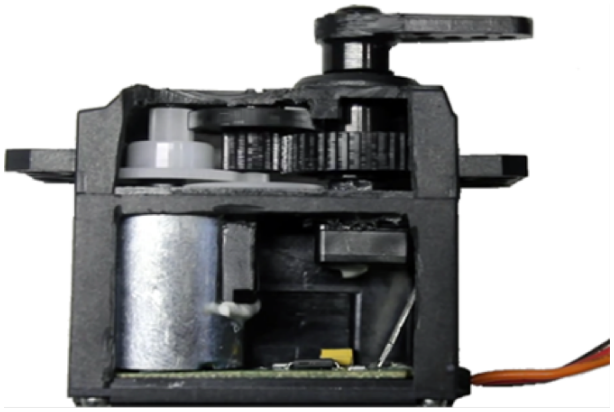
Obr. 7: Konstrukce dvojfázového hybridního krokového motoru
[https://www.servo-drive.cz/%C4%8Dasto_pokl%C3%A1dan%C3%A9_ot%C3%A1zky_o_krokov%C3%BDch_motorech.php]

2.3.2 Servo motor

V této práci se zabývám pouze servo motorem pro modelářské využití. Servo motor otáčí páčku v rozsahu od 0° do 180°, což je omezeno potenciometrem. Do řídící elektroniky servo motoru je vysílán každých 20 ms impulz určité délky. Délka tohoto impulzu určuje odpor, který je porovnáván s velikostí odporu potenciometru a na základě porovnání vysílá do stejnosměrného motoru proud buď v kladném směru, nebo opačném směru. Čím více se natočení blíží velikosti impulzu, tím menší impulzy vysílá řídící elektronika do motoru. Tento typ serva operuje na 5 V a neustále drží svou polohu. Na konci rozsahu jsou pak dorazy, které brání protočení servo motoru. [15]



Obr. 8: Řízení servo motoru [15]



Obr. 9: Struktura servo motoru [15]

2.4 Software

2.4.1 Marlin firmware

Marlin je open-source firmware, který byl vytvořen pro jednoduché ovládání 3D tiskáren. Firmware je kompatibilní s řadou řídících desek, včetně Arduina a jeho shieldu RAMPS 1.4, a řada firem, které produkují 3D tiskárny tento firmware využívá. Mezi nejznámější patří např. Prusa Research nebo Ultimaker. Hlavním důvodem jeho oblíbenosti je, že funguje i na velmi laciných mikrokontrolérech. [16]

Marlin je široce rozšířený, má i vlastní databázi desek, která má definované piny. Po nahrání firmwaru se mechanismus řídí příkazy G-kódu, kterých Marlin umí přes 150. S G-kódy se setkáme např. při ovládání obráběcích strojů. [16]

G-kód (M-kód)	Funkce G-kódu (M-kódu)
G00	Lineární interpolace - rychloposuv
G01	Lineární interpolace - pracovní posuv
G02	Kruhová interpolace - po směru hodinových ručiček
G03	Kruhová interpolace - proti směru hodinových ručiček
G28	Auto Home
G90	Absolutní souřadnicový systém
G91	Relativní souřadnicový systém
M280	Ovládání servomotoru

Tab. 1: Základní G-kód příkazy a příkaz pro ovládání servo motoru

Marlin obsahuje několik souborů (sekcí), z kterých uživatele zajímá configuration.h a configuration adv.h. Nastavit se dá téměř vše, ať už rychlosti a akcelerace krokových motorů, počet kroků na hodnotu G-kód příkazu nebo kinematika mechanismu. Marlin podporuje kinematiku v kartézské soustavě, delta kinematiku nebo mechanismy řízené metodou SCARA. [16]

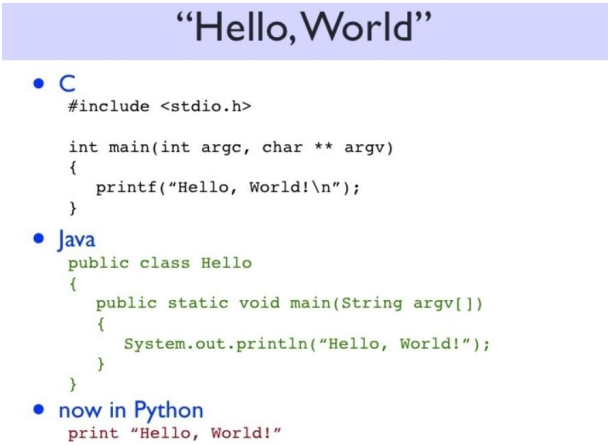
K projektu je uvolněna upravená verze firmwaru, která je přizpůsobena přímo robotické ruce. Je vynecháno např. kontrolování teplotních čidel či senzory filamentu.

2.4.2 Python

Python patří dnes mezi nejrozšířěnější programovací jazyky. Jeho obliba mezi uživateli stále roste, zejména díky jeho jednoduchosti, pochopitelnosti a čitelnosti. Patří do skupiny dynamicky typovaných jazyků, což umožňuje absenci deklarování datových typů. Jazyk je víceúrovňový, takže program zpracovává blíže k mysli člověka než počítače. [17]

Python vznikl již v roce 1991 a v roce 2008 byla vydána jeho třetí verze. Jazyk je multiplatformní, jeho užití je tedy vhodné pro širokou škálu operačních systémů. Stejný program lze spustit jak na Windows, tak Linuxu nebo Mac OS. [17] [18]

Jazyk je velmi expresivní, takže k naprogramování nám postačí méně řádků, než například u jazyků Java nebo C. Zde je porovnání zmiňovaných jazyků a program "Hello World". [17]



Obr. 10: "Hello World"- Porovnání jazyka Python, Java a C
[https://www.quora.com/Which-is-easier-to-learn-Java-or-Python]

Mezi další výhody Pythonu patří bezesporu široká škála dostupných knihoven a modulů.

Knihovna, modul	Použití
pandas	Analýza dat
NumPy	Vektory, matice a pole
Django	Web
random	Náhodná čísla
math	Matematická knihovna
tkinter	GUI
wxPython	GUI
Kivy	GUI
PyQt5	GUI

Tab. 2: Základní knihovny a moduly pro tvorbu GUI

tkinter

Modul tkinter pro tvorbu grafického rozhraní tvoří vazbu mezi Pythonem a Tk knihovnou, která byla vytvořena v jazycích C a Tcl. Pokud se nepožaduje multiplatformní využití, nabízí se i použití knihoven Ttk a Tix. Hlavní výhodou vytváření GUI v tkinteru je beze sporu jednoduchost a editor IDLE, který zvýrazňuje psanou syntaxi. Výsledkem programu je pak grafické okno, které má celkem zastaralý vzhled. Pokud se používá pouze knihovna Tk, je značné omezení sad ovládacích prvků. V porovnání s ostatními dostupnými možnostmi modul nenabízí žádný tzv. UI builder a výsledný kód zabere mnohem více řádků. Modul není vhodný pro tvorbu solidnějších aplikací. Pro komerční použití je k dispozici zdarma. [17]

wxPython

wxPython je tzv. obal knihovny wxWidgets, která je psaná v C++. Stejně jako předchozí modul, je multiplatformní. Vytváření GUI je velice snadné a jednoduché, práce se dá usnadnit použitím UI builderu, a to pomocí aplikací wxFormBuilder a wxDesigner. Aplikace vytvoří kód, který stačí otevřít v jazyce Python. Po vizuální stránce GUI vypadá slušně a lze jej použít i pro tvorbu složitějších a robustních rozhraní. Aplikace i moduly jsou také k dispozici zdarma, je zde ovšem nutná instalace modulu wxPython. [19], [20]

Kivy

Kivy je moderní framework, který se používá zejména pro vývoj mobilních aplikací. Vzhledem k jeho multiplatformnímu využití jej lze aplikovat také pro Windows, Mac OS nebo Linux se ztrátou jeho vícedotekového ovládání. Kivy je napsán v jazyce Python a Cython, takže na rozdíl od tkineru, wxPythonu a PyQt5 není žádným obalem nebo vazbou. Framework je open-source. [21]

PyQt5

PyQt5 je pátou verzí vazby mezi Pythonem a knihovnou Qt, která je vyvinuta v prostředí jazyka C++. PyQt5 je multiplatformní, lze ho tak použít jak pro vývoj desktopových, tak i pro vývoj mobilních aplikací. Aplikací knihovny Qt je široká škála a pro tvorbu složitějších aplikací přijde vhod UI Builder, tzv. Qt Designer. Vytvořené grafické rozhraní působí moderně a jeho tvorba je slušně zdokumentována. Nástrojů pro usnadnění práce existuje více, například Qt Creator, který je zpoplatněný. [20], [22]

Pro vývoj v Pythonu použitím knihovny Qt lze použít i vazbu PySide.

3 Praktická část

3.1 Mechanické sestavení

3.1.1 3D tisk

Robotická ruka při převzetí této práce byla po stránce mechanického sestavení hotova. Bohužel, po bližším přezkoumání zde bylo několik konstručních chyb, které byly tak zásadní, že bylo nutné robotickou ruku rozebrat a některé díly nahradit. Součásti, které byly potřeba nahradit byly vytištěny na 3D tiskárně technologií FFF/FDM.

V laboratoři byla k dispozici tiskárna Creality CR-10S PRO a bílá PLA tisková struna o průměru 1,75 mm. Parametry 3D tisku byly nastaveny tak, že části, které nesou vyšší zatížení, byly vytištěny s 35% výplní s hexagonovým vzorem, tzv. honeycomb. Pro části, které nejsou tolik mechanicky namáhány byla použita pouze 25% výplň, také s hexagonovým vzorem. Plná tloušťka materiálu byla na okrajích 1,2 mm. Použita byla tisková tryska o průměru 0,6 mm a výška jedné vrstvy byla pak 0,2 mm.

Obr. 11: Vytištěné díly, motory a další komponenty

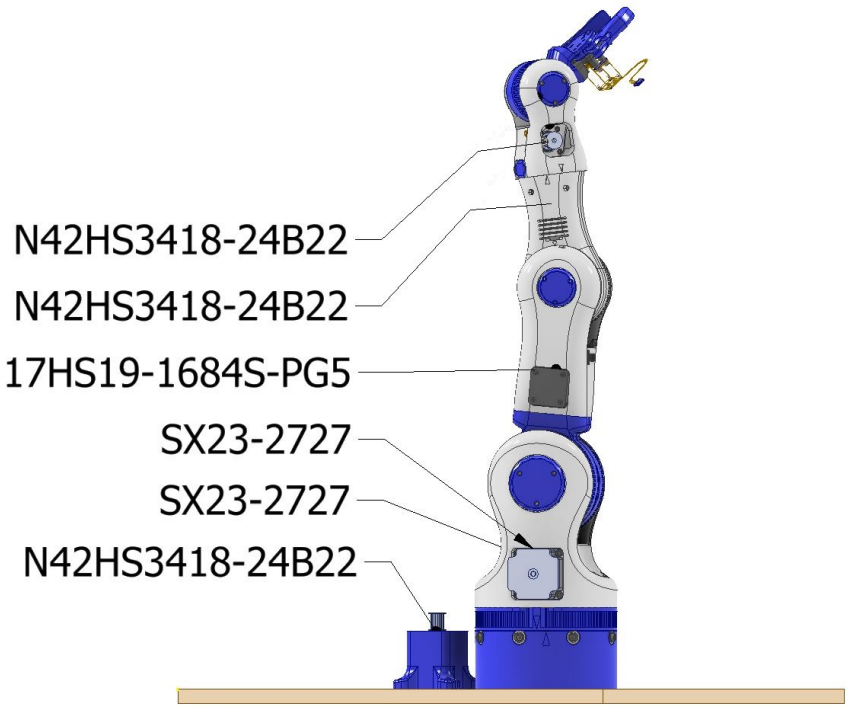
3.1.2 Sestavení robotické ruky

Kompletace mechanické ruky byla provedena podle návodu, jenž byl k projektu uvolněn. Během procesu se vyskytlo několik problémů, které byly způsobeny jinými druhy krokových motorů. Zde se nabízely dvě možnosti, buď úprava součástí a opětovné vytištění 3D modelů, nebo časově méně náročné frézování. Poté, co byly díly upraveny, byly motory upevněny do vytištěných součástí pomocí spojovacího materiálu. Jednotlivé osy byly zpřevodovány pomocí ozubených řemenic a ozubených řemenů profilu T05. Všechny řemeny byly nastaveny do správného předpětí pomocí napínáků.

Součástky, které byly potřeba k celkovému sestavení jsou uvedené v příloze.

Název motoru	SX23-2727	N42HS3418-24B22	17HS15-1684S-PG5
Statický moment [Nm]	2,7	0,5	2,3
Délka kroku [°]	1,8	1,8	0,35
Jmenovitý proud [A] - sér. / par. zapojení	2,7 / 5,4	1,5	1,68
Indukčnost [mH] - sér. / par. zapojení	6,4 / 1,6	50	3,2
Odpor [ohm] - sér. / par. zapojení	1,5 / 0,375	2,5	1,65
Hmotnost [kg]	1,18	0,24	0,55

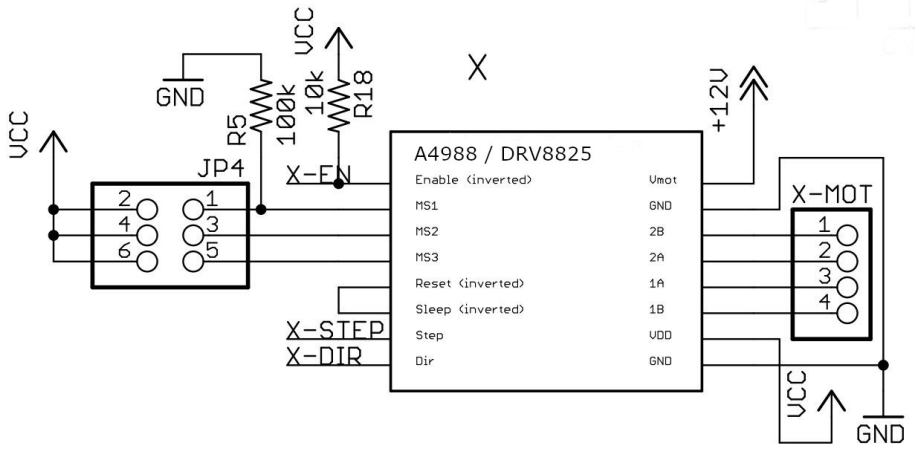
Tab. 3: Charakteristika jednotlivých krokových motorů



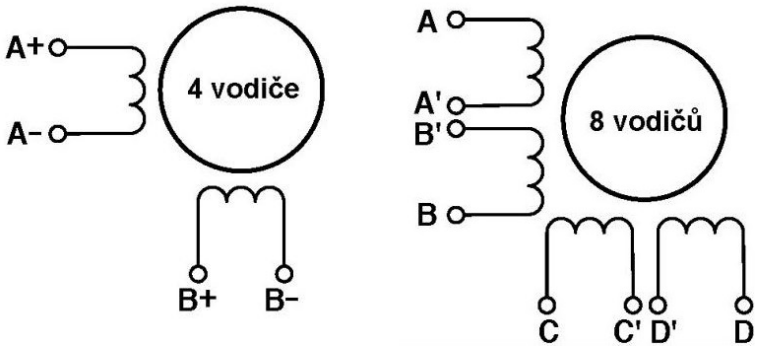
Obr. 12: Umístění krokových motorů v programu Autodesk Inventor 2020

3.2 Elektronické zapojení

Nejprve byl shield RAMPS 1.4 spojen s deskou Arduino Mega 2560. Do oblasti JP4 byly aplikovány jumpery, které propojily piny 1, 2, dále 3, 4 a 5, 6. Do shieldu RAMPS 1.4 byly postupně aplikovány stepper drivery. Protože driverů DRV8825 bylo nedostatek, na X, Y a E1 osu byly použity drivery A4988. Tyto osy budou mít tedy pouze 1/16 kroku.



Obr. 13: Elektrické schéma driverů a RAMPS 1.4 [10]



Obr. 14: Elektické schéma motorů NEMA 17 a SX23-2727
[http://robodoupe.cz/2020/krokove-motory-ii/]

Do pinů v oblasti X-MOT byly zapojeny motory, stejný způsob zapojení krokových motorů platí i pro osu Y, E0, E1.

Vinutí	Pin
A+	1
A-	2
B+	3
B-	4

Tab. 4: Zapojení krokových motorů do shieldu RAMPS 1.4

Motor SX23-2727 je zapojen do osy Z sériově. To znamená, že vodiče A´ a B´ a vodiče C´ a D´ jsou spojeny. Dále už je zapojení identické. Druhý motor do osy Z musíme připojit v opačném pořadí kvůli opačnému směru rotace.

Vinutí	Pin
A	1/4
B	2/3
C	3/2
D	4/1

Tab. 5: Zapojení krokových motorů osy Z do shieldu RAMPS 1.4

Deska RAMPS 1.4 byla připojena na zdroj napětí 12 V. Dále bylo potřeba upravit dodávaný proud do jednotlivých motorů pomocí potenciometru na stepper driveru. Pro driver A4988 platí vztah:

$$I_{MAX} = \frac{V_{REF}}{8R_{CS}} \tag{1}$$

kde I_{MAX} je maximální hodnota jmenovitého proudu krokového motoru a R_{CS} je odpor stepper driveru. Pro driver DRV8825 platí vztah:

$$I_{MAX} = 2V_{REF} \tag{2}$$

kde I_{MAX} je maximální hodnota jmenovitého proudu krokového motoru. Potenciometrem bylo nastaveno napětí následovně:

Osa	V_{REF}
X	0,6 V
Y	0,67 V
Z	1,35 V
E0	0,75 V
E1	0,6 V

Tab. 6: Nastavené napětí pomocí potenciometrů na driverech

Servo motor byl připojen do RAMPS 1.4 pomocí pinů D11, 5V a GND. Protože servo motoru napětí Arduina nestačilo, pomocí jumperu byly propojeny piny VCC a 5V.

3.3 Úprava firmwaru Marlin

Autoři projektu ovládali robotickou ruku pomocí odlišných driverů. Nezbytná byla tedy úprava firmwaru Marlin. V souboru configuration.h byla přepnuta základní deska na RAMPS 1.4 pro dva extrudery a bed.

Zdrojový kód 1: Marlin - boards.h - řádek 13 - 16, Marlin - Configuration.h - řádek 46 - 48

```
#define BOARD_RAMPS_13_EFB 33
// RAMPS 1.3 / 1.4 (Power outputs: Extruder, Fan, Bed)
#define BOARD_RAMPS_13_EEB 34
// RAMPS 1.3 / 1.4 (Power outputs: Extruder0, Extruder1, Bed)
#define BOARD_RAMPS_13_EFF 35
// RAMPS 1.3 / 1.4 (Power outputs: Extruder, Fan, Fan)
#define BOARD_RAMPS_13_EEF 36
// RAMPS 1.3 / 1.4 (Power outputs: Extruder0, Extruder1, Fan)

#ifndef MOTHERBOARD
  #define MOTHERBOARD 34
#endif
```

Dále bylo třeba definovat 2 extrudery.

Zdrojový kód 2: Marlin - Configuration.h - řádek 58

```
#define EXTRUDERS 2
```

Nakonec bylo třeba poupravit nastavení pohybu. Bylo potřeba přidat pátou osu a poupravit sekci DEFAULT_AXIS_STEPS_PER_UNIT. Přepočet byl proveden na 1° pootočení z následujícího vztahu:

$$\alpha = \frac{1}{\beta \cdot m \cdot i_{1,2}} \tag{3}$$

kde α je počet kroků, kdy jednotlivá osa se natočí o 1°, β délka kroku u motoru, m mikrokrokování a $i_{1,2}$ převod mezi hnací a hnanou ozubenou řemenicí.

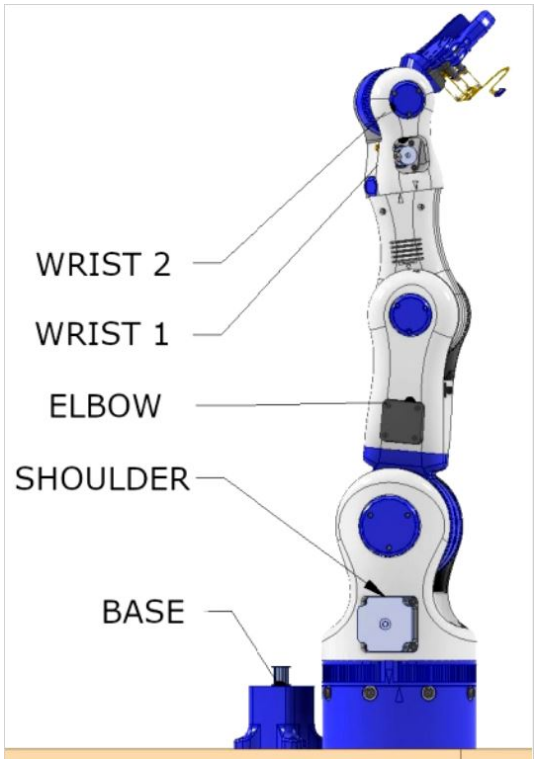
Zdrojový kód 3: Marlin - Configuration.h - řádek 477 - 478 a 482 - 484

```
#define NUM_AXIS 5
#define HOMING_FEEDRATE {50*60, 50*60, 4*60, 0, 0}
#define DEFAULT_AXIS_STEPS_PER_UNIT {8.88, 201.06, 101.6, 82.96, 93.63}
#define DEFAULT_MAX_FEEDRATE {500, 500, 500, 25, 25}
#define DEFAULT_MAX_ACCELERATION {60, 100, 5, 10000, 10000}
```

Po nastavení firmwaru Marlin bylo možné řidit robotickou ruku pomocí sériového monitoru. Nejprve bylo potřeba firmware nahrát na řídící desku Arduino po předem vybraném sériovém portu. Do otevřeného sériového monitoru poté stačilo zadat G-kód pro řízení krokových motorů a pro řízení servo motoru M-kód. Zde je uvedena tabulka s příkazy a rozsahy pro ovládání:

Pohyb jednotlivých os	G00	G01 (M280)	xxx (min. / max.)
Base Movement	T1, G00 Exxx	T1, G01 Exxx Fyyy	-135/135
Shoulder Movement	G00 Zxxx	G01 Zxxx Fyyy	-90/90
Elbow Movement	G00 Yxxx	G01 Yxxx Fyyy	-90/90
Wrist 1 Movement	G00 Xxxx	G01 Xxxx Fyyy	-180/180
Wrist 2 Movement	T0, G00 Exxx	T0, G01 Exxx Fyyy	-90/90
Gripper Control	/	M280 P0 Sxxx	0/120

Tab. 7: Ovládání jednotlivých krokových motorů



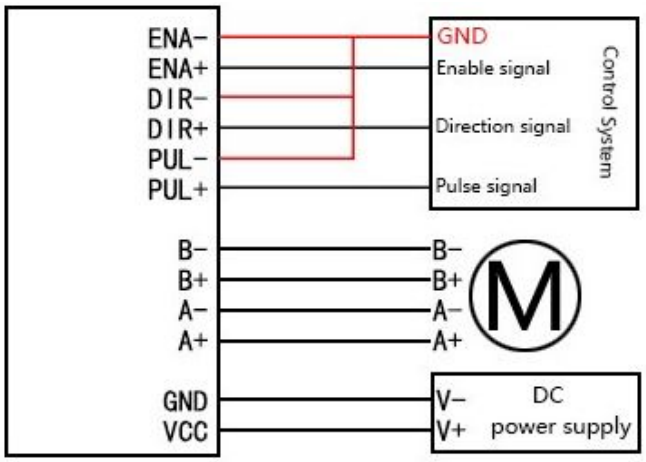
Obr. 15: Označení os robotické ruky

3.4 Úprava elektroniky

Rozpohybování robotické ruky bylo řízeno příkazy přes sériový monitor. Při prvních testech se ukázalo, že motory NEMA 23 nedisponovaly dostatečným výkonem, to bylo způsobeno nízkým napětím 12 V. Příčinou mohla být také zvýšená váha robotické ruky, kde původní motory byly nahrazeny těžšími. Tento problém lze vyřešit úpravou desky RAMPS 1.4, kdy všechny součásti s maximálním napětím do 24 V jsou vyměněny, u této verze pouze vratná pojistka a dioda v oblasti D1. Dalším řešením bylo externí napájení driverů.

Nakonec bylo rozhodnuto pro použití driverů TB 6600, které můžeme externě napájet s napětím 9 - 42 V. Výhodou těchto driverů je lepší odvod ztrátového tepla přes několikrát větší chladič, snadnější nastavení proudu a mikrokrokování pomocí DIP přepínačů. Tak jako původní drivery, nabízí mikrokrokování s maximální velikostí kroku 1/32. Maximální výstupní proud je pak 4 A. [23]

Původní drivery byly vyjmuty a TB 6600 byly zapojeny dle následujícího schématu.



Obr. 16: Schéma zapojení TB 6600, RAMPS 1.4 a krokového motoru
[http://habi.nowa.numap.mohammedshrine.org/tb6600-wiring-diagram.html]

Ze zdroje bylo odebíráno napětí o 24 V. RAMPS 1.4 byl stále napájen na 12 V, aby servo motor měl dostatečné napětí. Drivery byly nastaveny s mikrokrokováním 1/32. Proudy byly nastaveny podle již zmíněné tabulky č. 3. Dále bylo nutné přenastavit firmware Marlin podle již zmíněného vztahu (3).

3.5 Vizualizace GUI

Mezi širokým spektrem knihoven pro tvorbu GUI bylo rozhodnuto pro PyQt5, zejména kvůli jeho multiplatformní funkčnosti, slušnému vzhledu a také nástroji Qt Designer.

3.5.1 mainwindow.py

V programu Qt Designer byl vytvořen vzhled uživatelského rozhraní. Main Window, neboli hlavní okno, je složeno z několika rámců (tzv. frames). Rámce mezi sebou tvoří vazby a pokud dojde ke změně rozlišení, zobrazení se přizpůsobí.

V prvním rámci se nachází ovládání pro pohyb pátého kloubu. Obsahuje kruhovou stupnici (dial), posuvnou lištu (slider) a spin box pro zvolení požadované hodnoty rotace. Úhel, o který se má krokový motor pootočit, lze nastavit také pomocí tlačítek (push button) o 0.1°, 1° a 5°. Pro potvrzení zadaného pootočení je zde vytvořeno další tlačítko. V pravé části se vyskytují přepínače (radio button), která budou sloužit pro volbu rychloposuvu nebo lineárního posuvu. K úpravě rychlosti lineárního posuvu je k dispozici další spin box. Dále si zde můžeme všimnout textu (label), který je připraven pro zobrazení aktuální pozice. V levé části rozhraní jsou pro další klouby a upínač vytvořeny identické rámce.

V pravém horním rohu hlavního okna se nachází rámec nazvaný $Settings$ sloužící k základnímu nastavení. V něm se vyskytuje kombinované pole (combo box), které je připraveno pro zobrazení vyhledaných seriových portů. Pod ním nalezneme další kombinované pole pro zvolení symbolové rychlosti (baud rate). Tlačítko $Refresh$ je nachystáno pro vyhledávání seriových portů. Pro připojení rozhraní k sériovému portu bude sloužit tlačítko $Connect$. Stav připojení zobrazuje text, který je defaultně zobrazen na $Disconnected$.

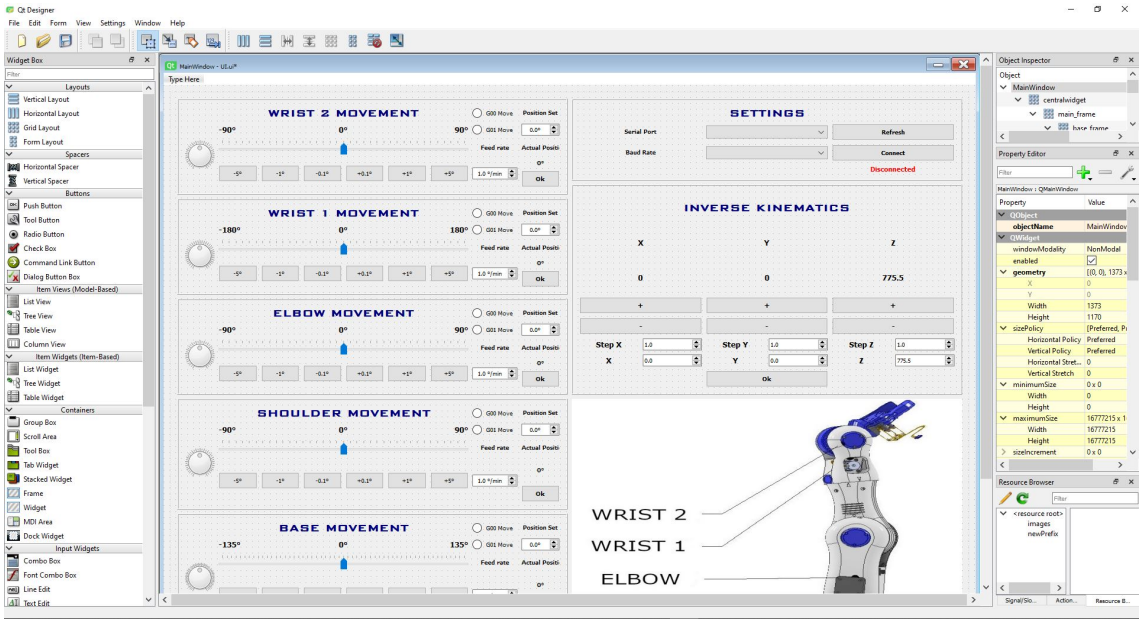
Pro ovládání robotické ruky je zde zaveden také rámec s inverzní kinematikou. Jsou zde nachystány čísla pro zobrazení aktuálních souřadnic z výpočtu přímé kinematiky (forward kinematics), tlačítka pro ovládání posuvu o jednotlivý krok, který se dá nastavit pomocí spin boxu. Rozhraní také nabízí možnost zadání přesných souřadnic, k tomu slouží další spin box. Tlačítko Ok bude sloužit pro potvrzení zadaných hodnot. Pod sekcí inverzní kinematiky je přidán JPEG s popisem jednotlivých os robotické ruky.

Kvůli přehlednosti byly definovány názvy všem objektům v rozhraní. Formát souboru ui byl převeden na py pomocí příkazu v terminálu .

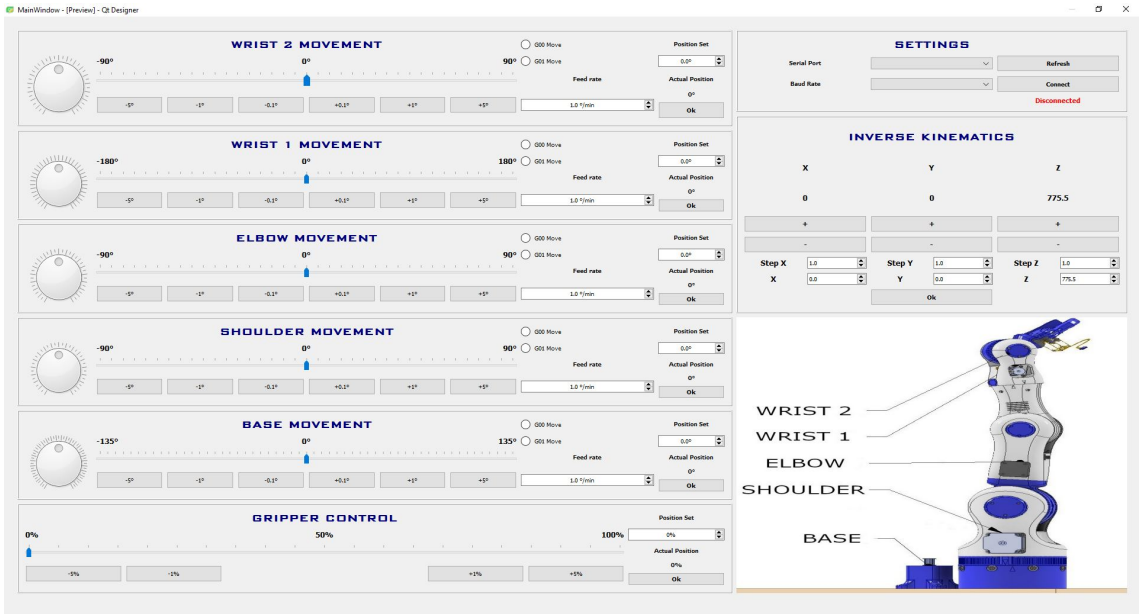
Zdrojový kód 4: Terminál

```
python -m PyQt5.uic.pyuic -x ui.ui -o mainwindow.py
```

Bylo nutné mít naimportované knihovny PyQt5 a pyuic5-tool.



Obr. 17: Ukázka vývojového prostředí v programu Qt Designer



Obr. 18: Vzhled rozhraní vytvořený v programu Qt Designer

Zdrojový kód 5: mainwindow.py - řádek 905 - 949

```
self.gridLayout_5.addWidget(self.base_label, 0, 1, 1, 6)
self.base_dial = QtWidgets.QDial(self.base_frame)
self.base_dial.setCursor(QtGui.QCursor(QtCore.Qt.ClosedHandCursor))
self.base_dial.setMinimum(-135)
self.base_dial.setMaximum(135)
self.base_dial.setInvertedAppearance(False)
self.base_dial.setInvertedControls(False)
self.base_dial.setWrapping(False)
self.base_dial.setNotchTarget(5.0)
self.base_dial.setNotchesVisible(True)
self.base_dial.setObjectName("base_dial")
self.gridLayout_5.addWidget(self.base_dial, 1, 0, 4, 1)
self.base_label0 = QtWidgets.QLabel(self.base_frame)
self.base_label0.setAlignment(QtCore.Qt.AlignCenter)
self.base_label0.setObjectName("base_label0")
self.gridLayout_5.addWidget(self.base_label0, 1, 3, 1, 2)
self.base_pushbutton1 = QtWidgets.QPushButton(self.base_frame)
self.base_pushbutton1.setCursor
(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.base_pushbutton1.setObjectName("base_pushbutton1")
self.gridLayout_5.addWidget(self.base_pushbutton1, 3, 1, 2, 1)
self.base_slider = QtWidgets.QSlider(self.base_frame)
self.base_slider.setCursor(QtGui.QCursor(QtCore.Qt.SizeHorCursor))
self.base_slider.setMinimum(-135)
self.base_slider.setMaximum(135)
self.base_slider.setSingleStep(1)
self.base_slider.setProperty("value", 0)
self.base_slider.setOrientation(QtCore.Qt.Horizontal)
```

V této ukázce kódu je definován rámec $Base\ Movement$. V první části je přidána kruhová stupnice s definovaným rozsahem, viditelnou stupnicí o kroku 5° a definovaným názvem $base\_dial$. Následně je zobrazen text s názvem $base\_label0$ a tlačítko $base\_pushbutton1$. V poslední části kódu je pak definována posuvná lišta s rozsahem, posuvným krokem a prvotně nastavenou hodnotou 0°. Při najetí na tyto objekty se kurzor myši mění.

3.6 Kinematika robotické ruky

Kinematika spadá pod obor mechaniky, zabývá se pohybem těles, ale ne jejich příčinami. Při řízení robotické ruky existují dva druhy úloh. První typ úlohy se nazývá přímá kinematika. Hledáme souřadnice koncového bodu ze známé geometrie robotické ruky a natočení jednotlivých os. Druhým typem je pak kinematika inverzní, známe polohu koncového bodu a hledáme řešení natočení os.

3.6.1 Denavit Hartenbergova transformace pro popis přímé kinematiky

Denavit-Hartenbergova úmluva je hojně využívaná pro popis kinematiky sériových manipulátorů. Patří mezi nejčastěji používanou úmluvu, zejména kvůli tomu, že převádí souřadnicový systém jednoho ramene na rameno druhé pomocí 4 parametrů. Mezi další úmluvy, které nám popisují souřadnicový systém pomocí 4 parametrů patří Khalil-Kleinfingerova úmluva.

Parametr	Popis
θ	rotace kolem osy z
d	translace v ose z
a	translace v ose x
α	rotace kolem osy x

Tab. 8: Popis parametrů Denavit-Hartenbergovy úmluvy

Parametry se popíší následujícími maticemi:

$$T_{Rz}(\theta_n) = \begin{pmatrix} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

$$T_z(d_n) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

$$T_x(a_n) = \begin{pmatrix} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

$$T_{Rx}(\alpha_n) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

Pro přesun ze souřadnicového systému $n-1$ na systém n vynásobíme předchozí 4 matice:

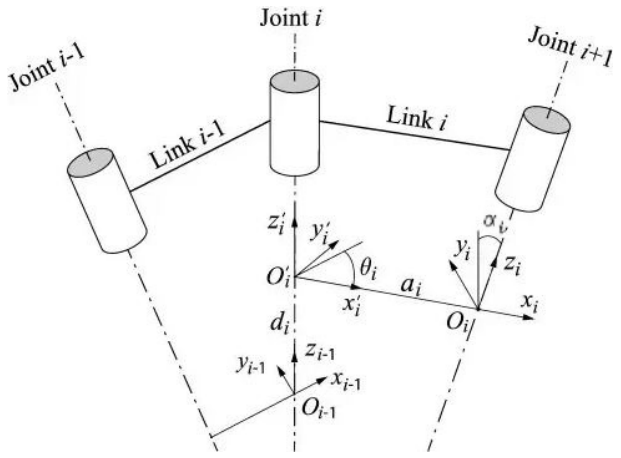
$$T_{n-1}^{n} = T_{Rz}(\theta_n) \cdot T_z(d_n) \cdot T_x(a_n) \cdot T_{Rx}(\alpha_n) \tag{8}$$

$$T_{n-1}^{n} = \begin{pmatrix} \cos\theta_n & -\sin\theta_n \cdot \cos\alpha_n & \sin\theta_n \cdot \sin\alpha_n & r_n \cdot \cos\theta_n \\ \sin\theta_n & \cos\theta_n \cdot \cos\alpha_n & -\cos\theta_n \cdot \sin\alpha_n & r_n \cdot \sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

Výpočet souřadnic koncového bodu je výsledkem součinu jednotlivých transformací z předchozího ramene na rameno následující:

$$T_0^n = \prod_{i=1}^{n} T_{i-1}^{i} \tag{10}$$

K vysvětlení stanovení parametrů použiji obrázek 19:

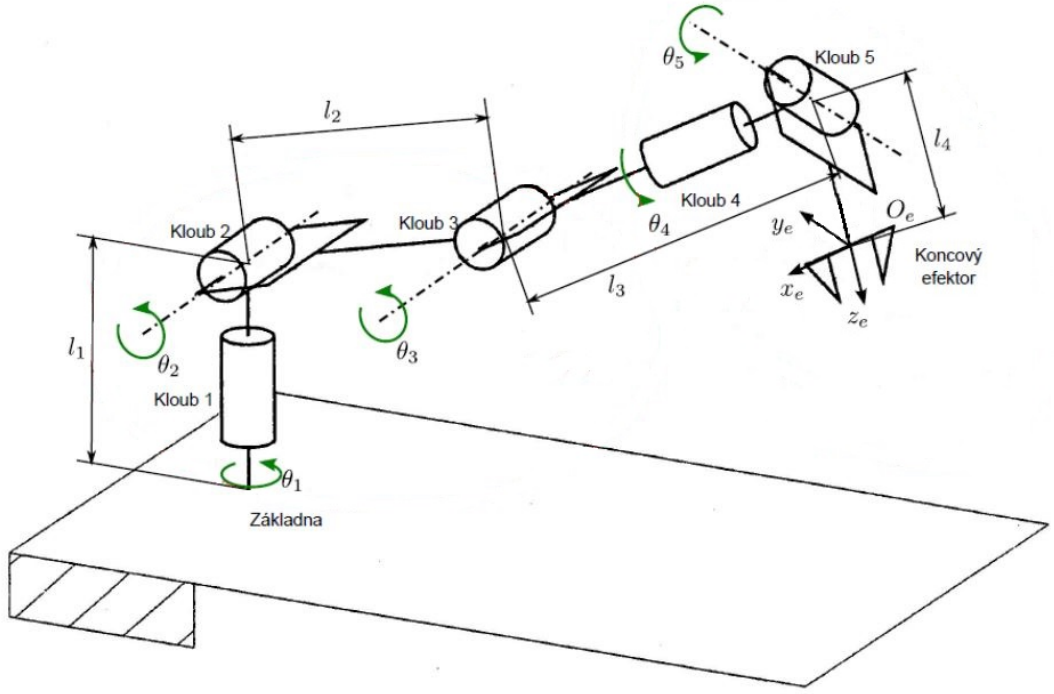


Obr. 19: Transformace souřadnicového systému pomocí Denavit Hartenbegovy úmluvy [https://www.quora.com/Robotics-What-is-the-best-resource-to-understand-Denavit%E2%80%93Hartenberg-parameters]

- Jako osu z_{i-1} označíme osu rotačního kloubu i, osu z_i označíme jako osu rotačního kloubu $i+1$.
- Počátek souřadnicového systému O_i bude umístěn na průniku osy z_i a normále os z_i a z_{i-1}.
- Zvolíme osu x_i tak, aby směřovala z kloubu i do kloubu $i+1$
- Osu y_i doplníme tak, aby byl souřadný systém pravotočivý. Pomoc si můžeme

pravidlem pravé ruky, kde palec bude osa z_i, ukazováček osa x_i a prostředníček nám ukáže osu y_i. Prsty mezi s sebou musí svírat 90°.

Následující obrázek 20 zobrazuje BCN3D Moveo, který má 5 stupňů volnosti.



Obr. 20: D-H parametry BCN3D Moveo, přepracováno na základě [https://otik.zcu.cz/bitstream/11025/7430/1/kaukal_diplomka.pdf]

Pomocí předchozí teorie a obrázku jsem určil následující parametry. Proměnná θ_2 a θ_3 je navýšena o $\pi/2$ kvůli zachování maximální vzdálenosti v ose z při nulovém natočení. θ_1 je záporná, aby byla dodržena konvence s možností manuální volby natočení této osy. Rozměry jsou v tabulce uvedeny v milimetrech.

Kloub	θ	d	a	α
1	$-\theta_1$	231,5	0	$\pi/2$
2	$\theta_2 + \pi/2$	0	221,5	0
3	$\theta_3 + \pi/2$	0	0	$\pi/2$
4	θ_4	224,5	0	$-\pi/2$
5	θ_5	0	0	$\pi/2$
Koncový efektor	0	98	0	0

Tab. 9: Denavit Hartenbergovy parametry pro popis souřadnicového systému robotické ruky BCN3D Moveo

Zdrojový kód 6: functions.py - řádek 404 - 425 a 462 - 473

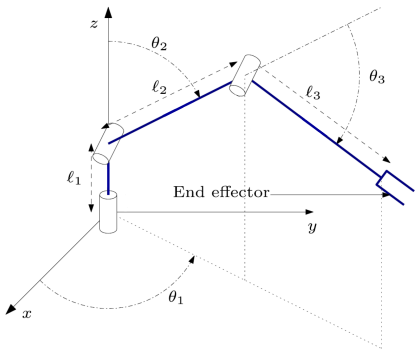
```
# Forward Kinematics
def forwardkinematics(self):

    theta1rad = -(self.base_spinbox_2.value() / 180) * np.pi + np.pi
    theta2rad = (self.shoulder_spinbox_2.value() / 180) * np.pi + np.pi / 2
    theta3rad = (self.elbow_spinbox_2.value() / 180) * np.pi + np.pi / 2
    theta4rad = (self.wrist1_spinbox_2.value() / 180) * np.pi
    theta5rad = (self.wrist2_spinbox_2.value() / 180) * np.pi

    dhmatrix = [[theta1rad, 231.5, 0, np.pi / 2],
                [theta2rad, 0, 221.5, 0],
                [theta3rad, 0, 0, np.pi / 2],
                [theta4rad, 224.5, 0, -np.pi / 2],
                [theta5rad, 0, 0, np.pi / 2],
                [0, 98, 0, 0]]

    t01 = [[np.cos(dhmatrix[0][0]),
            -np.sin(dhmatrix[0][0]) * np.cos(dhmatrix[0][3]),
            np.sin(dhmatrix[0][0]) * np.sin(dhmatrix[0][3]),
            dhmatrix[0][2] * np.cos(dhmatrix[0][0])],
           [np.sin(dhmatrix[0][0]),
            np.cos(dhmatrix[0][0]) * np.cos(dhmatrix[0][3]),
            -np.cos(dhmatrix[0][0]) * np.sin(dhmatrix[0][3]),
            dhmatrix[0][2] * np.sin(dhmatrix[0][0])],
           [0, np.sin(dhmatrix[0][3]),
            np.cos(dhmatrix[0][3]), dhmatrix[0][1]],
           [0, 0, 0, 1]]

    t02 = np.dot(t01, t12)
    t03 = np.dot(t02, t23)
    t04 = np.dot(t03, t34)
    t05 = np.dot(t04, t45)
    t06 = np.dot(t05, t56)

    self.x_pos_label.setText(str(round(t06[0][3], 1)))
    self.x_pos_label.setFont(QtGui.QFont("Ms Shell Dlg 2", 10,
    weight=QtGui.QFont.Bold))
    self.y_pos_label.setText(str(round(t06[1][3], 1)))
    self.y_pos_label.setFont(QtGui.QFont("Ms Shell Dlg 2", 10,
    weight=QtGui.QFont.Bold))
    self.z_pos_label.setText(str(round(t06[2][3], 1)))
    self.z_pos_label.setFont(QtGui.QFont("Ms Shell Dlg 2", 10,
    weight=QtGui.QFont.Bold))
```

Matice $t12, t23, t34, t45, t56$ jsou definovány obdobně.

3.6.2 Inverzní kinematika

Robotická ruka v průmyslové praxi je řízena nejčasteji pomocí inverzní kinematiky. Jak již bylo zmíněno, ze zadaného bodu zjišťujeme jednotlivé natočení kloubů. Výsledkem je poté několik řešení, která nemusí být kvůli geometrii robota uskutečnitelná. Omezeni jsme například rozsahem natočení jednotlivých kloubů a délce ramen. V uživatelském rozhraní jsem implementoval řešení inverzní kinematiky pro 3 stupně volnosti. Znamená to tedy, že poslední 2 klouby, konkrétně Wrist 1 a Wrist 2 jsou při výpočtu inverzní kinematiky v poloze $\theta_4 = 0$ a $\theta_5 = 0$. Při této podmínce jsem mohl inverzní kinematiku vyřešit pouze analyticky. Pokud bychom chtěli vyřešit inverzní kinematiku pro 5 stupňů volnosti, museli bychom postupovat takto:

- Nejprve dekompozitovat první 3 klouby a poslední dva klouby. První tři klouby by sloužily k polohování a poslední dva pro orientaci kleští.
- Analyticky bychom vyřešili inverzní kinematiku prvních třech kloubů.
- Vypočítali bychom přímou kinematiku prvních tří kloubů a zjistili tak transformační matici T_0^3, z které bychom extrahovali rotační část matice R_0^3.
- Našli bychom inverzní matici k R_0^3.
- Vypočítali bychom přímou kinematiku posledních dvou kloubů a zjistili tak transformační matici T_4^5, z které bychom extrahovali rotační část matice R_4^5.
- Určili bychom, jakou chceme výslednou rotační matici R_0^5.
- Ze zadaných souřadnic bychom vyřešili jednotlivé natočení prvních třech kloubů.
- Pomocí známých natočení θ_1, θ_2, θ_3 a použití rotační matice bychom zjistili jednotlivé natočení θ_4, θ_5.



Obr. 21: Zjednodušení robotické ruky na 3 stupně volnosti pro analytické řešení [https://www.mdpi.com/2076-3417/9/17/3516/htm]

Pro výpočet natočení prvních třech kloubů byly použity vztahy, které byly odvozeny pomocí trigonometrie. Upraveny jsou tak, aby při nulovém natočení kloubů robotická

ruka dosahovala maximální hodnoty osy z. Koriguji také znaménkovou konvekci, aby byla dodržena se stanovenou konvekcí u přímé kinematiky.

$$\theta_1 = -\arctan\left(\frac{y}{x}\right) \tag{11}$$

$$r = \sqrt{x^2 + y^2} \tag{12}$$

$$d = \sqrt{(z_1 - l_1)^2 + r^2} \tag{13}$$

$$\theta_3 = \arccos\left(\frac{d^2 - l_3^2 - l_2^2}{2l_3}\right) \tag{14}$$

$$\theta_2 = \arctan\left(\frac{r}{z - l_1}\right) - \arctan\left(\frac{l_2 + l_3 \cdot \cos\left(-\theta_3\right)}{l_3 \cdot \sin\left(-\theta_3\right)}\right) \tag{15}$$

Výpočet inverzní kinematiky je omezený jednotlivými rozsahy natočení. Při výpočtu, kdy je θ_1, θ_2, θ_3 mimo mez, text v sekci inverzní kinematiky se změní na $Theta\ not\ found$. Pokud je výpočet v mezích, nastaví se jednotlivé kruhové stupnice, posuvné lišty a spin boxy. Poté stačí potvrdit jednotlivé osy a robotická ruka se přesune do vypočteného bodu.

Zdrojový kód 7: functions.py - řádek 475 - 525

```
# Inverse Kinematics
def inversekinematics(self):
    x = self.x_spinbox.value()
    y = self.y_spinbox.value()
    z = self.z_spinbox.value()

    if theta1 * 180 / np.pi > -90 and theta1 * 180 / np.pi < 90:
        if (theta2 + np.pi / 2) * 180 / np.pi > -90 and (theta2 + np.pi / 2)
          * 180 / np.pi < 90:
            if theta3 * 180 / np.pi > -90 and theta3 * 180 / np.pi < 90:
                self.elbow_spinbox_2.setValue(round(theta3 * 180 / np.pi, 1))
                self.base_spinbox_2.setValue(round(theta1 * 180 / np.pi, 1))
                self.shoulder_spinbox_2.setValue(round((theta2 + np.pi / 2)
                * 180 / np.pi, 1))
                self.wrist2_spinbox_2.setValue(round(theta5 * 180 / np.pi, 1))
                self.wrist1_spinbox_2.setValue(round(theta4 * 180 / np.pi, 1))
            else:
                self.z_label.setText("Theta 3 not found")
        else:
            self.y_label.setText("Theta 2 not found")
    else:
        self.x_label.setText("Theta 1 not found")
```

Z ukázky kódu bylo vynecháno formátování textů a výpočetní vztahy.

3.7 Funkce GUI

3.7.1 serialport.py

Tento skript byl vytvořen pro vyhledávání dostupných sériových portů. Je naprogramován tak, že je opět multiplatformní. Výsledkem tohoto skriptu je seznam, který je poté nabízen v kombinovaném poli.

Zdrojový kód 8: serialport.py - převzato z [https://stackoverflow.com/questions/12090503/listing-available-com-ports-with-python]

```
import serial
import sys
import glob

def serial_ports():
    if sys.platform.startswith("win"):
        ports = ["COM%s" % (i+1) for i in range(256)]
    elif sys.platform.startswith("linux")
         or sys.platform.startswith("cygwin"):
        ports = glob.glob("/dev/tty[A-Za-z]*")
    elif sys.platform.startswith("darwin"):
        ports = glob.glob("/dev/tty.*")
    else:
        raise EnvironmentError("Unsupported platform")

    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result

if __name__ == '__main__':
    print(serial_ports())
```

3.7.2 functions.py

Funkčnost grafického rozhraní má na starost skript functions.py. V hlavičce programu je naimportováno několik knihoven a provázanost mezi ostatními skripty.

Zdrojový kód 9: function.py - řádek 1 - 10

```
from PyQt5 import QtWidgets
from PyQt5 import QtGui
from mainwindow import Ui_MainWindow
import sys
import serial
import serial_port as sp
import numpy as np
import time

ser = serial.Serial()
```

Při spuštění skriptu se vyvolá funkce $serialportcombobox$, která začne do kombinovaného pole ukládat dostupné sériové porty. Stisknutím tlačítka $Refresh$ se tato funkce vyvolá znovu.

Zdrojový kód 10: functions.py - řádek 107 - 113

```
# Serial port
def serialportcombobox(self):
    self.serialport_combobox.clear()
    self.serialport_combobox.addItems(sp.serial_ports())
    self.serial_label.setText("Disconnected")
    self.serial_label.setStyleSheet("QLabel { color : red }")
    self.serial_label.setFont(QtGui.QFont("Ms Shell Dlg 2", 7,
    weight=QtGui.QFont.Bold))
```

Následně je nutné zvolit $Baud\ rate$, firmware Marlin je nastaven na hodnotu 250 000. Kliknutím na tlačítko Ok v sekci $Settings$ je spuštěna funkce $serialportconnection$.

Zdrojový kód 11: functions.py - řádek 115 - 134

```
def serialportconnection(self):
    serialport = self.serialport_combobox.currentText()
    baudrate = self.baudrate_combobox.currentText()
    if serialport != "":
        ser.port = serialport
        ser.baudrate = baudrate
        ser.timeout = 1
        try:
            ser.close()
            ser.open()
```

```
            self.serial_label.setText("Connected")
            self.serial_label.setStyleSheet("QLabel { color : green }")
            bytes = ser.inWaiting()
            ser.read(bytes)
        except:
            self.serial_label.setText("No connection")
            self.serial_label.setStyleSheet("QLabel { color : red }")
    else:
        self.serial_label.setText("Error - Serial value")
        self.serial_label.setStyleSheet("QLabel { color : red }")
```

Kruhová stupnice, posuvná lišta, spin box a tlačítka jsou vždy v jedné sekci mezi s sebou provázany. Při potvrzení tlačítka Ok, například v sekci $Wrist2\ Movement$, se vyvolá funkce $wrist2ok$.

Zdrojový kód 12: functions.py - řádek 165 - 182

```
def wrist2ok(self):
    if ser.isOpen():
        if self.wrist2_radiobutton1.isChecked():
            move = "G00 " + "E" + str(self.wrist2_spinbox_2.value()) + "\n"
        else:
            move = "G01 " + "E" + str(self.wrist2_spinbox_2.value()) + " F"
                    + str(round(self.wrist2_spinbox.value(), 1)) + "\n"
        ser.write("T0\n".encode())
        time.sleep(0.5)
        ser.write(move.encode())
        time.sleep(0.5)
        print(ser.readline().decode("ascii"))
        self.wrist2_labelactvalue.setText(str(self.wrist2_spinbox_2.value()))
        self.forwardkinematics()
    else:
        self.serial_label.setText("No connection")
        self.serial_label.setStyleSheet("QLabel { color : red }")
```

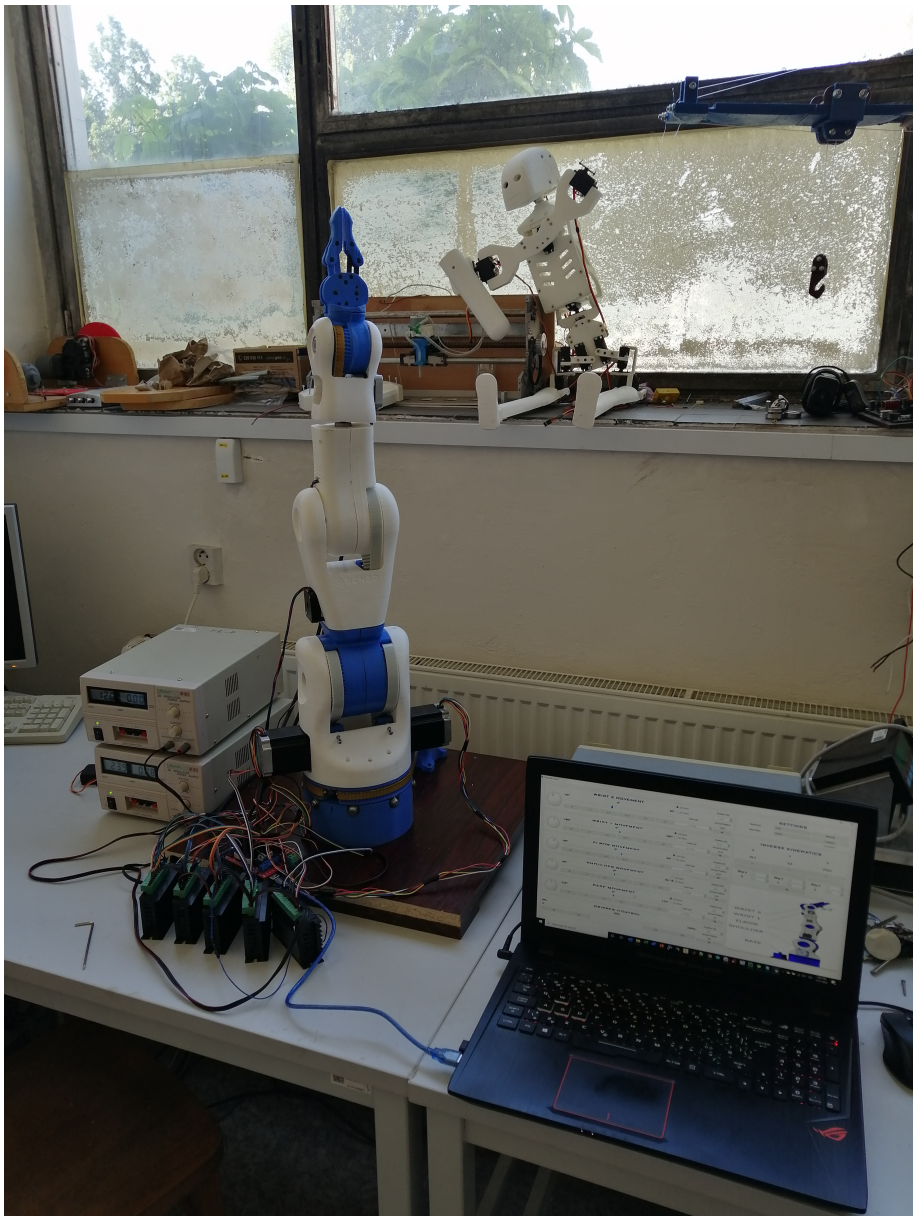
Všechny tyto funkce se nachází v $classAplication(Ui\_MainWindow)$.

Zdrojový kód 13: functions.py - řádek 19 - 22 a 551 - 556

```
class Application(Ui_MainWindow):
    def __init__(self, dialog):
        Ui_MainWindow.__init__(self)
        self.setupUi(dialog)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    application = QtWidgets.QMainWindow()
```

```
application.show()
prog = Application(application)
sys.exit(app.exec())
```

Tento skript byl úspěšně otestován na sestavené robotické ruce v laboratoři.



Obr. 22: Ovládání robotické ruky BCN3D Moveo

4 Závěr

První část bakalářské práce se zabývala představením projektu pětiosé robotické ruky BCN3D Moveo. Postupně byly představeny jednotlivé komponenty, jež jsou nezbytné pro sestavení robotické ruky. Dále byly sepsány programovací možnosti pro tvorbu grafického rozhraní a zmíněn firmware Marlin, který se používá pro řízení 3D tiskáren.

Nosné části robota byly vytištěny na 3D tiskárně Creality CR-10S PRO s tiskovou strunou PLA. Robotická ruka byla mechanicky sestavená dle návodu, jenž je k projektu uvolněn. Elektronika pro řízení byla zapojena a seřízena podle parametrů výrobce a jmenovitých proudů motorů. Drivery krokových motorů by se mohly umístit na desku robota, k nim by se mohlo přidělat přídavné chlazení. Následně bylo potřeba zkalibrovat a poupravit firmware Marlin tak, aby vyhovoval elektronickým součástkám.

Na základě předchozí rešerše byl zvolen programovací jazyk Python. Vizualizace grafického rozhraní byla navržena pomocí programu Qt Designer. Zde bylo vytvořených několik rámců, které byly ukotveny tak, aby se hlavní okno přizpůsobilo jakémukoliv rozlišení. Připraveny byly i spin boxy pro nastavení souřadnic bodu, které po potvrzení nastaví rotaci krokových motorů tak, aby se koncový bod robota dostal do námi zadaného bodu. Zobrazení aktuálních souřadnic je počítáno pomocí Denavit-Hartenbergovy úmluvy. Rozhraní s výpočtem přímé a inverzní kinematiky řídí robotickou ruku pomocí sériového portu a není tak nutné použít žádný další software. Součástí mého projektu je i skript, který automaticky vyhledává sériové porty.

Výsledkem mé práce je grafické prostředí, kterým lze ruku řídit. Všechny cíle práce byly splněny a grafické rozhraní bylo úspěšně vyzkoušeno na sestavené robotické ruce. Na toto rozhraní by mohlo být navázáno a bez větších problémů lze okno rozšířit, například o živé vysílání z přidělané kamery.

Závěrem práce bych chtěl přidat několik poznatků, které by nebylo špatné implementovat. První osa robota by se dala zpřevodovat pomocí uzavřeného ozubeného řemene, aby její rozsah byl 360°. To by sice nezlepšilo rozsah robotické ruky, ale zkrátila by se doba přejezdů. Na druhou stranu, při neopatrné manipulaci by mohlo dojít k poškození elektrického zapojení. Přidáním šesté osy rotace by nabízela větší počet možností ovládání koncového efektoru. S tím ale souvisí návrh nové řídící desky nebo použití PLC. Co se samotného rozhraní týče, nebylo by špatné přidat několik tlačítek pro zapamatování nastavených bodů a také příkazový řádek pro polohování motorů pomocí příkazů.

Seznam použitých značek a symbolů

1. PLC - Programmable Logic Controller (Programovatelný logický automat), str. 8
2. HMI - Human-Machine Interface (Rozhraní mezi člověkem a strojem), str. 9
3. CAD - Computer-Aided design (Počítačem podporované projektování), str. 9
4. STL - Stereolithography (Stereolitografie), str. 9
5. BOM - Bill of materials (Kusovník), str. 9
6. PWM - Pulse Width Modulation (Pulzně šířková modulace), str. 10
7. UART - Universal asynchronous receiver-transmitter (Univerzální asynchronní přijímač-vysílač), str. 10
8. ICSP - In-Circuit Serial Programming (Sériové programovaní mikrokontrolerů), str. 10
9. LED - Light-Emitting Diode (Elektroluminiscenční dioda), str. 10
10. SRAM - Static Random Access Memory (Statická paměť), str. 10
11. EEPROM - Electrically Erasable Programmable Read-Only Memory (Elektronicky vymazatelná paměť, pouze pro čtení), str. 10
12. RAMPS - RepRap Arduino Mega Shield, str. 11
13. LCD - Liquid-crystal display (Displej z kapalných krystalů), str. 11
14. SD - Secure Digital (Paměťová karta), str. 11
15. MOSFET - Metal-oxide-semiconductor field-effect transistor (Polem řízený tranzistor), str. 12
16. GND - Ground (Uzemnění), str. 12
17. VCC - Voltage Common Collector (Napájecí napětí), str. 12
18. SCARA - Selective Compliace Articulated Robot Arm (Selektivní kompatibilní kloubové robotické rameno), str. 16
19. GUI - Graphic User Interface (Grafické uživatelské rozhraní), str. 17
20. IDLE - Integrated DeveLopment Environment (Integrované vývojové prostředí), str. 18
21. UI - User Interface (Uživatelské rozhraní), str. 18
22. FFF - Fused Filament Fabrication (Nanesení roztaveného materiálu v tenké vrstvě), str. 19
23. FDM - Fused Deposition Modeling (Nanesení roztaveného materiálu v tenké vrstvě),

Seznam použité literatury a zdrojů

[1] *V Japonsku vznikla téměř plně robotizovaná rostlinná farma* [online]. 6D HUB, 2016 [cit. 2020-07-28]. Dostupné z: https://www.theguardian.com/environment/2016/feb/01/japanese-firm-to-open-worlds-first-robot-run-farm

[2] *Robotika* [online], poslední aktualizace 31. října 2019 13:06 [cit. 2020-07-28], Wikipedie. Dostupné z: https://cs.wikipedia.org/wiki/Robotika

[3] *Vývoj průmyslové robotizace v roce 2019 a 2020* [online]. Antonín Vojáček, 2019 [cit. 2020-07-28]. Dostupné z: https://automatizace.hw.cz/vyvoj-prumyslove-robotizace-v-roce-2019-a-2020.html

[4] *BCN3D Moveo - A fully Open Source 3D printed robot arm* [online]. BCN3D Technologies, 2020 [cit. 2020-02-03]. Dostupné z: https://www.bcn3d.com/bcn3d-moveo-the-future-of-learning/

[5] *BCN3D Moveo* [online]. kitusmark, 2016 [cit. 2020-02-03]. Dostupné z: https://github.com/BCN3D/BCN3D-Moveo/

[6] VODA, Zbyšek. *Průvodce světem Arduina*. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.

[7] *ARDUINO MEGA 2560 REV3* [online]. Arduino, 2020 [cit. 2020-02-04]. Dostupné z: https://store.arduino.cc/arduino-mega-2560-rev3/

[8] *Arduino a sériová komunikace* [online]. Vlastimil Slinták, 2011 [cit. 2020-02-04]. Dostupné z: https://uart.cz/139/arduino-a-seriova-komunikace/

[9] *Arduino v příkladech - III.díl - Jak pracovat s pamětí poprvé.* [online]. Jaroslav Boháč, 2017 [cit. 2020-02-04]. Dostupné z: https://www.arduinotech.cz/inpage/eeprom/

[10] *RAMPS 1.4* [online]. RepRap contributors, 2020 [cit. 2020-02-11]. Dostupné z: https://reprap.org/wiki/RAMPS_1.4

[11] *DRV8825 Stepper Motor Controller IC* [online]. Texas Instruments, 2014 [cit. 2020-02-13]. Dostupné z: http://www.ti.com/lit/ds/symlink/drv8825.pdf

[12] *TL-smoother, jak to funguje?* [online]. Miroslav Kováčik, 2018 [cit. 2020-02-13]. Dostupné z: https://3dfactory.cz/2018/09/04/tl-smoother-jak-to-funguje/

[13] UHLÍŘ, Ivan. *Elektrické stroje a pohony*. Vyd. 2., přeprac. Praha: Nakladatelství ČVUT, 2007. ISBN 978-80-01-03730-0.

[14] *Krokové motory* [online]. Hobbyrobot, 2013 [cit. 2020-02-16]. Dostupné z: http://profirobot.cz/wp-content/uploads/2018/03/Krokov%C3%A9-motory.pdf

[15] *Servo motor* [online]. Martin S., 2017 [cit. 2020-02-17]. Dostupné z: https://navody.arduino-shop.cz/arduino-projekty/servo-motor.html

[16] *What is Marlin?* [online]. Marlin Firmware, 2020 [cit. 2020-03-22]. Dostupné z: https://marlinfw.org/docs/basics/introduction.html

[17] SUMMERFIELD, Mark. *Python 3: výukový kurz*. Brno: Computer Press, 2010. ISBN 978-80-251-2737-7.

[18] *A Brief Timeline of Python* [online]. Guido van Rossum, 2009 [cit. 2020-03-22]. Dostupné z: http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

[19] *Overview of wxPython* [online]. The wxPython Team, 2020 [cit. 2020-03-26]. Dostupné z: https://wxpython.org/pages/overview/index.html

[20] *wxPython* Python Top 3 GUI Frameworks In 2019 (PyQt5, wxPython, TKinter). In: Youtube [online]. 26.12.2018 [cit. 2020-03-26]. Dostupné z: https://www.youtube.com/watch?v=0OdCAsC7g4I. Kanál uživatele Parwiz Forogh.

[21] *Představení Kivy frameworku a tvorba prvních aplikací* [online]. MQ, 2018 [cit. 2020-03-27]. Dostupné z: https://www.itnetwork.cz/python/kivy-framework-pro-python/predstaveni-kivy-frameworku-a-tvorba-prvnich-aplikaci

[22] *PyQt5 5.15.0* [online]. Riverbank Computing Limited, 2020 [cit. 2020-03-28]. Dostupné z: https://pypi.org/project/PyQt5/

[23] *TB 6600 Stepper Driver* [online]. Amazon, 2020 [cit. 2020-08-2]. Dostupné z: https://www.amazon.com/TB6600-Stepper-Driver-Controller-tb6600/dp/B07B9ZQF5D

Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek

Seznam použitého SW

– Texmaker, MiKTeX (LaTeX), Autodesk Inventor Professional 2020, Arduino, Python, Qt Designer

Seznam příloh