STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 9: Strojírenství, hutnictví a doprava

Univerzální robotický manipulátor

Autoři: Jakub Průcha, Josef Dušek
Škola: Gymnázium Vlašim, Tylova 271, 258 01 Vlašim
Kraj: Středočeský kraj
Konzultant: RNDr. Petra Surynková, Ph.D.
Vlašim 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 9: Strojírenství, hutnictví a doprava

Univerzální robotický manipulátor

Universal robotic manipulator

Autoři: Jakub Průcha, Josef Dušek
Škola: Gymnázium Vlašim, Tylova 271, 258 01 Vlašim
Kraj: Středočeský kraj
Konzultant: RNDr. Petra Surynková, Ph.D.
Vlašim 2025

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

Vlašim 21. března 2025 ……

Jakub Průcha, Josef Dušek

Poděkování

Na prvním místě bychom chtěli poděkovat všem sponzorům, kteří nás při vývoji finančně podpořili a umožnili nám projekt dotáhnout do konce. Jmenovitě děkujeme firmám Denso, Pastell, Mavel, Velteko, Viking Mašek, Prádelna Kyselý, LED servis a Hobysport Vlašim.
Dále bychom chtěli poděkovat RNDr. Petře Surynkové, Ph.D., která se ochotně stala naší konzultantkou.
Poslední poděkování patří Mgr. Olze Mládkové, jíž jsme vděční za kontrolu práce po gramatické stránce.

Anotace

V naší práci jsme se zabývali konstrukcí a řešením převodu pohybu v kartézském systému souřadnic pro 4osý robotický manipulátor. Cílem naší práce bylo sestrojit stolní robotický manipulátor vhodný pro vyučování na školách. Pro ovládání manipulátoru jsme vyvinuli vlastní software, řízený G-code příkazy. Ty náš program přepočítá na rotaci jednotlivých motorů tak, že dochází k zadanému pohybu. Dále jsme se zabývali vývojem nástrojů na robota, jako jsou kreslicí nástroj pro více barev nebo kleště pro manipulaci s předměty. Manipulátor i nadále vylepšujeme a vytváříme další verzi, která by měla dosáhnout vyšších přesností než stávající rameno. Zároveň ještě vyvíjíme grafické rozhraní pro snazší ovládání robota.

Klíčová slova

Robotický manipulátor, řemenový převod, kreslicí nástroj, manipulační kleště, G-code převodník, Python, Klipper, pohyb po přímce, inverzní kinematika

Annotation

In our work, we focused on the design and solution of motion transition in a Cartesian coordinate system for a 4axis robotic manipulator. The aim of our project was to construct the desktop robotic manipulator suitable for teaching purposes in schools. We developed custom software to control the manipulator, which is operated through G-code commands. Our program translates these commands into the rotation of individual motors, resulting in the desired movement. Furthermore, we worked on developing tools for the robot, such as a multi-color drawing tool and a gripper for handling objects. We continue to improve the manipulator and are working on the next version, which is expected to achieve higher precision than the current arm. At the same time, we are developing a graphical interface to make the robot easier to operate.

Keywords

Robotic manipulator, belt transition, drawing tool, manipulating gripper, G-code converter, Python, Klipper, linear motion, inverse kinematics

Obsah

1 Úvod

Od počátku nás fascinovalo fungování průmyslových manipulátorů. Chtěli jsme si tedy vytvořit něco podobného v menším měřítku pro vzdělávací účely na všeobecně zaměřených školách (např. naše Gymnázium Vlašim).

Inspirací se pro nás stala přednáška pana prof. RNDr. Pavla Surynka, Ph.D. týkající se jeho robotického manipulátoru RR1, kdy jsme si řekli: „To přeci zvládneme taky!"

Cílem bylo zkonstruovat 4osý robotický manipulátor, který bude schopen pohybu po přímce pomocí příkazu G-code[1]. Pro převod z G-code jsme si chtěli vytvořit vlastní software, který by vypočítával úhly natočení a rychlosti rotace pro jednotlivé krokové motory. Následně také vytvořit nástroje pro přesun předmětů (kleště) a pro kreslení několika barvami.

Problematice jsme chtěli do detailu porozumět. Pro lepší pochopení fungování robotického manipulátoru jsme tedy vytvořili program, který na základě pozic v kartézském systému souřadnic daných G-codem vypočítá pohyby jednotlivých motorů pro přesun nástroje po přímce z bodu A do bodu B. Myslíme si, že nás tento postup posunul k lepšímu a hlubšímu poznání problematiky pohybu manipulátoru a jsme schopni předat získané vědomosti zájemcům.

Navržený manipulátor by měl být v budoucnu využíván na našem gymnáziu ve výuce informatiky. Studenti by se díky němu měli naučit, jak lze matematické poznatky z oblasti goniometrických funkcí využívat v problematice robotiky a pohybu po přímce. Ovládání robota je navrženo tak, aby bylo uživatelsky přívětivé a srozumitelné i pro začátečníky.

2 Teorie

2.1 Stručná historie robotického ramene

Mezi první průmyslově používaná robotická ramena patřil robot Unimate (obr. 1), který byl představen na veřejnosti poprvé v roce 1961. Sestrojili ho George Devol a Joseph Engelberg, čímž si zasloužili přízvisko „otcové robotiky". První roboty byly hydraulicky ovládané, což jim dávalo poměrně velkou sílu, ale menší rychlost.

Z počátku roboty sloužily pro nahrazení těžké práce v obtížných nebo nebezpečných pracovních podmínkách (vysoké teploty, prach). Vynikaly v monotónních pohybech, které prováděly s mnohem větší přesností a opakovatelností než lidé.

Prvním plně elektrickým robotem se stal přístroj od společnosti ASEA jménem IRB 6 (obr. 2), představený roku 1974. Vybaven byl elektrickými servomotory, díky nimž dosahoval významně vyšší přesnosti a rychlosti pohybů.

V 90. letech došlo k aplikování pokročilých senzorů a strojového vidění. Robot mohl sám analyzovat svoje pohyby a hledat prostory, kde by mohl být efektivnější.

[1] G-code je nejrozšířenější programovací jazyk pro ovládání CNC a 3D tiskáren.

Začátkem milénia nastal další významný milník, kterým bylo zavedení umělé inteligence, která zásadně přispěla v oblasti flexibility a kooperace strojů. Pokročilé systémy umožnily vznik kolaborativních robotů, známých jako coboti (obr. 3), které byly koncipované pro spolupráci s lidmi. Díky spolehlivým senzorům jsou schopné sledovat lidi v jejich okolí a vyvarovat se pohybů, které by pro ně mohly být nebezpečné.

Obrázek 1: Unimate

Obrázek 3: cobot SWIFTI™ CRB 1300

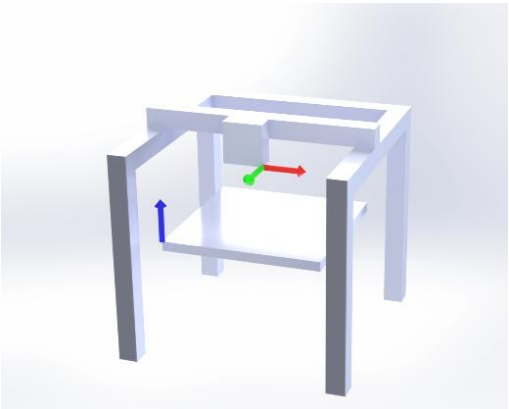
Obrázek 2: robot IRB 6

2.2 Druhy robotů podle konstrukce

Roboty se podle konstrukce dělí do několika kategorií. Každá má své výhody a nevýhody a je proto vhodná pro určité typy aplikací.

Kartézská konstrukce

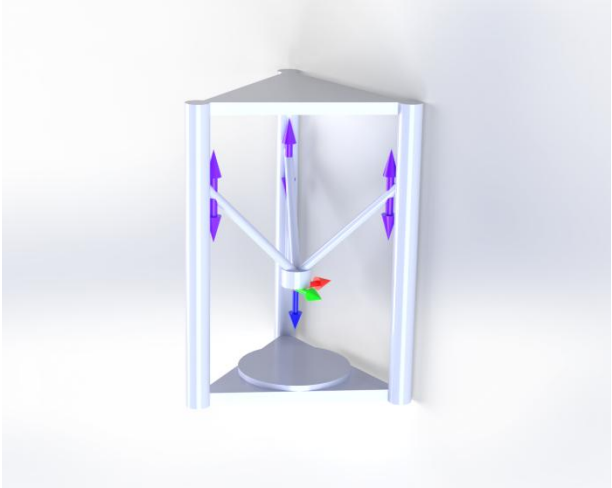
Roboty s kartézskou konstrukcí se vyznačují třemi lineárními pohony, které jsou na sebe vzájemně kolmé a umožňují pohyb nástroje v osách X, Y a Z. Mezi jejich přednosti patří velká tuhost a přesnost pohybu. Nevýhodou je obyčejně větší velikost v porovnání s pracovní oblastí.



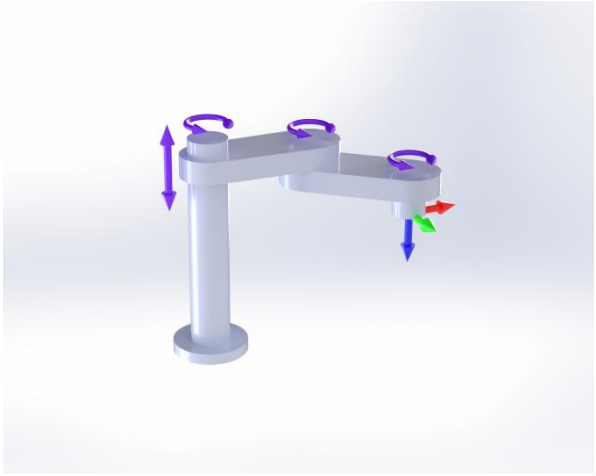
Obrázek 4: kartézská konstrukce

Delta

Konstrukce delta je založená na třech ramenech, které jsou ovládány svislými lineárními posuny po stranách. Ramena jsou umístěna pod tělem robota. Předností je vysoká rychlost a manévrovatelnost. Hlavním nedostatkem je nízká zatížitelnost a omezený pracovní prostor.



Obrázek 5: delta konstrukce



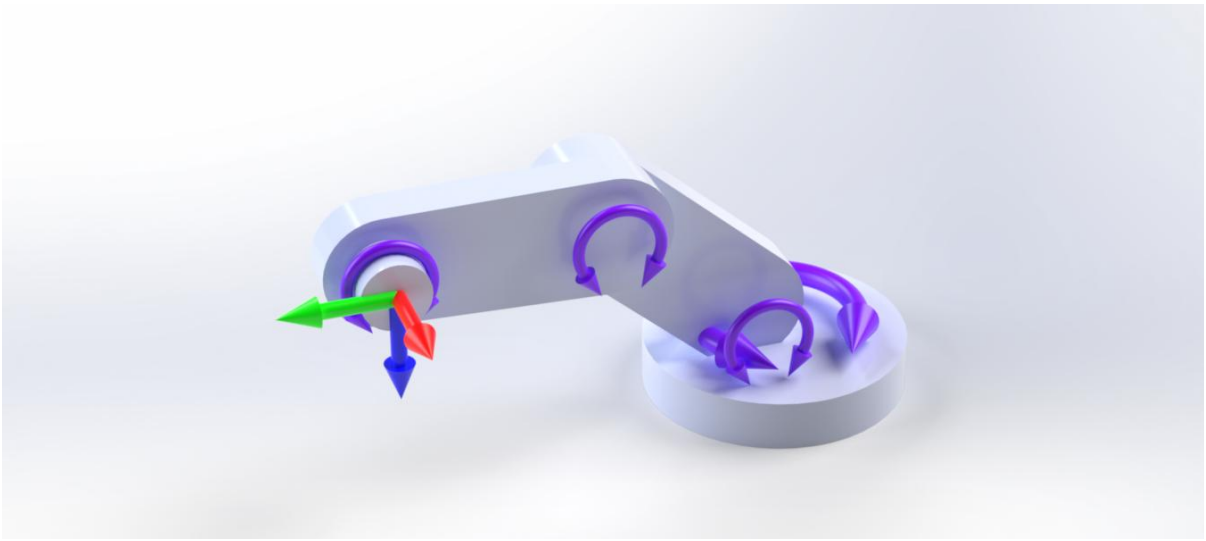
Obrázek 6: SCARA konstrukce

SCARA

Robot SCARA je specifický tím, že volnost v osách X a Y zajišťují dvě ramena napojená na obvykle válcové tělo, které pohybuje tělem robota lineárně v ose Z. Možnou nevýhodou jsou vysoké nároky na kvalitní podložku a pevné uchycení.

Kloubová konstrukce

Kloubová konstrukce se může pyšnit vysokou flexibilitou a velkým dosahem robota. Obyčejně se v průmyslu používá 6osý robot, ale v současné době se stále častěji objevují i 7osé roboty, u kterých vzniká více možností, jak se mohou dostat na cílovou pozici. To se může hodit například při vyhýbání se překážce. Nevýhodou je pak vyšší pořizovací cena a nižší rychlost.



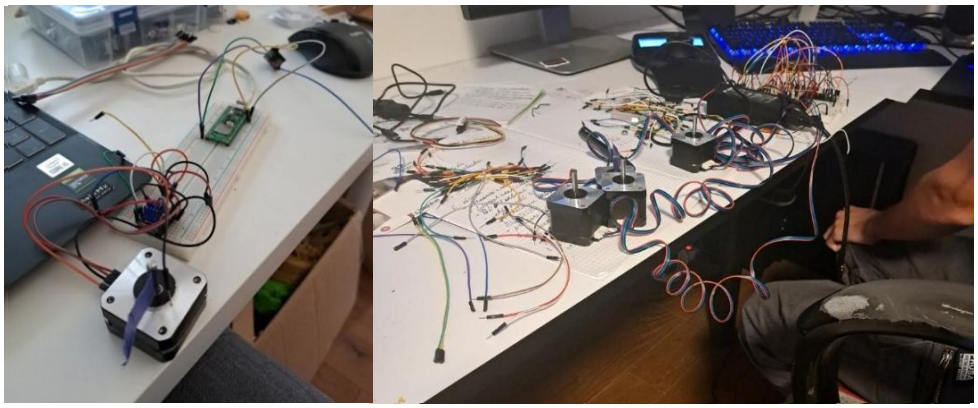
Obrázek 7: kloubová konstrukce

3 Vývoj softwaru

Software jsme chtěli vytvořit od úplného začátku, abychom co nejvíce pochopili fungování manipulátoru. Již z minulosti jsme měli zkušenosti s Pythonem, proto jsme se rozhodli v něm pokračovat.

3.1 Raspberry Pi pico s TMC2209 UART

Prvotní myšlenkou ovládání byl mikroprocesor Raspberry Pi pico 1W s kombinací driverů krokových motorů TMC2209 od BTT. Nejprve jsme zkoušeli zapojení na nepájivém poli (obr. 8), kde jsme sice dokázali ovládat až čtyři motory, ale neměli jsme nad nimi dostatečnou kontrolu. Nechtěli jsme se ale pouštět do vývoje vlastního softwaru pro ovládání rychlostí, akcelerací a souběžného chodu motorů. Proto jsme využili již existující řešení – Klipper, který je primárně navržený pro ovládání 3D tiskáren. Rozhodli jsme se pro něj, jelikož jsme s ním měli kladné zkušenosti z minulosti.



Obrázek 8: zapojení motorů k Raspberry Pi pico

3.2 Klipper manual stepper

Vzhledem k tomu, že Klipper nemá integrovanou kinematiku pro víceosé roboty, museli jsme využít řešení manual stepper, které ovládá jednotlivé motory pomocí příkazu manual stepper (kód 1). V tomto příkaze udáváme rychlost pohybu, akceleraci a cílovou pozici nebo to, zda má dojet na spínač. Důležitou součástí je příkaz SYNC=0/1. Pokud je zde hodnota 1, znamená to, že současně se vykoná pouze tento příkaz, nebo že se jedná o poslední příkaz ve skupině. Naopak hodnota 0 značí, aby byl s tímto příkazem zároveň vykonán i příkaz na dalším řádku.

Kód 1: vzor příkazu manual stepper

```
MANUAL_STEPPER STEPPER=config_name ENABLE=0/1 SET_POSITION=pos
SPEED=speed ACCEL=accel MOVE=pos STOP_ON_ENDSTOP=1/2 SYNC=0/1
```

V konfiguraci manual stepper (kód 2), se zadávají piny, ke kterým je připojený driver (TMC 2209). My jsme využili komunikaci skrze UART, díky čemuž jsme mohli jednoduše ovládat proudy motorů. Dále jsme zadali i převodový poměr, takže jej již nemusíme přepočítávat v převodníku.

Kód 2: konfigurace manual stepper Klipper

```
[manual_stepper alfa]
step_pin: PF11
dir_pin: PG3
enable_pin: !PG5
endstop_pin: PG6
microsteps: 16
rotation_distance: 1
gear_ratio: 1:10
```

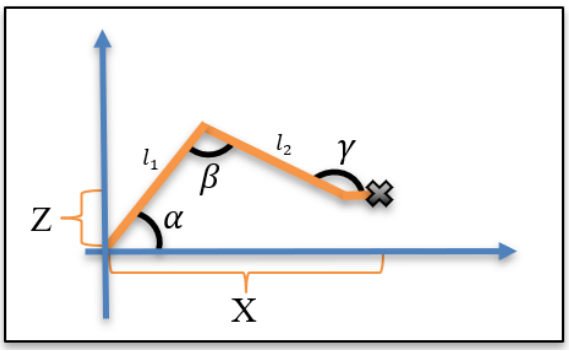
Využití manual stepper pro nás bylo průlomové, protože jsme mohli jednoduše pohybovat s jednotlivými motory a snadno generovat příkazy pro pohyb. Mezi nevýhody patří délka příkazu, tedy jeho datová velikost (tabulka 1).

Tabulka 1: srovnání velikosti příkazů G-code s manual stepper

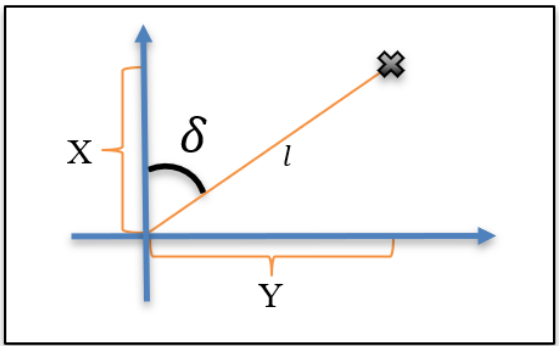
G-code příkaz	Manual stepper příkaz
G1 X10 Y50 Z10	MANUAL_STEPPER STEPPER=alfa MOVE=102.3 SPEED=0.5 SYNC=0 MANUAL_STEPPER STEPPER=beta MOVE=-151.4 SPEED=0.2 SYNC=0 MANUAL_STEPPER STEPPER=gama MOVE=10.6 SPEED=0.3 SYNC=0 MANUAL_STEPPER STEPPER=delta MOVE=78.6 SPEED=10 SYNC=1

3.3 Inverzní kinematika

Dopředná kinematika je soupisem vzorců, které ze zadaných úhlů v kloubech dopočítají koncovou souřadnici nástroje. Inverzní kinematika je opakem, tedy ze zadaných souřadnic vypočítává úhly natočení jednotlivých kloubů. Na obrázcích 9 a 10 můžeme vidět grafické znázornění jednotlivých úhlů.



Obrázek 10: znázornění úhlů α, β a γ
(pohled z boku)



Obrázek 9: znázornění úhlů δ
(pohled z vrchu)

Níže můžeme vidět vzorec pro výpočet úhlu δ ze souřadnic X a Y (obr.9).

$$\delta = \mathrm{tg}^{-1}\left(\frac{X}{Y}\right)$$

Následující rovnice popisuje vzdálenost nástroje od středu otáčení podstavy robota.

$$l = \sqrt[2]{X^2 + Y^2 + Z^2}$$

Další rovnicí již vypočítáme úhel natočení α (obr. 10), kdy l_1 je délka prvního článku robota a l_2 je délka druhého článku.

$$\alpha = \cos^{-1}\left(\frac{{l_1}^2 + l^2 - {l_2}^2}{2 * l_1 * l}\right) + \sin^{-1}\left(\frac{Z}{l}\right)$$

Rovnice sloužící k výpočtu úhlu β.

$$\beta = \cos^{-1}\left(\frac{{l_1}^2 + {l_2}^2 - l^2}{2 * l_1 * l_2}\right)$$

Poslední rovnice je určená k výpočtu úhlu γ.

$$\gamma = \alpha + \beta$$

V kódu 3 jsou již aplikované rovnice v Pythonu s přidaným výpočtem pro odsazení nástroje oproti středu otáčení Gamy.

Kód 3: přepočet α (Alfa), β (Beta), γ (Gama) a δ (Delta) + odsazení nástroje

```
def prepocet(X,Y,Z):
   global L1
   global L2
   global tool_ofset
   if X != 0:
      delta = (math.atan(Y/X))*(180/math.pi)
   else: delta=0
   X1=X-tool_ofset[0]*math.cos(delta/(180/math.pi))-
tool_ofset[1]*math.sin(delta/(180/math.pi))
   Y1=Y-tool_ofset[0]*math.sin(delta/(180/math.pi))-
tool_ofset[1]*math.cos(delta/(180/math.pi))
   L=math.sqrt(Z*Z+X1*X1+Y1*Y1)
   alfa =((math.acos((L1*L1+L*L-L2*L2)/(2*L1*L)))+(math.asin(Z/L)))*(180/math.pi)
   beta =(math.acos((L1*L1+L2*L2-L*L)/(2*L1*L2)))*(180/math.pi)+alfa*(12/60)-180
   gama = ((180-(beta+180-alfa*(12/60)))+(180-alfa))-180+beta*(14/36)
   return [alfa,beta,gama, delta]
```

Kód 4 je podprogramem, který z vypočtených úhlů a rychlostí daných motorů, vygeneruje příkaz pro manual stepper. Zároveň se program dělí podle toho, zda má dojít k synchronizaci s dalšími motory či nikoli. To udává konec příkazu SYNC=0/1

Kód 4: podprogram generující příkazy

```
def prikaz(motor, uhel, rychlost, synchronizace):
   if synchronizace:
      return "MANUAL_STEPPER STEPPER=" + motor + " MOVE="+str(uhel)+"
SPEED="+str(rychlost)+" ACCEL="+str(max_akcelerace)+" SYNC=0"
   else:
      return "MANUAL_STEPPER STEPPER=" + motor + " MOVE="+str(uhel)+"
SPEED="+str(rychlost)+" ACCEL="+str(max_akcelerace)+" SYNC=1"
```

3.4 Přepočet a pohyb po přímce

Abychom se co nejvíce přiblížili k pohybu po přímce, rozhodli jsme se rozdělit trasu z A do B na několik menších úseků. Bez tohoto opatření by se robot pohyboval po křivce, což je pro většinu našich aplikací nevhodné. Podle potřeby je možné přesnost pohybu změnit na základě jedné proměnné. Dělení probíhá do doby, dokud nejdelší úsek není kratší než hodnota maximální povolené délky/přesnosti (kód 5). S rostoucí přesností se zvyšuje doba výpočtu a samotného pohybu.

Kód 5: podprogram pro zjištění min. počtu dělení

```
def presnost(delka, max_pohyb):
   return round(delka/max_pohyb)
```

Pro pohyb po přímce je nutné, aby motory začaly a skončily svůj pohyb ve stejnou dobu. Vzhledem k tomu, že vykonávají různou rotaci, je nutné měnit i jejich rychlost. Motor, který má vykonat nejdelší pohyb, rotuje nejvyšší nastavenou rychlostí. Rychlosti ostatních motorů jsou ve stejném poměru, jako je poměr rotace, které mají motory vykonat. O tyto výpočty se stará kód 6.

Kód 6: podprogram vypočítávající rychlosti rotací

```
def rychlost(uhel_n, max_speed):
   global uhel_o
   uhel=[abs(uhel_n[0] - uhel_o[0]),abs(uhel_n[1] - uhel_o[1]),abs(uhel_n[2] -
uhel_o[2]),abs(uhel_n[3] - uhel_o[3])]
   uhel_o=uhel_n
   if uhel[0] > uhel[1] and uhel[0] > uhel[2] and uhel[0] > uhel[3]:
      return [max_speed,
max_speed/uhel[0]*uhel[1],max_speed/uhel[0]*uhel[2],max_speed/uhel[0]*uhel[3]]
   elif uhel[1] > uhel[0] and uhel[1] > uhel[2] and uhel[1] > uhel[3]:
      return [max_speed/uhel[1]*uhel[0],
max_speed,max_speed/uhel[1]*uhel[2],max_speed/uhel[1]*uhel[3]]
   elif uhel[2] > uhel[0] and uhel[2] > uhel[1] and uhel[2] > uhel[3]:
      return [max_speed/uhel[2]*uhel[0], max_speed/uhel[2]*uhel[1],
max_speed,max_speed/uhel[2]*uhel[3]]
   else:
      return [max_speed/uhel[3]*uhel[0],
max_speed/uhel[3]*uhel[1],max_speed/uhel[3]*uhel[2], max_speed]
```

3.5 Domovská pozice

Před každým programem se manipulátor dostane do výchozí pozice, kvůli možnosti vypnutí motorů nebo přeskočení kroků. Pro nalezení domovské pozice jsme si v Klipperu napsali macro, které zavoláme příkazem “home” (kód 7). Po zadání tohoto příkazu se manipulátor začne pohybovat podle určené sekvence, kde na krajních bodech pohybů jsou umístěny koncové spínače s fixní pozicí, které zaznamenají, když je rameno stiskne, tedy když se dostalo do koncové pozice.

Kód 7: macro pro zahomování

```
[gcode_macro home]
gcode:
  M84
  MANUAL_STEPPER STEPPER=gama ENABLE=0
  MANUAL_STEPPER STEPPER=gama SPEED=20 MOVE=150
  MANUAL_STEPPER STEPPER=beta ENABLE=0
  MANUAL_STEPPER STEPPER=beta ENABLE=1 SPEED=20 SET_POSITION=-200
MOVE=10 STOP_ON_ENDSTOP=-2
  MANUAL_STEPPER STEPPER=beta ENABLE=0
  MANUAL_STEPPER STEPPER=alfa ENABLE=0
  MANUAL_STEPPER STEPPER=alfa ENABLE=1 SPEED=10 SET_POSITION=120 MOVE=-3
STOP_ON_ENDSTOP=-2
  MANUAL_STEPPER STEPPER=beta ENABLE=1 SPEED=20 SET_POSITION=200
MOVE=78 STOP_ON_ENDSTOP=-2
  MANUAL_STEPPER STEPPER=alfa SPEED=20 MOVE=0 SYNC=0
  MANUAL_STEPPER STEPPER=beta SPEED=20 MOVE=0 SYNC=1
  MANUAL_STEPPER STEPPER=gama ENABLE=1 SPEED=20 SET_POSITION=200 MOVE=-
2 STOP_ON_ENDSTOP=-2
  MANUAL_STEPPER STEPPER=gama SPEED=20 MOVE=0
```

3.6 Převodník G-code

Kdybychom museli vypočítávat jednotlivé souřadnice sami, zabralo by to zbytečně mnoho času. Proto jsme si zvolili Orca slicer jako generátor G-codu, který naimportujeme do našeho převodníku. Ten pak dopočítá úhly pro pohyby daných krokových motorů k dosažení pozic daných G-codem (kód 8).

Kód 8: ukázka příkazu G-code pro pohyb na souřadnice [10; 50; 10]

```
G1 X10 Y50 Z10
```

Náš program načte celý řádek jako string, který následně rozdělíme do listu s dělicím znakem “mezera”. Pokud je na první pozici G1, program rozpozná, že se jedná o příkaz pro pohyb, a načte cílovou pozici v prostoru udanou G-codem.

Kód 9: příkaz importující G-code

```
with open("/home/admin/prevodnik/test.gcode") as f:
   line = f.readlines()
for i in line:
   k=i.split()
   if len(k)>=1:
      q=k.pop(0)
      if q=="G28":
         fronta.append(["http://localhost:7125/printer/gcode/script?script=home",""])
      elif q[:1]==";":
         r=1
      elif q=="MAX_DELKA":
         max_delka=float(k[0])
      elif q[:1]=="T":
         fronta.append(["http://localhost:7125/printer/gcode/script?script="+prikaz("alfa",fro
nta[-1][1][0]+5,max_speed,False ),fronta[-1][1]])
         fronta.append(["http://localhost:7125/printer/gcode/script?script=T"+q[1:],fronta[-
1][1]])
         fronta.append(["http://localhost:7125/printer/gcode/script?script="+prikaz("alfa",fro
nta[-1][1][0],max_speed,False ),fronta[-1][1]])
      else:
         for i in k:
            if i[:1] =="E":
            elif i[:1] =="X":
               x2=float(i[1:])
            elif i[:1] =="Y":
               y2=float(i[1:])
            elif i[:1] =="Z":
               z2=float(i[1:])
```

3.7 Odesílání příkazu skrze API

Příkazy vygenerované naším programem jsme nejdříve ručně kopírovali skrze webové rozhraní Klipperu do lokálního G-code na manipulátoru. Toto řešení bylo příliš komplikované, proto jsme zvolili API příkaz pro odeslání příkazu G-code do Klipperu na Raspberry Pi. Tímto způsobem odesíláme příkazy přímo z převodníku do manipulátoru.

Pro efektivnější průběh programu jsme ho rozdělili do dvou nezávislých vláken, kdy první dělá samotný přepočet a ukládá příkazy do listu a druhé dané příkazy odesílá do manipulátoru. Odeslané příkazy se odstraňují a první vlákno dopočítává následující. U většiny případů dojde k přepočtení G-codu ještě před ukončením zahomování. Rozhodli jsme se tak z důvodu snahy o zvýšení plynulosti pohybu. Kód 10 je příkazem API, kde oranžově zvýrazněná část je příkaz k vykonání.

Kód 10: příkaz pro API Klipperu

```
"http://localhost:7125/printer/gcode/script?script=home"
```

Kód 11 je podprogramem v převodníku, který se stará o odesílání příkazů.

Kód 11: podprogram odesílající příkazy

```
def odeslani():
    global fronta
    global last_comand
    fronta.pop()
    time.sleep(2)
    while fronta[0][0] !="end":
        while len(fronta)==0:
            d=1
        response = requests.get((fronta[0][0]))
        last_comand=fronta.pop(0)
```

3.8 Nahrání programu na Raspberry

Program nejprve běžel na externím počítači, který v reálném čase odesílal jednotlivé příkazy, kvůli čemuž na něm byl manipulátor závislý. Převodník se nám podařilo nahrát přímo na Raspberry Pi, ovládající manipulátor, čímž se robot stal zcela nezávislý na externím zařízení. Díky zrychlení odesílání příkazů se zvýšila maximální rychlost pohybu ramena (tab. 2).

Tabulka 2: srovnání pingu

Ping externí počítač	Ping Raspberry
Reply from 10.0.0.191: bytes=32 time=**6 ms** TTL=64	64 bytes from localhost (::1): icmp_seq=2 ttl=64 time=**0.107ms**

3.9 Proces převodu modelu na pohyb

Vše začíná výběrem vhodného 3D modelu. Ten je zpracován v slicovacím programu (např. PrusaSlicer, OrcaSlicer[2]). V případě kreslení vygenerujeme G-code první vrstvy „tisku“ a nahrajeme ho pomocí WinSCP[3] na Raspberry pi. Dále spustíme převodník, který si načte příkazy G-code a přepočítá je na pohyby jednotlivých motorů. Ty poté odesílá pomocí API do Klipperu, který pohyby vykoná skrze desku Octopus a její periferie.

[2] Konkrétně tento slicer jsme používali.
[3] Software pro vzdálenou správu souborů.

4 (B)RUM (Belt Robotic Universal Manipulator)

Jedná se o námi navržený a zkonstruovaný 4osý manipulátor. Naším cílem bylo mít všechny těžké komponenty na základně a vyhnout se přidávání váhy na jednotlivé segmenty ramena, a proto jsme se rozhodli využít přenos rotační energie pomocí řemenů. Na obrázku 11 je render 3D modelu manipulátoru.

Obrázek 11: (B)RUM

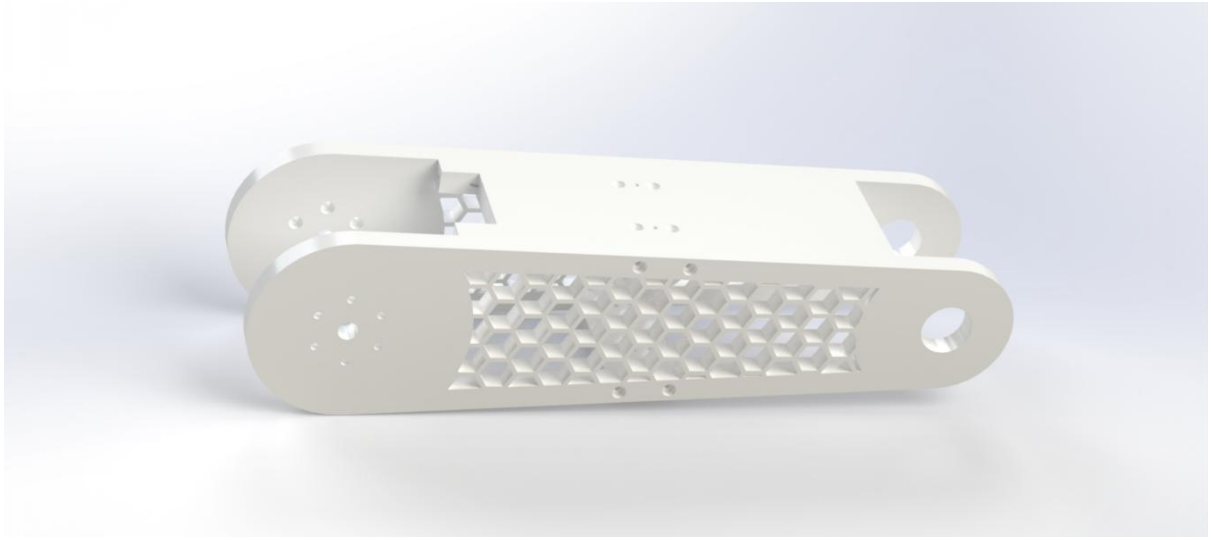
4.1 Konstrukce

Celá konstrukce byla nejdříve navržena v programu Solidworks jako sestava s možností všech pohybů, díky čemuž jsme spoustu chyb odhalili ještě před tím, než jsme začali jednotlivé díly tisknout.

4.1.1 Ramena

Rameno prvního stupně

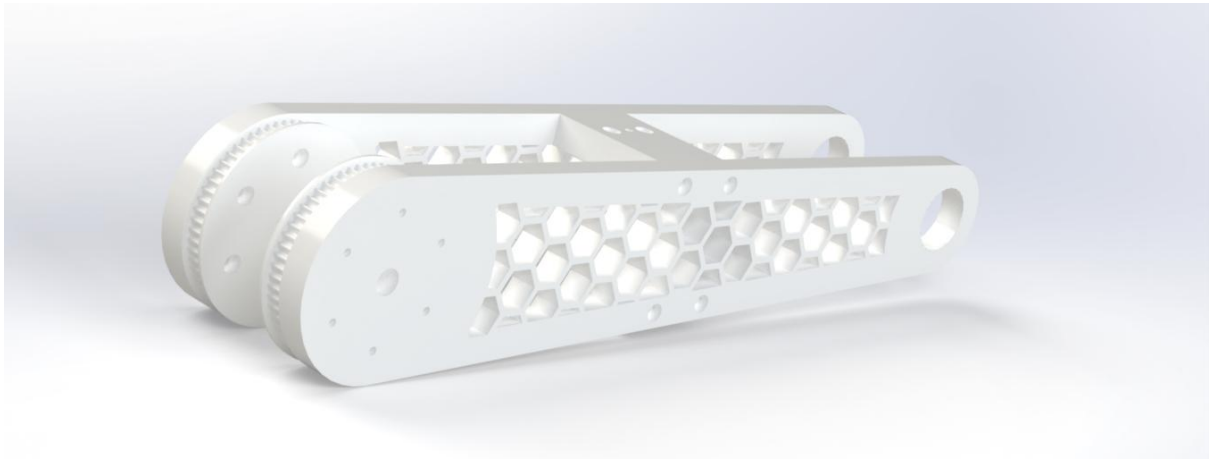
První článek se skládá ze 2 nosníků (obr. 12), odlehčených hexagonovou strukturou, propojených v horní části po celé délce a ve spodní části středovým nosníkem, na kterém je ukotvený napínací systém řemenů. Boční nosníky jsme propojili za účelem zvýšení stability ramena. Mezi bočnicemi jsou napnuté řemeny. Vzdálenost mezi body otáčení je 250 mm.



Obrázek 12: rameno prvního stupně

Rameno druhého stupně

Druhý článek je pouze zmenšené rameno prvního stupně s menší šířkou a jen s jedním středovým nosníkem (obr. 13). Vzdálenost os otáčení je též 250 mm.



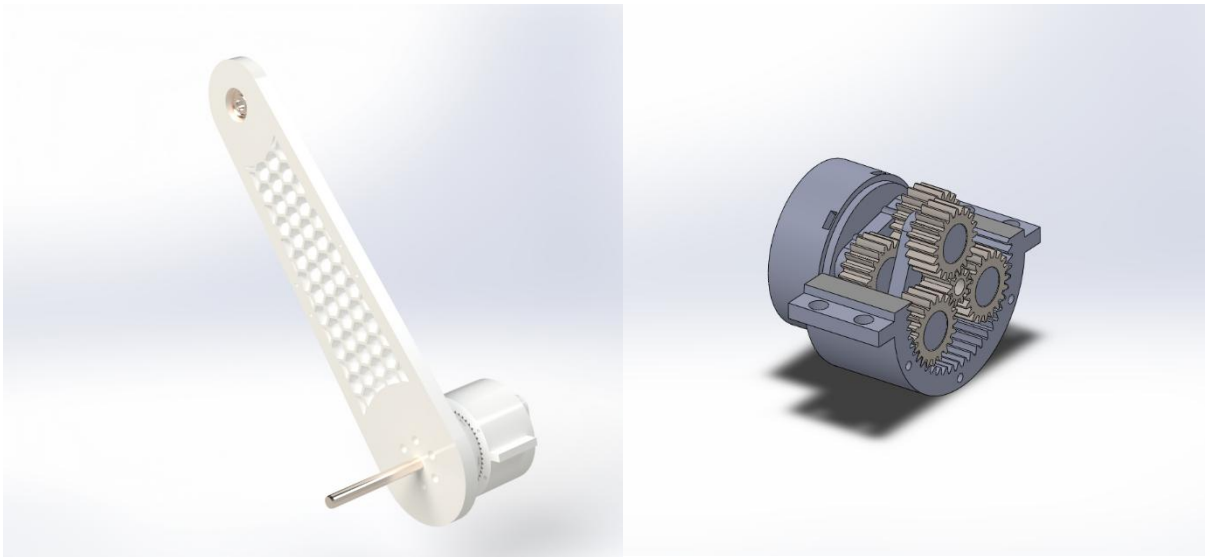
Obrázek 13: rameno druhého stupně

4.1.2 Systém náhonů

Pro osy otáčení ramen druhého a třetího stupně jsme použili náhon skrze řemeny. Osa ramena prvního stupně je řešena převodovkami v osách otáčení. Otočná podstava rotuje za pomoci ozubeného převodu.

Alfa

Náhon je řešený skrze dvě tištěné převodovky. Konkrétně se jedná o dvoustupňové planetové převodovky o převodovém poměru 1 : 36 (obr. 14). Jelikož jsou převodovky tištěné, mají vůli. Tuto vůli jsme vyřešili pootočením převodovek proti sobě.



Obrázek 14: náhon Alfa a jeho převodovka

Beta a Gama

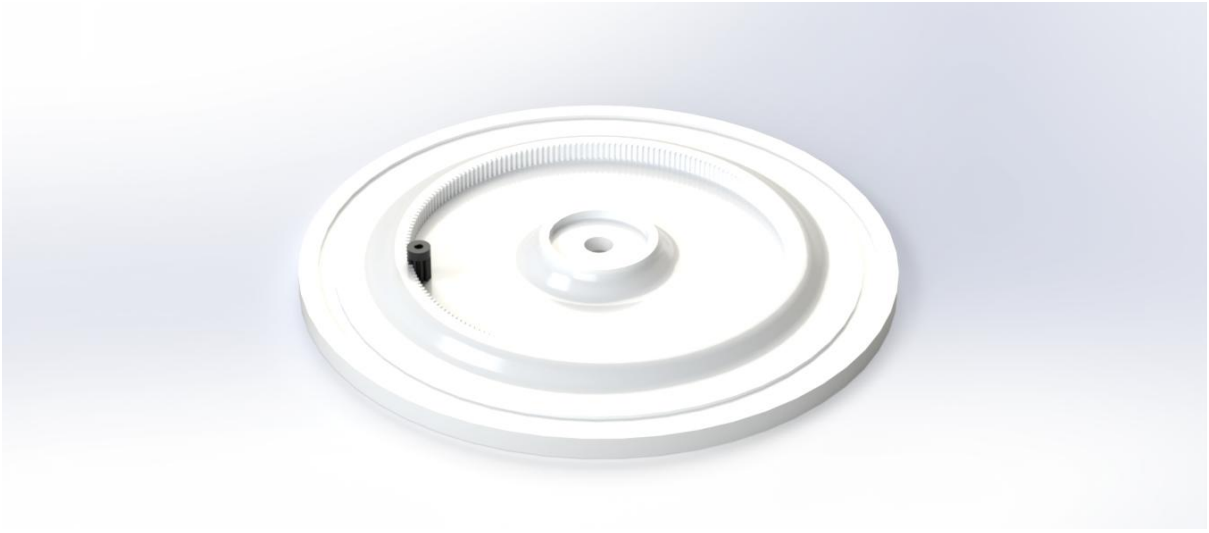
Pohyb zajišťují motory umístěné na základně, jejichž rotace je přenášena řemeny vedenými mezi bočnicemi ramena. Díky různým velikostem řemenic jsme dosáhli dostatečného převodu, nebylo tedy nutné přidávat externí převodovku. Výsledný poměr je cca 1 : 34 u Bety (obr. 15) a 1 : 44 u Gamy (obr. 16).

Obrázek 15: náhon Beta

Obrázek 16: náhon Gama

Delta

Delta je ovládána pastorkem, který hýbe vrchní částí podstavy. Na spodní části podstavy je velké vnitřní ozubené kolo, po kterém se pastorek může pohybovat (obr. 17). Plynulý pohyb zajišťují ložiska připevněná po obvodu podstavy a centrální osa otáčení.

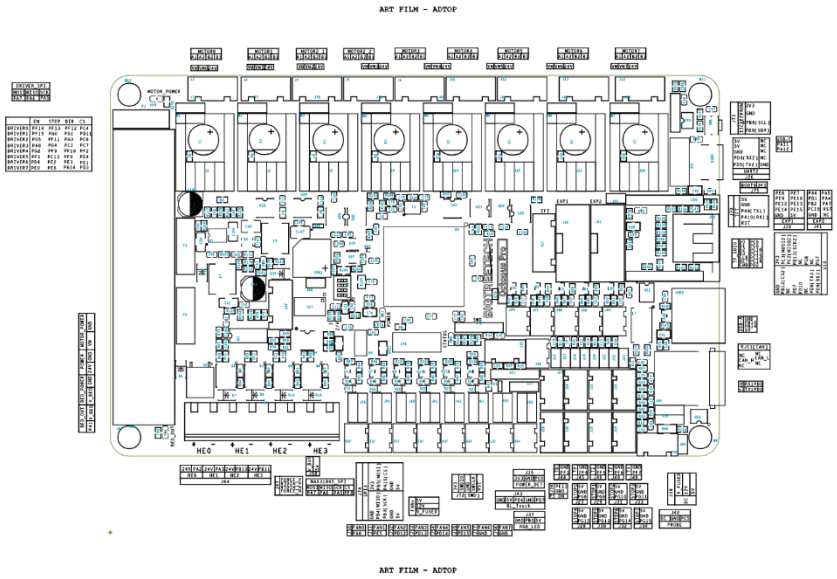


Obrázek 17: náhon Delta

4.2 Elektronika a software

Jako řídicí počítač jsme použili Raspberry Pi 3b+ z důvodu jeho velikosti, konektorové výbavy a ceny. Na tomto počítači probíhají veškeré výpočty, viz kapitola 3.8, a následně jsou posílány skrze UART sběrnici do desky s připojenými periferiemi.

Při výběru základní desky, ke které se připojují jednotlivé komponenty (motory, tlačítka, …), jsme se rozhodli pro BTT Octopus pro H723 (obr. 18), jelikož s ní máme zkušenosti a je k ní možné připojit až 8 driverů a mnoho dalších užitečných komponent. Víme, že by stačila levnější deska, ale chtěli jsme mít rezervu pro další možná rozšíření a vylepšení.



Obrázek 18: pinout desky Octopus pro H723

Motory

Použili jsme krokové motory řady NEMA17. Specifikace pro jednotlivé motory viz tab. 3.

Tabulka 3: soupis užitých motorů (B)RUM

	Velikost	Točivý moment	Převod	Proud
Alfa	2 * 60 mm	2 * 60 Ncm	36 : 1	2 * 0,9 A
Beta	1 * 60 mm	60 Ncm	240 : 7 cca 34 : 1	1 A
Gama	1 * 38 mm	42 Ncm	2160 : 49 cca 44 : 1	0.6 A
Delta	1 * 38 mm	42 Ncm	20 : 1	0.6 A

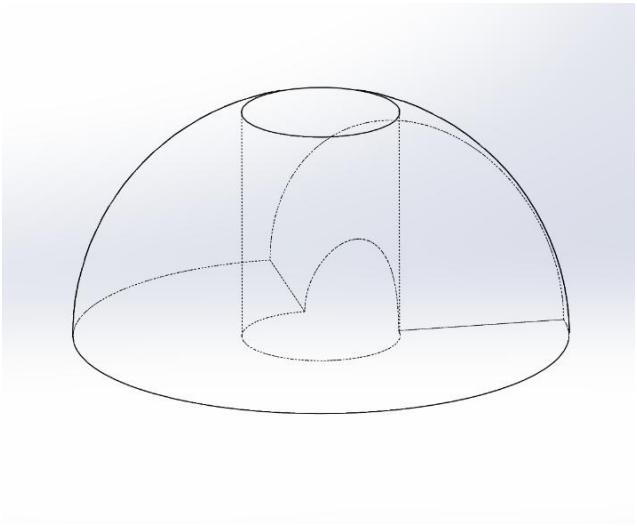
Zdroj jsme použili 24 V s 14,6 A od Meanwell kvůli jeho spolehlivosti a dostatečné výkonové rezervě.

4.3 Výsledky

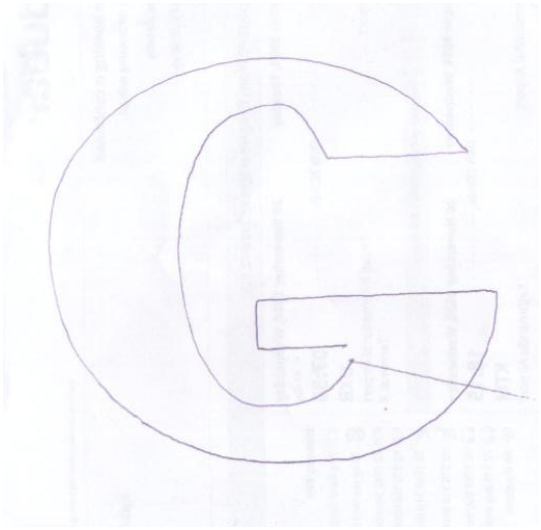
Výsledný manipulátor má oblast pohybu ve tvaru zobrazeném na obrázku číslo 20, zjednodušeně se jedná o polokouli o poloměru 500 mm.

Teoretická manipulační hmotnost může dosáhnout až 2 kg. Otestovali jsme 0,5 kg

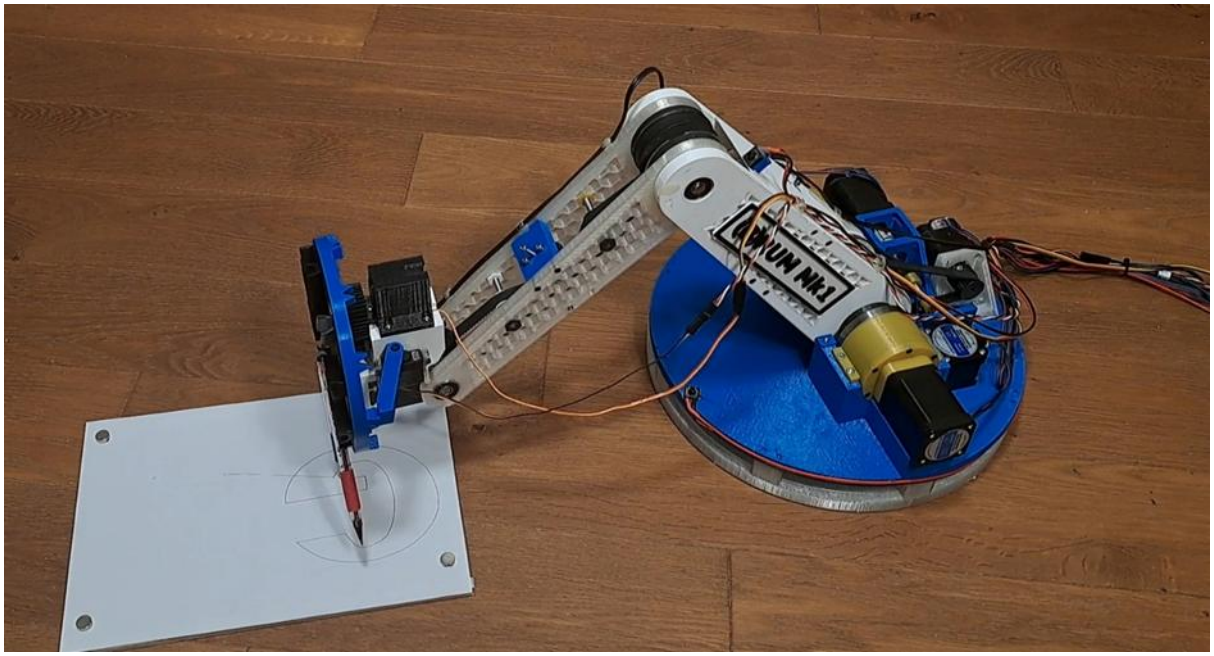
Přesnost manipulátoru se pohybuje kolem +- 1 mm, ale pohyb je opakovatelný. Tato odchylka vzniká v ose Delta a v důsledku torze v rameni Alfa. Tento výsledek nás příliš nepotěšil, proto jsme se rozhodli navrhnout a vyrobit menší a přesnější manipulátor s názvem (M)RUM.



Obrázek 20: pracovní oblast robota



Obrázek 19: logo gymnázia Vlašim



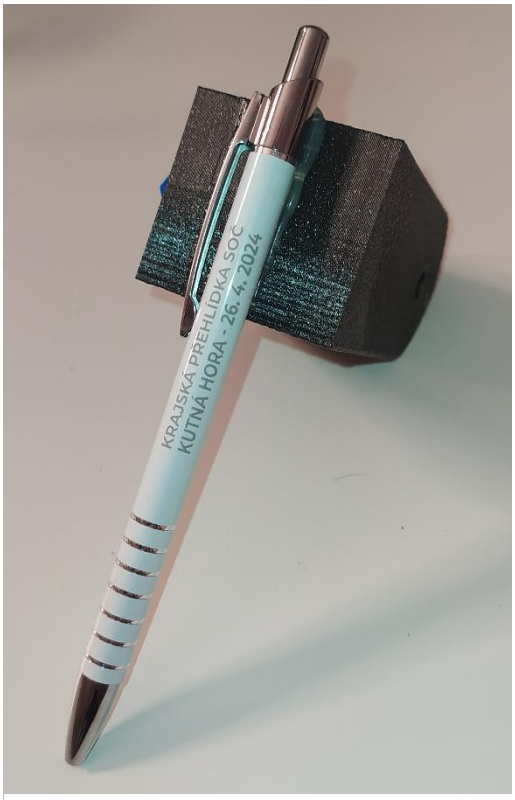
Obrázek 21: (B)RUM

5 Nástroje

V naší práci jsme se zabývali i vývojem nástrojů pro naše manipulátory. Myšlenkou bylo, že nástroje pro (M)RUM budou kompatibilní s (B)RUM, ale ne naopak z důvodů velikosti a síly manipulátorů.

5.1 První nástroj (bastl propiska)

První nástroj byl velice primitivní a spočíval pouze v přilepení propisky na konec gamy pomocí tavného lepidla. Jeho cílem bylo pouze ověřit funkčnost našeho manipulátoru, zdali se pohybuje po přímce a odpovídají-li jeho pohyby tomu, co jsme mu zadali v G-code.

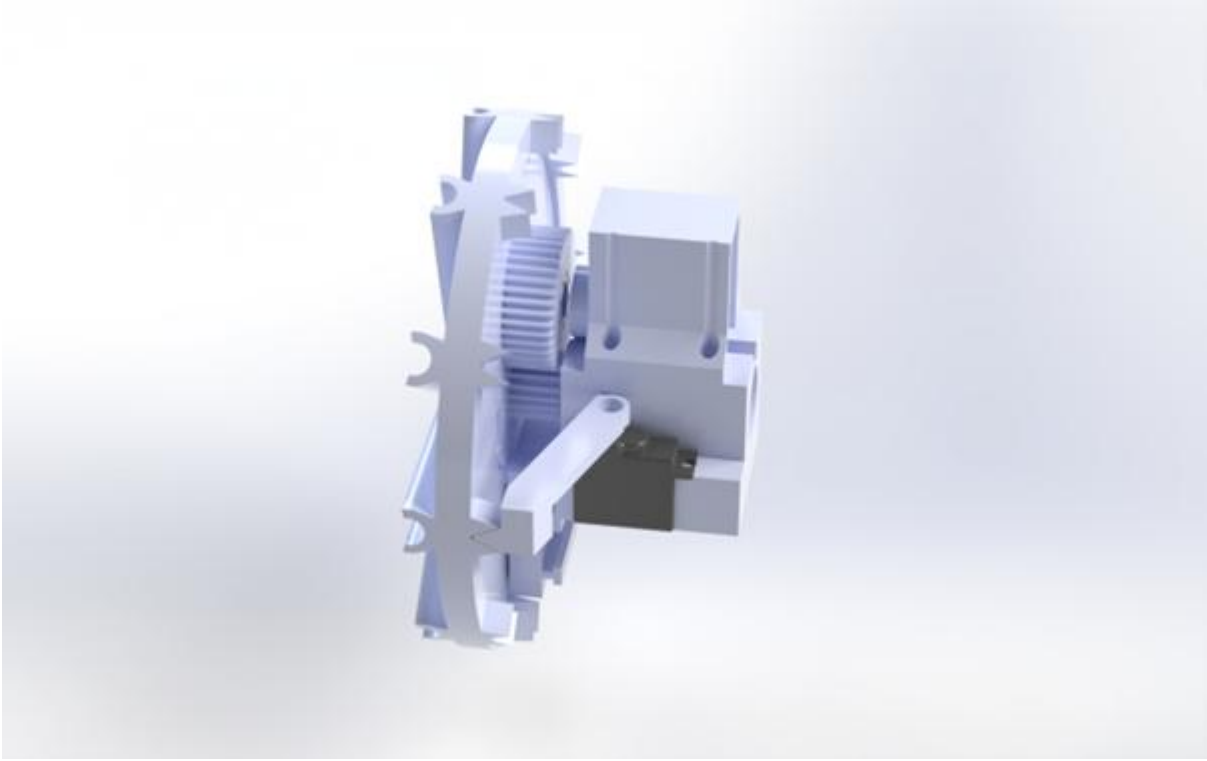


Obrázek 22: první nástroj – propiska

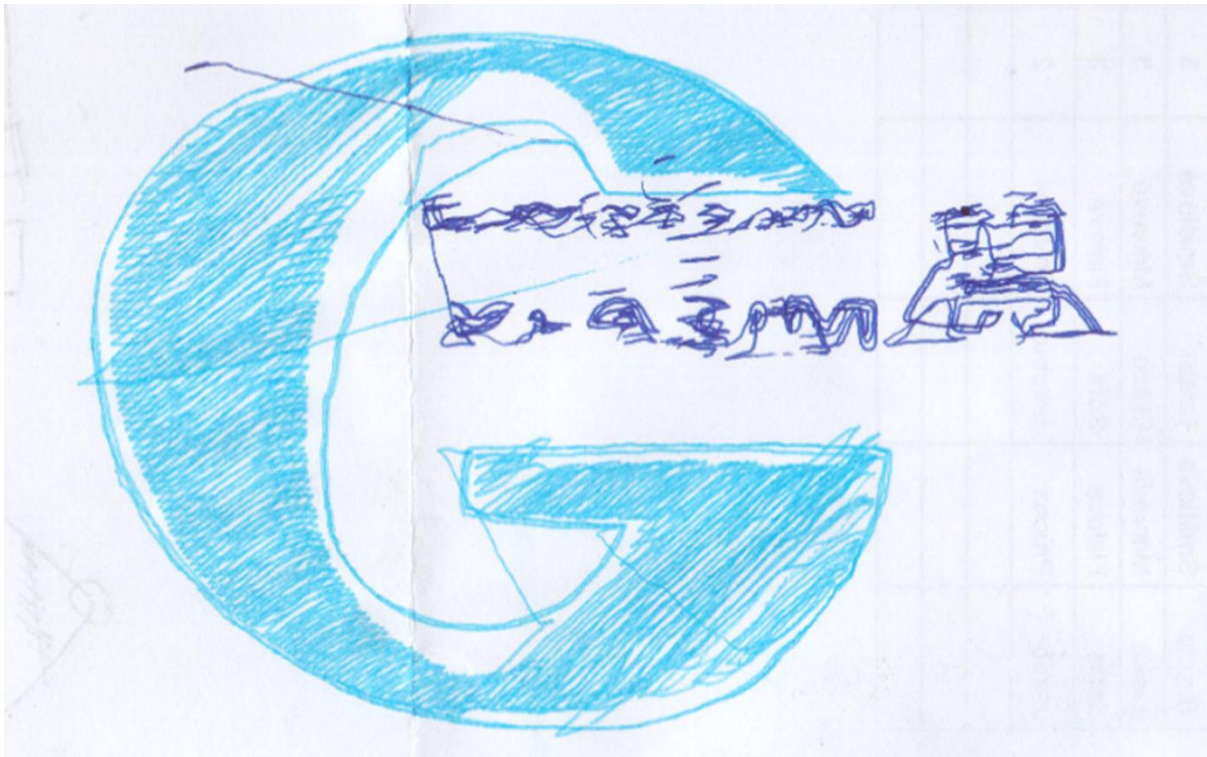
5.2 Multi propiska (slunečnice)

Tento nástroj nám měl umožnit kreslení více barvami. Skládá se z otočného disku, na který je možné přidělat až deset různobarevných propisek (obr. 23). Výměna propisky je možná díky 180° robotickému servu s přidanou převodovkou 1 : 2 pro pokrytí kompletní rotace. Ke zvětšení stability a přesnosti jsme přidali zajišťovací servo, které uzamkne rotaci disku, když má dojít ke psaní.

Rozhodli jsme se pro gelové propisky, které vyhovovaly našim požadavkům. Po několika experimentech jsme dospěli k jejich vylepšení. Přidali jsme do nich pružinu, která umožňovala propisce psát ve větším rozsahu výšky nad papírem. Díky tomu jsme dosáhli lepších a plynulejších výsledků při kreslení obrazců.



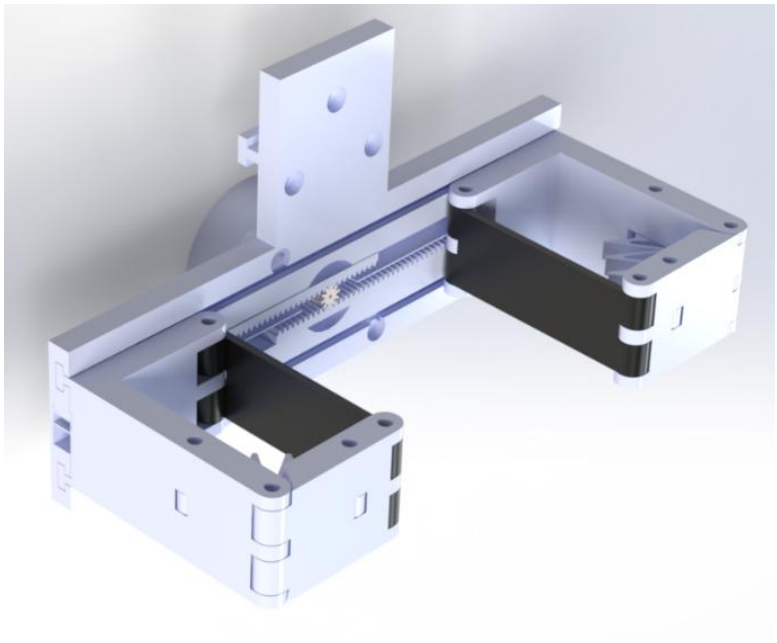
Obrázek 23: multi propiska



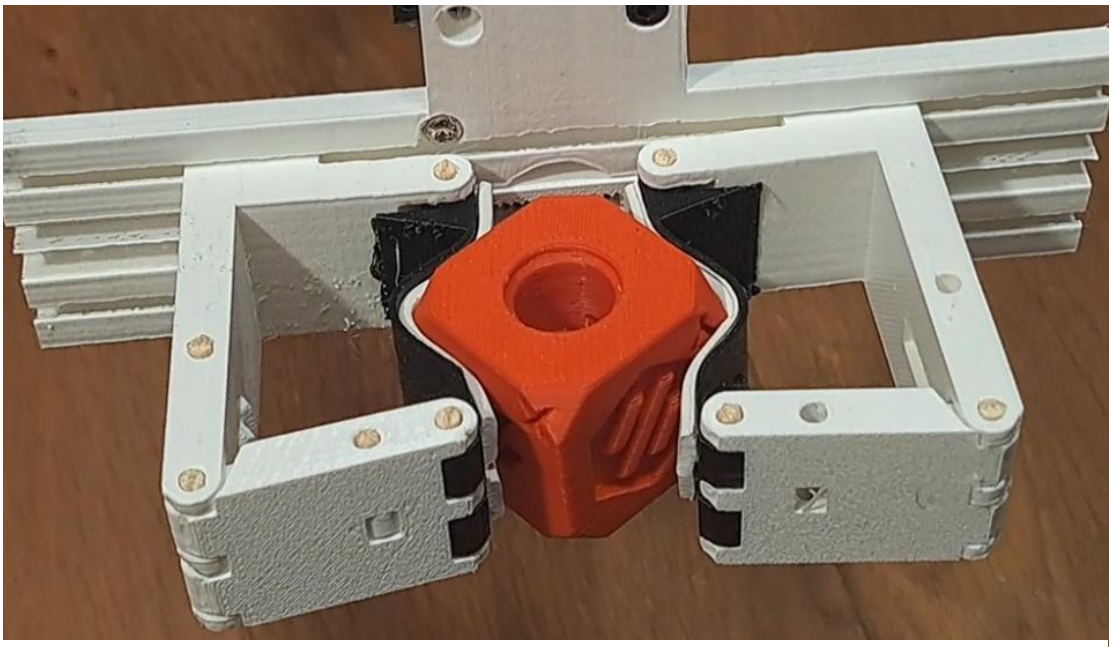
Obrázek 24: barevné logo gymnázia Vlašim

5.3 Kleště (klepítka)

Chtěli jsme vytvořit co nejuniverzálnější kleště, proto jsme navrhli kleště s prohnutelnou kontaktní plochou, tištěnou z flexibilního materiálu (obr. 25, 26). Jelikož tištěná membrána neměla dostatečné tření, nalepili jsme na ni těsnění, které se nám osvědčilo z podstavy, kam jsme ho umístili z důvodu zabránění skluzu po podložce. V softwaru si můžeme změnit tlak stisku pomocí odlišných proudů v motoru, který skrze pastorek a hřebenovou tyč pohybuje s čelistmi.



Obrázek 25: redner kleště



Obrázek 26: kleště s přizpůsobenou kontaktní plochou

6 Aktuální činnost

Bohužel jsme do data odevzdání nestihli zcela dodělat menší manipulátor (M)RUM a grafické rozhraní pro ovládání manipulátorů. V dalších kapitolách popisujeme aktuální stav vývoje.

6.1 (M)RUM

Jedná se o menší 4osý manipulátor, u něhož jsme se chtěli vyvarovat nepřesností, které vznikly na (B)RUM. Na obrázku 27 je render 3D modelu.

Obrázek 27: (M)RUM

6.1.1 Konstrukce

Změnili jsme konstrukci ramen, a to z původně dvou rovnoběžných nosníků na pravidelný dutý šestiboký hranol, díky čemuž se zvýšila odolnost v torzi a celková tuhost manipulátoru.

6.1.1.1 Ramena

Rameno prvního stupně

První článek na sobě nese 2 krokové motory, které ovládají úhly Beta a Gama. Tyto motory jsou umístěné za bodem otáčení Alfa, čímž působí jako protiváha ramena.

Rameno druhého stupně

Druhý článek má rovněž tvar šestibokého hranolu. Na jeho konci je umístěn článek třetího stupně s nástrojem.

6.1.1.2 Systém náhonů

Alfa

Zde došlo ke změně a nyní se o rotaci stupně Alfa stará pouze řemenový převod, a to v poměru 1 : 5.

Beta a Gama

O převod se již nestará pouze řemen, ale byla přidána i jednostupňová planetová převodovka v poměru 1 : 5. Samotný převod na řemenu je 1 : 4, čímž je celkový převodový poměr 1 : 20. Toto řešení je pro Betu a Gamu shodné.

Delta

O rotaci Delta se již nestará vnitřní ozubené kolo s pastorkem. Nahradili jsme ho řemenovým převodem v poměru 20 : 1. Předpokládáme, že dojde ke zmenšení vůle oproti původní konstrukci.

6.1.2 Elektronika a software

Oproti (B)RUM jsme využili silnější Raspberry Pi desku, protože jsme chtěli přidat grafické rozhraní programu a stávající deska by tomuto účelu již nestačila. Dále jsme použili menší motory (tab. 4). Zbytek elektroniky zůstává shodný.

Tabulka 4: soupis užitých motorů (M)RUM

	Velikost	Točivý moment	Převod	Proud
Alfa	2 * 48f mm	52 Ncm	5 : 1	0,8 A
Beta	1 * 38 mm	42 Ncm	20 : 1	0,8 A
Gama	1 * 38 mm	42 Ncm	20 : 1	0,6 A
Delta	1 * 38 mm	42 Ncm	20 : 1	0,6 A

6.1.3 Změny vůči (B)RUM

Největší změnou je velikost, kdy byl 1. článek zmenšen z 250 mm na 150 mm a 2. rameno z 250 mm na 200 mm. Další změna je ve tvaru jednotlivých článků. Přešli jsme ze dvou rovnoběžných nosníků na pravidelný dutý 6boký hranol. Tento přechod rapidně zvýšil odolnost v torzi. Stále není žádný motor zavěšen před bodem otáčení Alfa. Oproti (B)RUM, kde byly motory umístěny na podstavě robota, jsou nyní přímo na prvním článku. Dále elektronika je již na samotné podstavě robotu, a ne vedle něj.

7 Vylepšení po krajském kole

Po odevzdání práce jsme nezaháleli a pokračovali jsme ve vývoj, který popisujeme níže.

7.1 Grafické rozhraní

Pro využití ve výuce bylo zcela zásadní vytvořit přehledné prostředí, ve kterém by manipulátor mohli ovládat i uživatelé, kteří se v programování příliš nevyznají. Proto jsme se rozhodli naprogramovat GUI, které by usnadnilo ovládání manipulátoru, požadovali jsme od něj tyto funkce:

- **nastavení cílové pozice robota pomocí zadání souřadnic X, Y, Z nebo konkrétních úhlů**
- **vizualizace úhlů díky zjednodušeným modelům ramene**
- **import G-code, který se následně přepočte na pohyby jednotlivých motorů a začne se na robotu provádět**
- **přímé ovládání robota klávesnicí (ovladačem)**
- **ovládání nástrojů**
- **macra – předem naprogramované sekvence pohybů (homování, transportní pozice)**

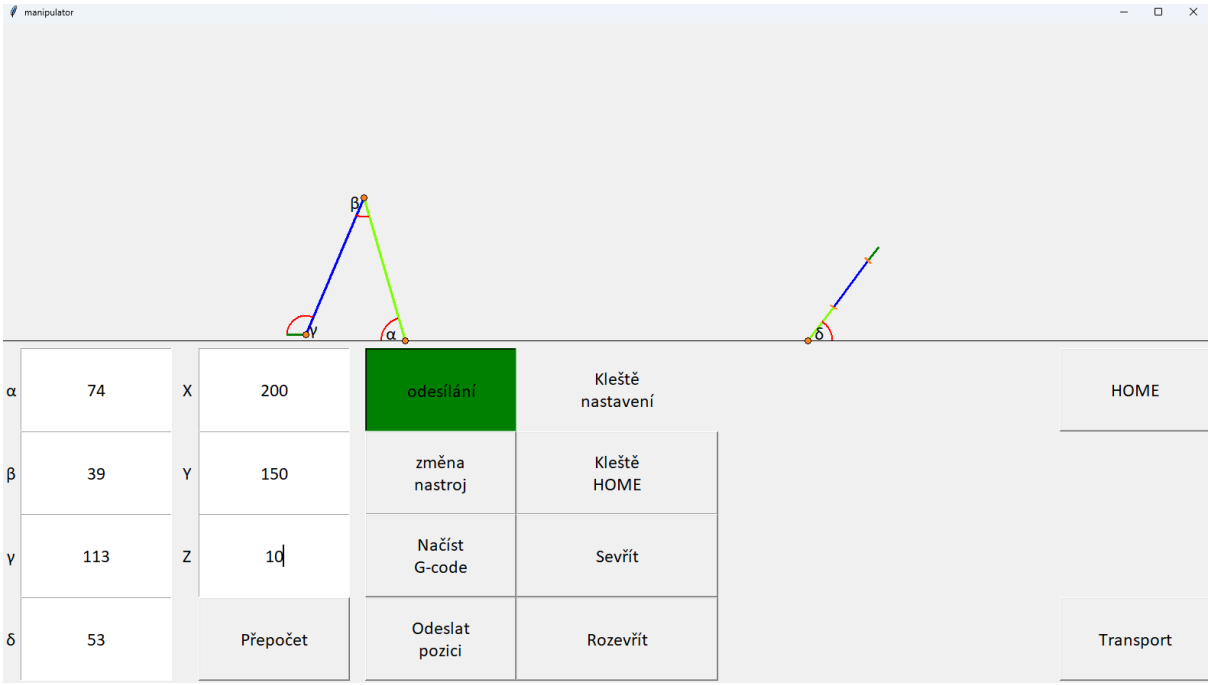
7.1.1 Provedení

Toto rozhraní je programované v pythonu pomocí knihovny Tkinter. Celé rozhraní komunikuje s novou verzí převodníku. Komunikace mezi programy probíhá skrze 2 soubory, do prvního se ukládají příkazy a do druhého status ramena. Díky tomuto řešení je možné naprogramovat rozhraní v jakémkoliv programovacím jazyce, stačí se řídit předdefinovanými příkazy. Také je takto jednoduché udělat různá oprávnění uživatelů. Pro realizaci jsme použili Raspberry OS[4] s grafickým rozhraním. Tento systém vyžaduje vyšší výkon, proto jsme se rozhodli použít výkonnější Raspberry konkrétně Pi 5 s 8 GB RAM. K tomuto počítači jsme připojili dotykový 7palcový display.

7.1.2 Schopnosti

Naše rozhraní vykresluje aktuální pozici ramena i jeho pohyb. Dále je možné skrze rozhraní odeslat požadovanou pozici ramena, či načíst celý G-code, který rameno vykoná. Při odesílání G-code je možné zapnout smyčku, která spustí program několikrát po sobě, nebo nastaví jeho nekonečné vykonávání, dokud ho uživatel nevypne. Výsledné rozhraní můžete vidět na obrázku 28.

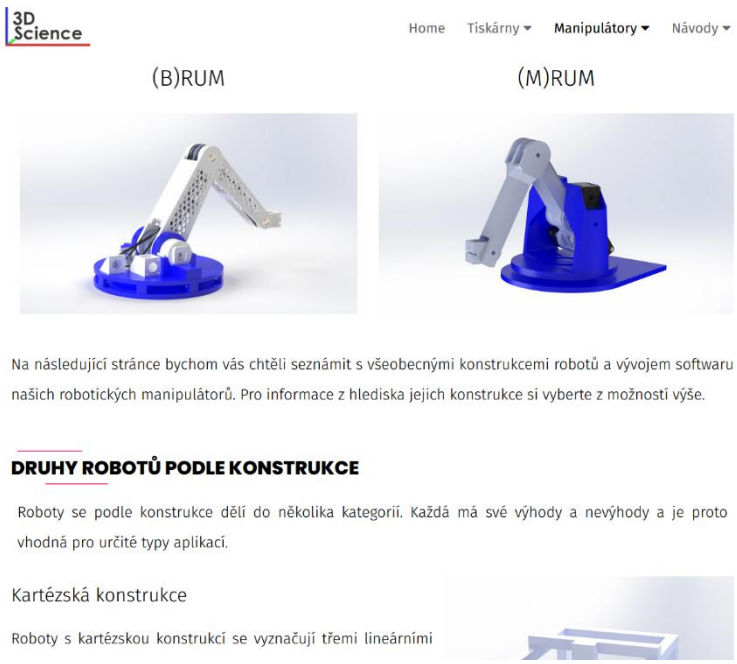
[4] Raspberry OS je fork linuxu Debian



Obrázek 28: grafické rozhraní

7.2 Webové stránky

Publikovali jsme stránky, na kterých popisujeme princip a konstrukci našeho manipulátoru a naše další projekty. Na stánkách boudou kompletní návody na instalaci softwaru pro ovládání manipulátoru a návod na tisk a sestavení manipulátoru (M)RUM a jeho nástrojů. Odkaz viz níže. (obr. 29)
www.3dscience.cz



Obrázek 29: snímek z naší webové stránky

8 Závěr

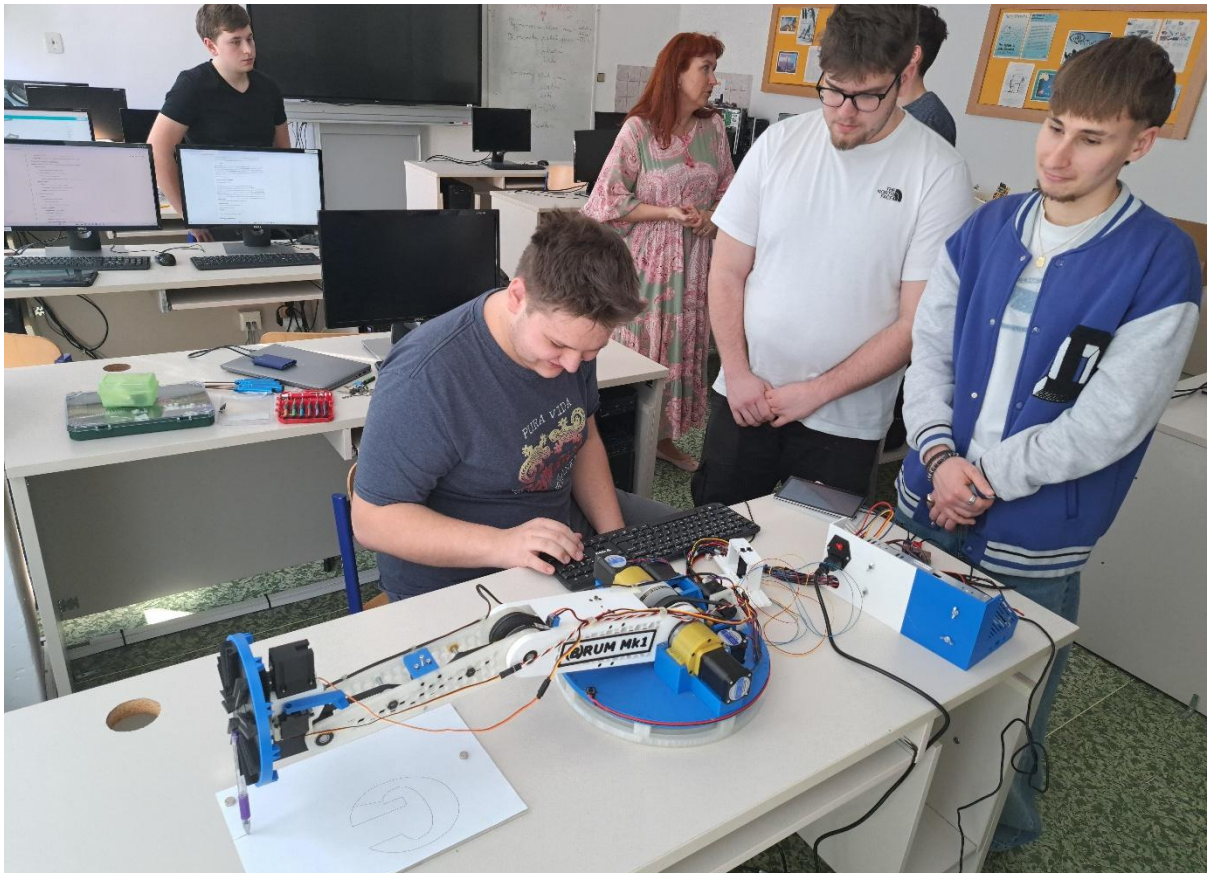
Podařilo se nám sestrojit funkční manipulátor schopný pohybu po přímce. Vytvořili jsme si vlastní převodník, který dokáže přečíst G-code, převést ho na pohyby jednotlivých motorů a odeslat do manipulátoru. Dále jsme naprogramovali přehledné grafické rozhraní. Navrhli a zhotovili jsme vyměnitelný nástroj pro kreslení 10 různými barvami, které se dají v průběhu kreslení měnit, a kleště pro přesun objektů.

Naše poznatky, závěry a vzniklý software jsme zveřejnili na našem webu, kde návštěvníkům nabídneme i kompletní přístup k 3D modelům, potřebným pro sestrojení manipulátoru (M)RUM a jeho nástrojů. Dále zde zveřejníme i postup, jak manipulátoru vyrobit.

Postupný vývoj převodníku nám zabral poměrně mnoho času, ale získali jsme tak velké množství cenných zkušeností. Jsme si vědomi toho, že použití již existujícího softwaru by bylo pravděpodobně jednodušší, ale jsme rádi, že jsme si na kód mohli přijít sami.

Zároveň jsme přesvědčeni, že jsme takto mohli lépe pochopit vztahy mezi goniometrickými funkcemi a pohybem robota a prozkoumali problematiku více do hloubky. Myslíme si, že jsme schopni získané vědomosti předat zájemcům.

Manipulátor jsme donesli na hodinu semináře z informatiky pro 3. a 4. ročníky gymnázia, vysvětlili jsme spolužákům jeho fungování a nechali jsme je vyzkoušet si manipulátor na vlastní kůži (obr. 30).



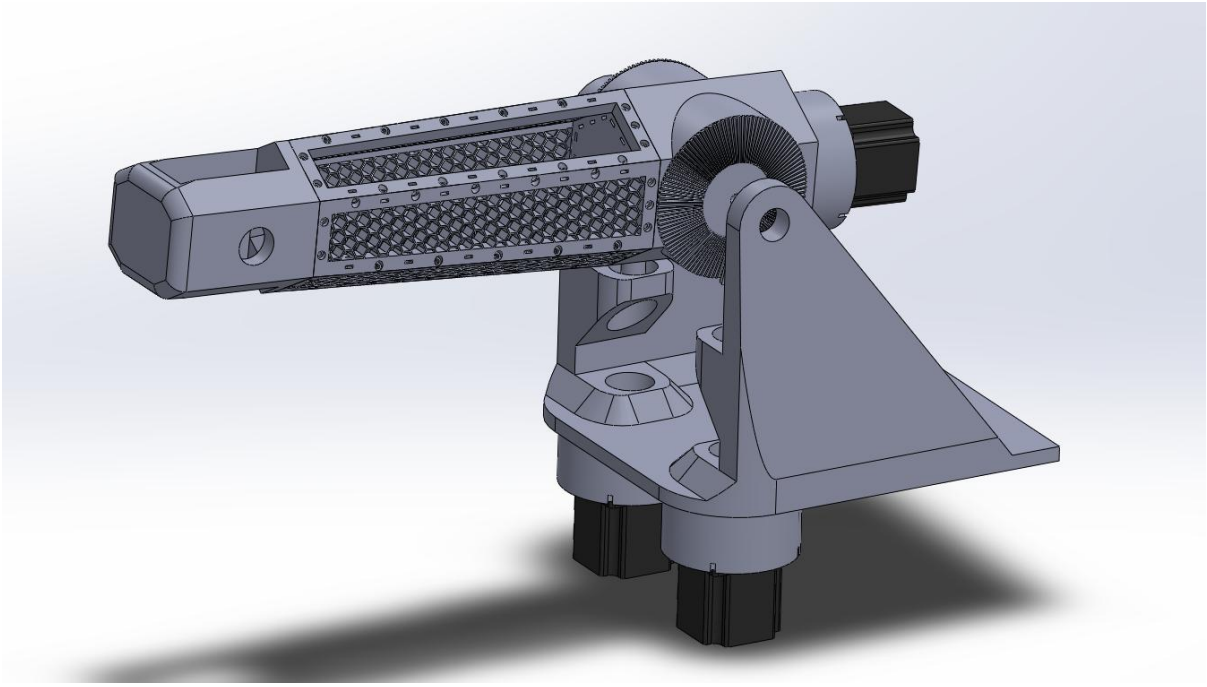
Obrázek 30: manipulátor na semináři informatiky

Zpočátku jsme u manipulátoru (B)RUM nedosahovali nejvyšší přesnosti, ta se nám však podařila několika úpravami znatelně navýšit. Ale i přes její navýšení jsme navrhli další model (M)RUM, kde jsme se snažili některým vzniklým chybám předejít a docílit tak ještě vyšší přesnosti. V době psaní práce ještě nevíme, jaké přesnosti (M)RUM dosáhne, ale jsme přesvědčeni, že bude vyšší.

V rámci oslovování sponzorů jsme vytvořili krátké video o fungování robota, které přikládáme na tomto odkaze: https://www.youtube.com/watch?v=OPUwMpHik0M

8.1 Možnosti dalšího vývoje

Během vývoje robotů jsme se setkali s mnoha oblastmi, o které bychom mohli naši práci obohatit a do budoucna bychom jistě některé z těchto projektů chtěli uskutečnit. Jedním z nejvíce promyšlených je (C)RUM – Cardan Robotic Universal Manipulator (obr. 31). Jistou dobu jsme se zabývali myšlenkou, zda neudělat rameno s ještě větším dosahem poháněné kardany. Nakonec však k realizaci i vzhledem k finanční náročnosti nedošlo.



Obrázek 31: (C)RUM – podstava a první rameno

Uvažovali jsme i nad ještě menší verzí manipulátoru, která by byla levnější a snazší na sestavení. Kdybychom u manipulátorů dosáhli vyšší přesnosti, rádi bychom jako jeden z nástrojů přidali i tiskovou hlavu.

Dalším nápadem bylo vytvořit ovladač na robota nebo více kreativní zmenšený model robota, se kterým by se dalo ručně pohybovat. Na bodech otáčení by byly umístěny potenciometry. Pomocí nich by bylo možné zaznamenávat pohyby miniatury a rovnou je provádět na manipulátoru.

Zajímavým projektem by bylo použití programu Scratch pro ovládání manipulátoru. Myslíme si, že by to mohlo motivovat i mladší studenty pro práci s robotem. Věříme, že na náš projekt naváží mladší studenti z gymnázia a možnosti manipulátoru o něco rozšíří.

Přemýšleli jsme i nad možným využitím našeho manipulátoru. Od počátku jsme ho koncipovali jako studijní pomůcku. Jsme si tedy vědomi toho, že nedosahuje kvalit a spolehlivosti potřebné pro průmyslové využití. V rámci možných projektů nás napadlo například třídění kostek podle barvy. Na kleště by se přidal senzor barvy. V předem daných sloupcích by byly vyskládány kostky náhodné barvy. Cílem robota by bylo kostky přeskládat do sloupců podle barev. Další zajímavou výzvou by bylo vytvořit program, který by stavěl domino. Možností je však mnoho. Ne nadarmo jsme náš manipulátor pojmenovali univerzální.

9 Bibliografie

Klipper developers. Klipper documentation. Online. 2016. Dostupné z: https://www.klipper3d.org/. [cit. 2025-03-16].

Redakce časopisu Elektroprůmysl.cz. Od mechanických ramen k autonomním systémům: Historie a vývoj průmyslových robotů. Online. 2024. Dostupné z: https://www.elektroprumysl.cz/automatizace/od-mechanickych-ramen-k-autonomnim-systemum-historie-a-vyvoj-prumyslovych-robotu. [cit. 2025-03-16].

W3Schools. Python Requests get() Method. Online. Dostupné z: https://www.w3schools.com/python/ref_requests_get.asp. [cit. 2025-03-16].

GeeksforGeeks. Multithreading in Python. Online. 2025. Dostupné z: https://www.geeksforgeeks.org/multithreading-python-set-1/. [cit. 2025-03-16].

10 Seznam obrázků a tabulek