

# A Používateľská príručka

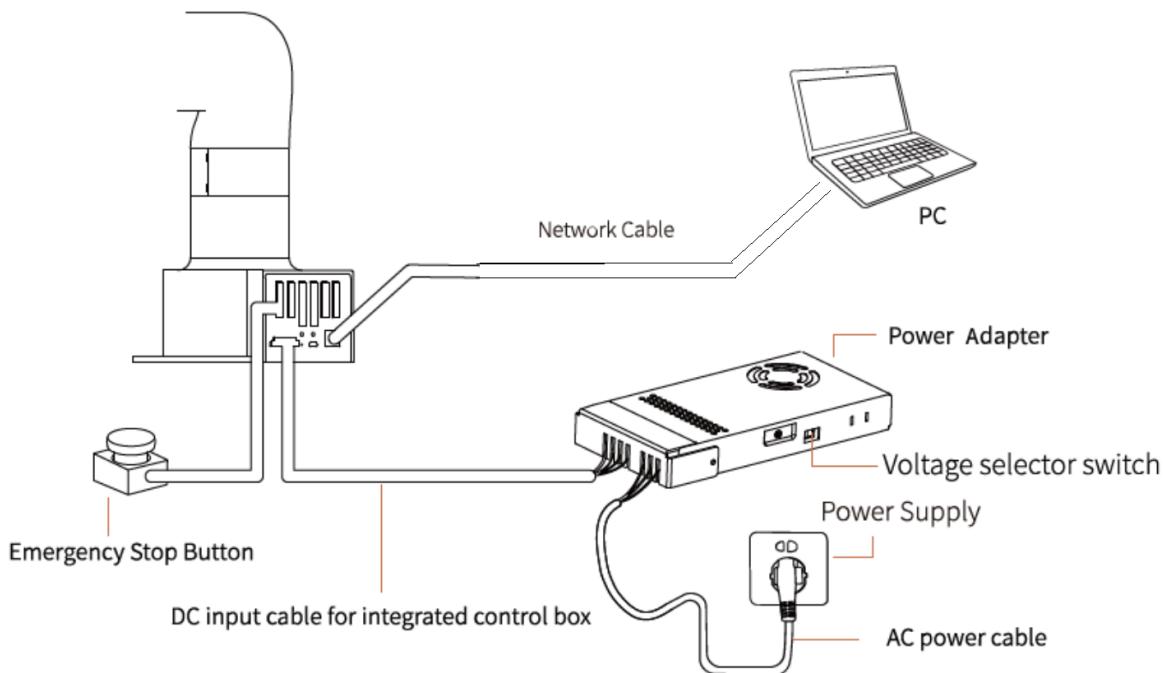
Tento dokument obsahuje všetky potrebné kroky pre úspešné spustenie procesu automatizovanej 3D rekonštrukcie objektu, zahŕňajúc softvérové a hardvérové zapojenie.

## A.1 Zapojenie hardvérovej časti

Prvým krokom k úspešnému spusteniu automatizovanej 3D rekonštrukcie objektu je zapojenie hardvérových komponentov, ktoré sa skladajú z robotického ramena Ufactory Lite 6, rotačnej podložky, používateľského počítača, kamery a svetla.

### A.1.1 Ufactory Lite 6

Pre zapojenie ramena je potrebné skontrolovať pracovný priestor tak, aby v pracovnej ploche neboli žiadne prekážky. Rameno sme upevnili k základni pomocou montážnych skrutiek a príslušenstva. Následne sme pripojili vstupný DC kábel k integrovanej ovládacej skrinke robotického ramena. Taktiež sme pripojili núdzový vypínač pre bezpečné odpojenie energie od robota pri možnej kolízii. Pokračuje sa zapojením kábla LAN, čím sa prepojí komunikáciu medzi PC a samotným ramenom.

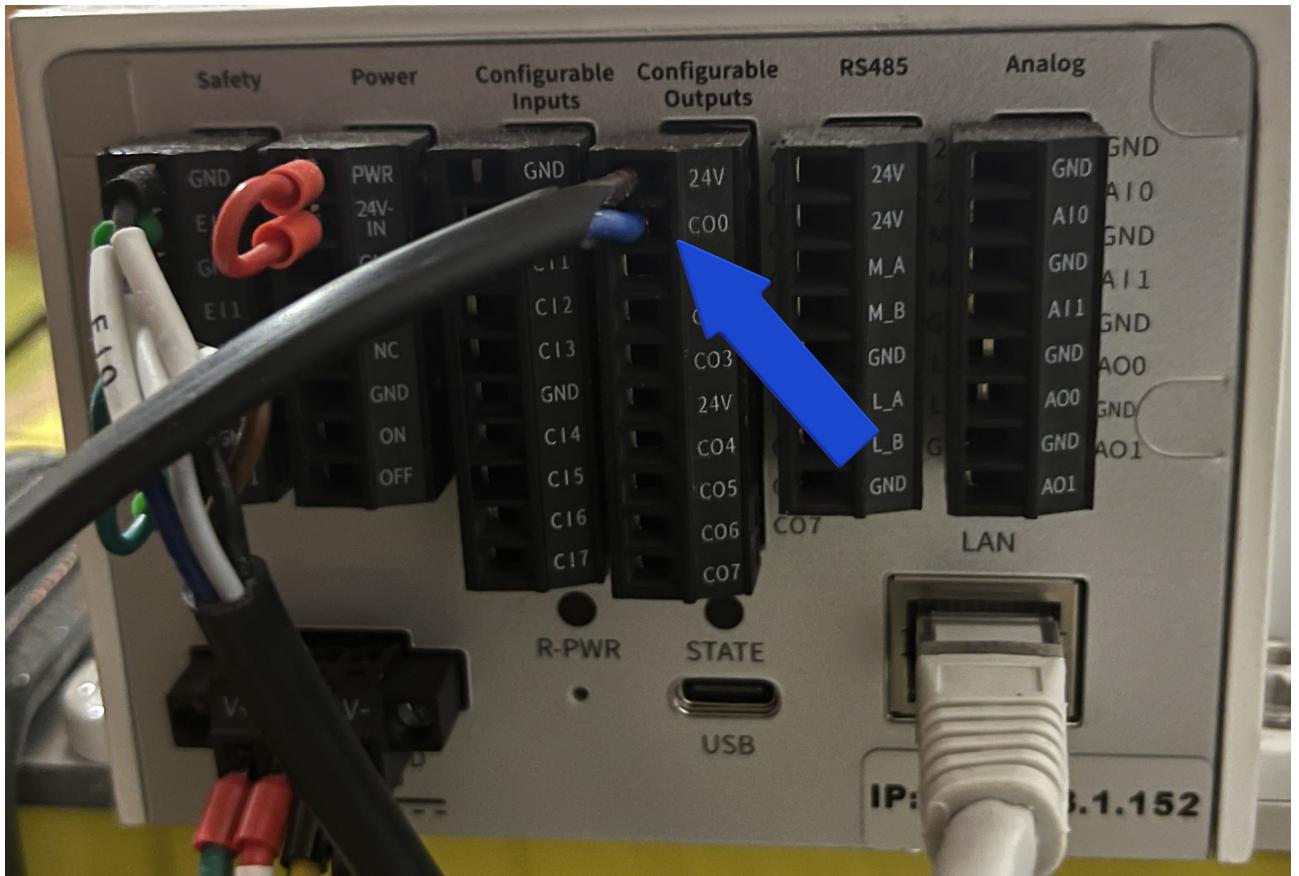


Obr. A.1: Schéma zapojenia kolaboratívneho robota[29]

Pokiaľ je všetko pripojené tak ako na obrázku vyššie, nasleduje zapojenie rotačnej podložky.

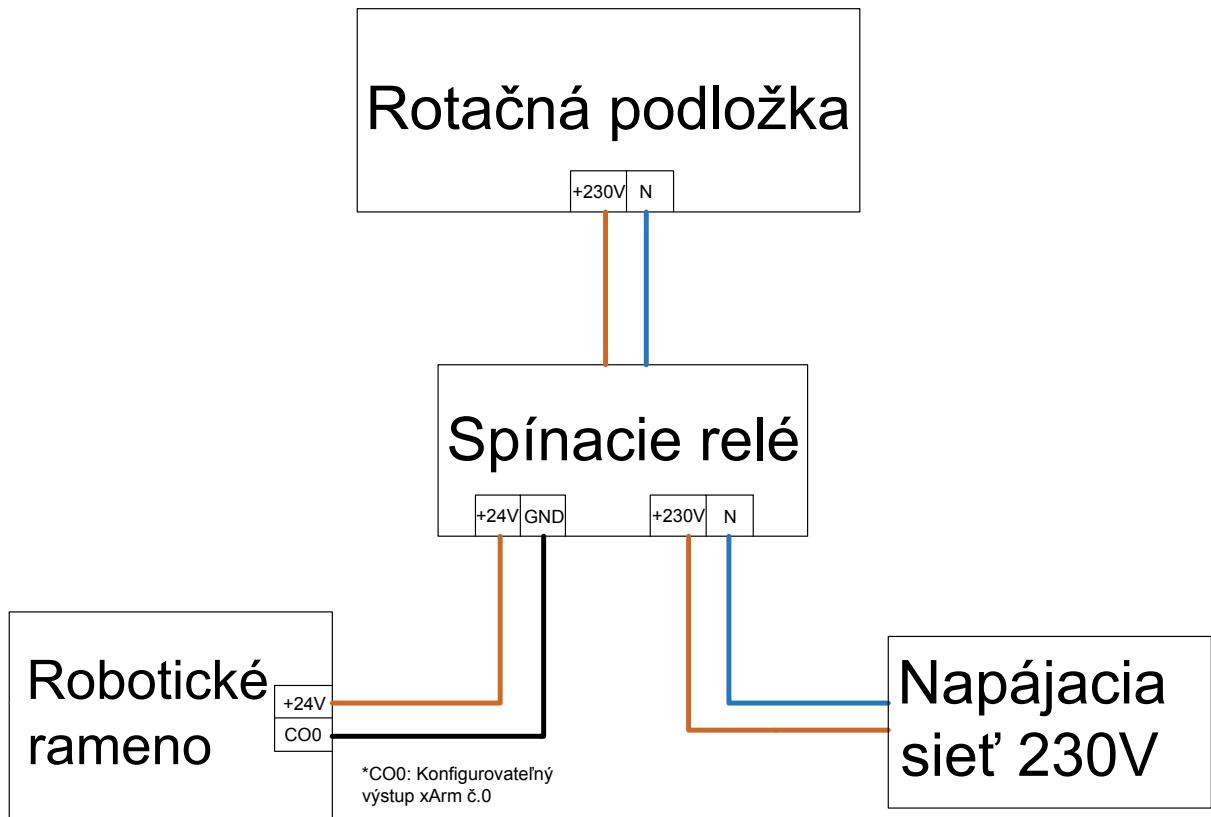
### A.1.2 Rotačná podložka

Táto podložka sa aktivuje cez konfigurovateľný výstupový signál, ktorý je možné ovládať výstupom z kobota. Hnedý kábel treba priradiť k 24V a modrý k CO0 v sekcií **Configurable Outputs** tak ako na obrázku:



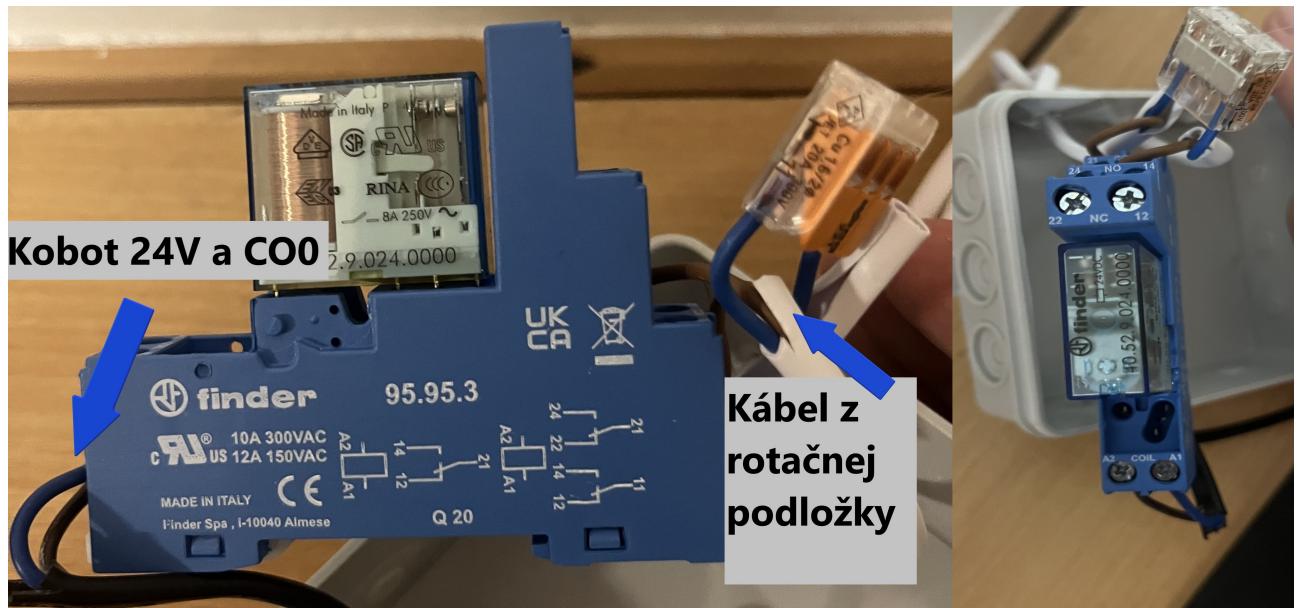
Obr. A.2: Pripojenie 24V a CO0 z kobota

Na spínacie relé treba napojiť 24 voltové napätie a taktiež pripojiť výstup CO0 z kobota, čo predstavuje konfigurovateľný výstup s číslom 0, ktorý pri aktivovaní zopne uzemnenie, ktoré aktivuje relé obvod a zapríčiní spojenie 230 voltového kábla pre napájanie rotačnej podložky.



Obr. A.3: Schéma zapojenia rotačnej podložky

24V a CO0 (spínacie uzemnenie) pri spustení aktivuje relé obvod, ktorý zopne pred tým rozpojené napájanie rotačnej podložky. Jeden koniec kábla je zapojený v sieti na 230V a druhý koniec je v rotačnej podložke.



Obr. A.4: Relé pre spúšťanie rotačnej podložky

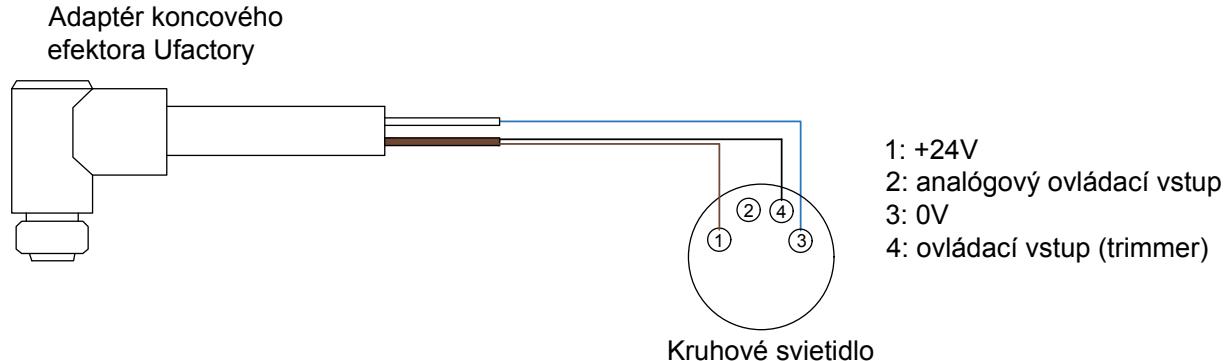
### A.1.3 Kamera a svetlo

Svetlo a kamera určené pre túto prácu sú spojené Merkur stavebnicou, preto ich budeme zapájať v jednom kroku. Najskôr tento spojený komponent priskrutkujeme o koniec ramena, čo predstavuje miesto pre koncový efektor.



Obr. A.5: Uchytenie kamery a svetla o kobota

Kábel adaptéra koncového efektora je spojený s káblom, pričom treba použiť kábel hnedej farby, ktorý slúži na prenos 24 voltového napájania, a biely kábel, ktorý je určený na uzemnenie, čo zabezpečuje aktiváciu osvetlenia. Kábel z kruhového svetla obsahuje 4 káble, ktoré treba spojiť s káblom adaptéra koncového efektora nasledovne:



Obr. A.6: Schéma zapojenia kruhového svetla

Následne treba zapojiť jeden koniec do svetla a druhý do kobota:



Obr. A.7: Spojenie svetla a kobota

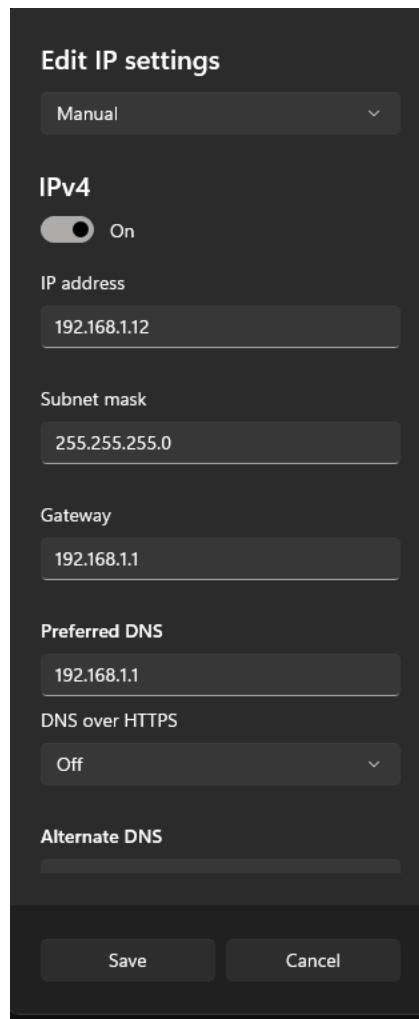
Pre hardvérové zapojenie kamery je potrebné zapojiť káblové USB rozhrania do počítača, ku ktorému je pripojené taktiež robotické rameno.

## A.2 Prepojenie softvérovej časti

Kobot poskytuje štandardné gigabitové rozhranie Ethernet TCP/IP, ktoré je prepojené sietovým káblom. Jedno koncové rozhranie je integrované v robotickom ramene a druhé je pripojené k počítaču, z ktorého chceme kobot ovládať. Po úspešnom pripojení indikátor sietového portu bliká, signalizujúc spojenie.

### A.2.1 Nastavenie IP adresy

IP adresa musí byť v rovnakom LAN prostredí ako naše robotické rameno, konkrétnie nastavená na 192.168.1.x, pričom "x" predstavuje číslo, medzi 1 a 255. Toto koncové číslo sa musí lísiť oproti číslu, ktoré má robotické rameno. Presná IP adresa kobota je zobrazená na informačnom štítku. Model Ufactory Lite 6, pridelený pre túto diplomovú prácu, má presnú IP adresu **192.168.1.152**.



Obr. A.8: Nastavenie IP adresy

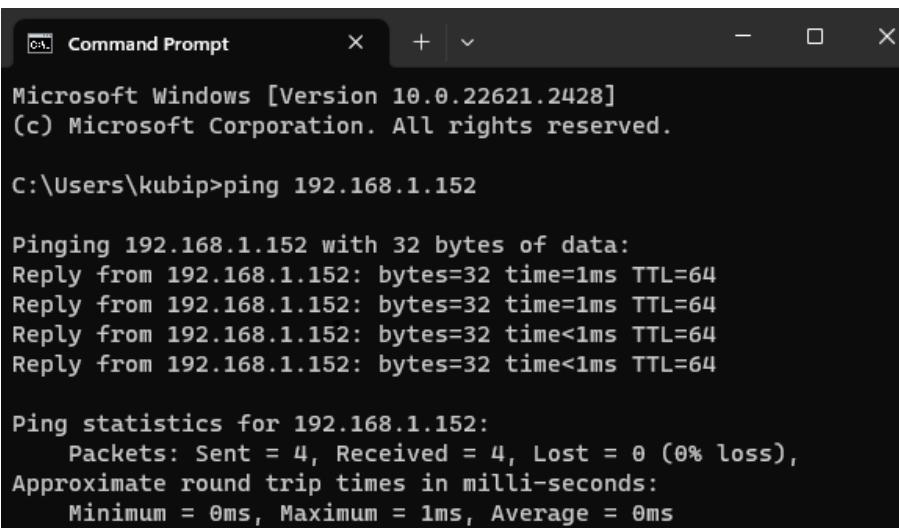
Táto konfigurácia zabezpečuje, že kobot a počítač sú súčasťou rovnakého lokálneho sietového prostredia.

## Overenie spojenia medzi kobotom a počítačom

Po správnom hardvérovom pripojení a nastavení IP adresy je možné overiť IP adresu počítača cez príkazový riadok. Používateľ zadá príkaz ***ping*** na adresu IP robotického ramena a pri správnej konfigurácii mu prídu informácie o stave pripojenia. V našom prípade sme zadávali príkaz:

```
1 C:\Users\Youness>ping 192.168.1.152
```

Listing A.1: Overenie pripojenia cez CMS



```
Command Prompt
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kubip>ping 192.168.1.152

Pinging 192.168.1.152 with 32 bytes of data:
Reply from 192.168.1.152: bytes=32 time=1ms TTL=64
Reply from 192.168.1.152: bytes=32 time=1ms TTL=64
Reply from 192.168.1.152: bytes=32 time<1ms TTL=64
Reply from 192.168.1.152: bytes=32 time<1ms TTL=64

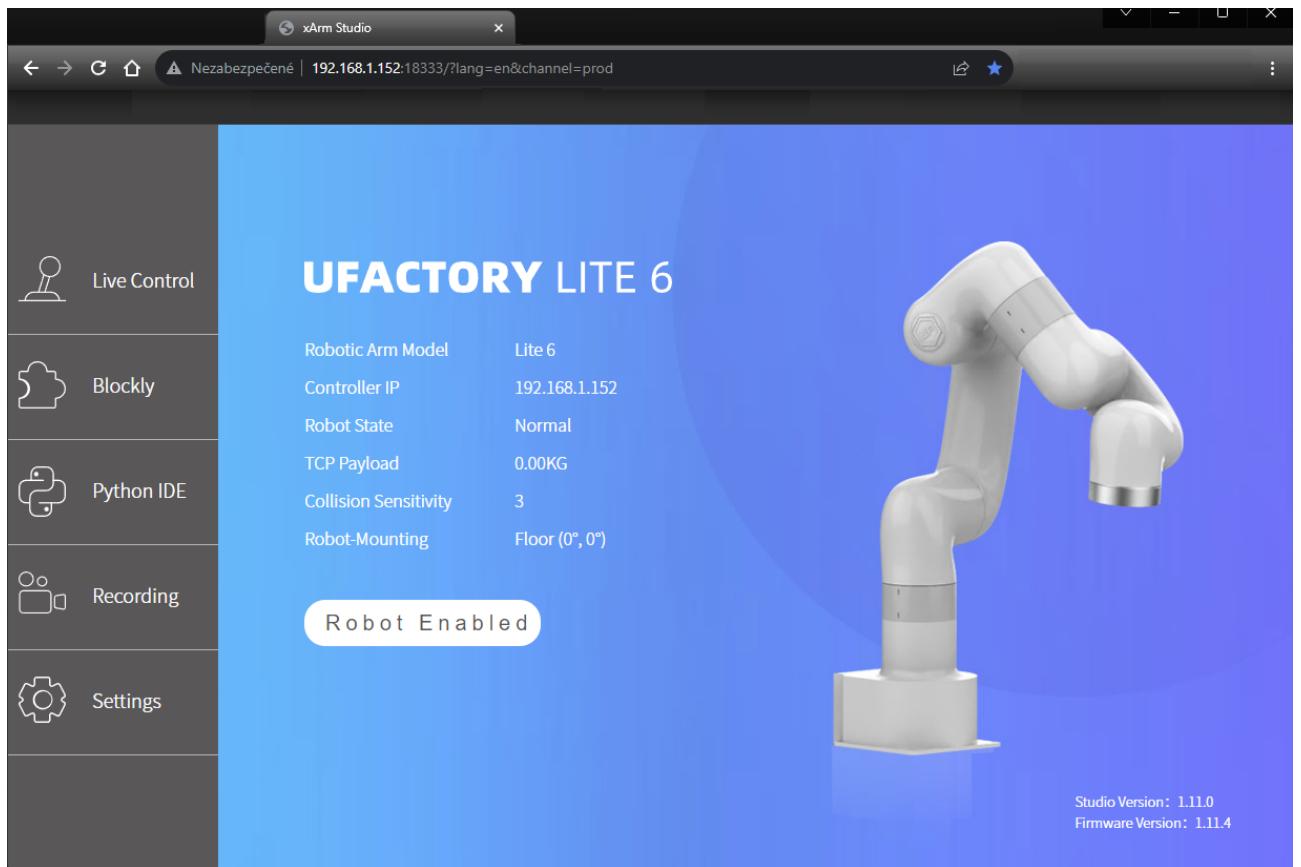
Ping statistics for 192.168.1.152:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Obr. A.9: Overenie pripojenia príkazového riadku

Pred overením pripojenia je dôležité sa uistieť, že kobot nie je deaktivovaný **núdzovým tlačidlom**. Kobot sa aktivuje otočením tlačidla smerom naznačeným šípkou, pričom sa tlačidlo zdvihne, indikátor napájania sa rozsvieti, čo zaručí napájanie kobota.

Overenie spojenia medzi kobotom a počítačom je možné taktiež zistiť aj spustením **UFACTORY Studia**, vložením na miesto URL v internetovom prehliadači našu IP adresu robota, ktorú je potrebné oddeliť dvojbodkou a následne pripisať hodnotu 18333. Adresa URL Ufactory Studia pre kobota prideleného pre túto prácu znie:

***http://192.168.1.152:18333/***.



Obr. A.10: Prostredie Ufactory Studio

Ak je kobot správne zapojený a je s počítačom súčasťou rovnakého lokálneho sieťového prostredia, zobrazí sa obsah aplikácie ako vyšie na obrázku.

### A.3 Implementácia praktickej časti

Ked je hardvérová časť správne zapojená a taktiež vytvorené spojenie medzi kobotom a počítačom, môže nasledovať stiahnutie praktickej časti diplomovej práce.

Je potrebné si stiahnuť celý repozitár tejto práce, ktorý je uložený na tomto **GitHub linku**. Tento projekt je súčasťou xArm-Python-SDK, ktorý predstavuje súpravu na vývoj softvéru od spoločnosti Ufactory. Pridaný je tu priečinok s názvom **DP\_scripts**, ktorý obsahuje hlavné skripty aplikácie a priečinky pre vykonané snímky a objekty.

Následne po naklonovaní projektu je potreba spustiť príkazy:

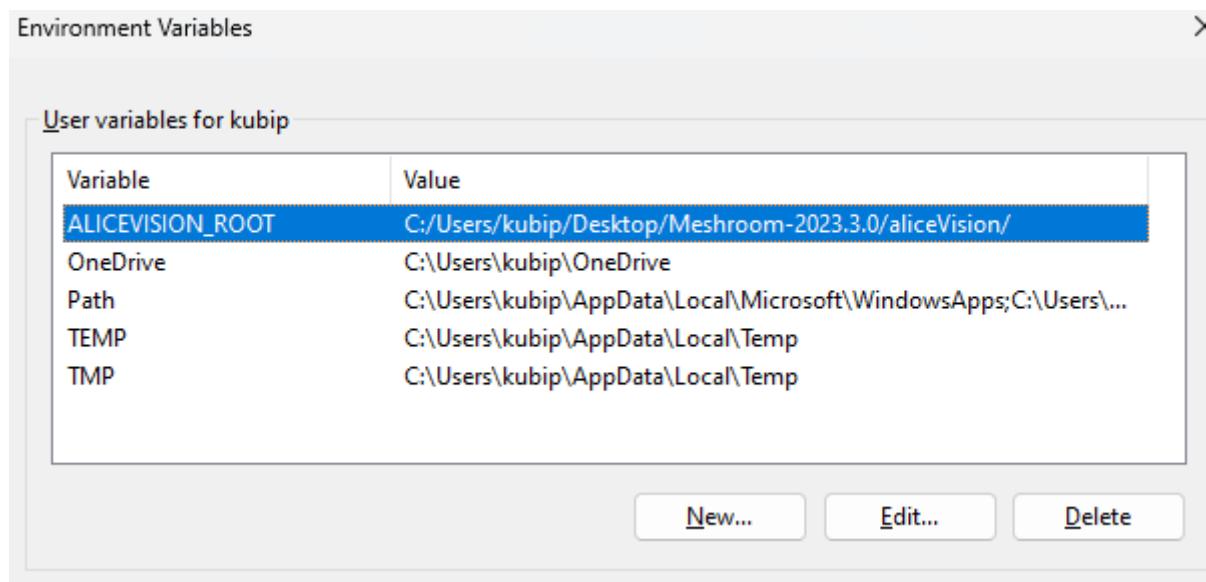
- ***python setup.py install,***
- ***pip install opencv-python,***
- ***pip install pyueye.***

Pre implementáciu kamery bude taktiež potrebné spustiť ***uEye64\_49110\_WHQL.exe***, dostupný na linku: ***Google Disk.***

Kedže súčasťou práce je aj vytvorenie 3D objektu, je potrebné si stiahnuť softvér Meshroom, ktorý je dostupný na oficiálnej stránke: <https://alicevision.org/#meshroom>.

Projektový súbor Meshroom je odporúčané si rozbalit do miesta pod používateľom, napríklad na pracovnú plochu.

Pre používanie softvéru Meshroom priamo cez Python skript je dôležité priradiť environmentálnu premennú. Je potrebné ju vytvoriť s názvom **ALICEVISION\_ROOT** a priradiť cestu k zdrojovému priečinku Meshroom aplikácie, konkrétnie **../Meshroom-2023.3.0/aliceVision/**. Po priradení enviromentálnej premennej, alebo známej ako systémovej premennej, sa odporúča reštartovať počítač a skontrolovať, či je správne uložená.



Obr. A.11: Environmentálne premenné

Po stiahnutí **Meshroom softvéru** a vytvorení enviromentálnej premennej **ALICEVISION\_ROOT** je potreba prepísat v praktickej časti diplomovej práci, konkrétnie v skripte **\xArmPythonSDKmaster\DP\_scripts\meshing\_script.py** premennú s názvom **binPath** a priradiť do nej umiestnenie Meshroom priečinku **bin**. Vzor pre priradenie premennej vyzerá následovne:

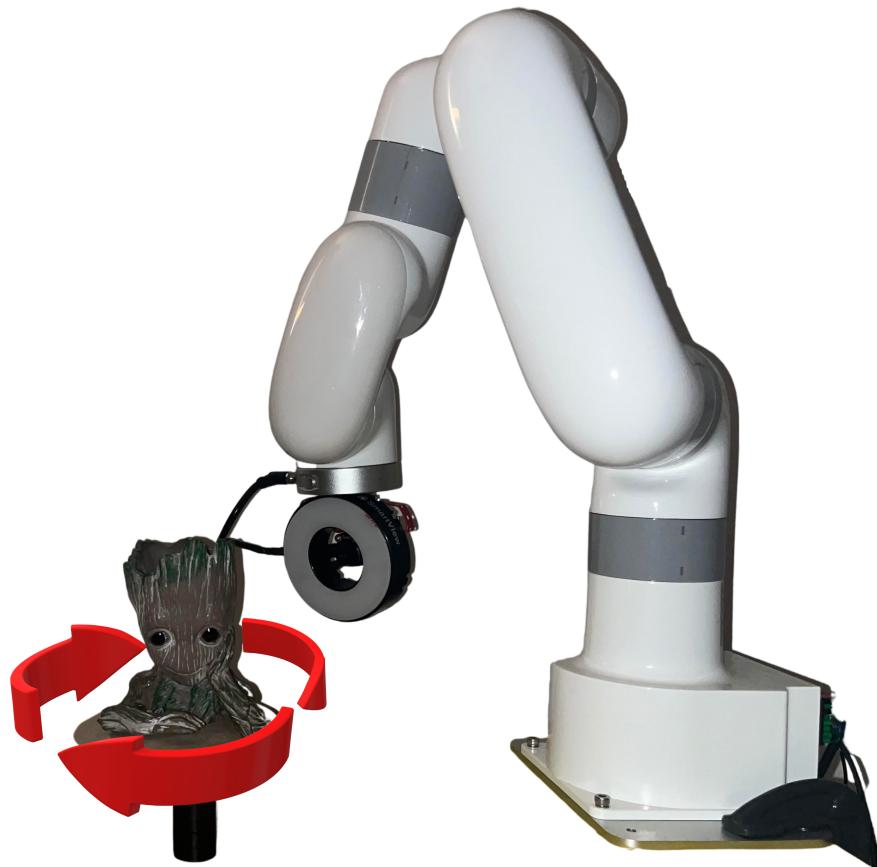
```
binPath = "C:|\Users|\kubip|\Desktop|\Meshroom-2023.3.0|\aliceVision|\bin"
```

## A.4 Spustenie automatizovaného procesu 3D rekonštrukcie

Po úspešných predchádzajúcich hardvérových a softvérových krokoch môže nasledovať spustenie automatizovaného procesu 3D rekonštrukcie. Tieto skripty sa nachádzajú v priečinku ***DP\_scripts***

## A.5 Snímanie objektu bez využitia rotačnej podložky

Vrámcí diplomovej práce boli vytvorené dva spôsoby automatizovanej rekonštrukcie. Prvý spôsob využíva snímanie objektu bez využitia rotačnej podložky, čo znamená, že dataset pre 3D rekonštrukciu sa tvorí krúžením ramena okolo objektu.



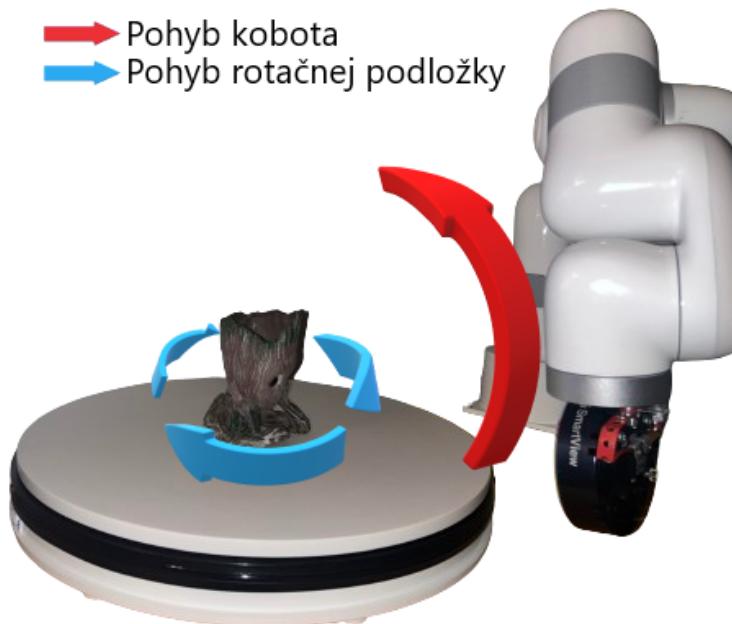
Obr. A.12: Snímanie objektu bez využitia rotačnej podložky

Takýto proces je možné realizovať spustením skriptu -  
***automatic\_3Dreconstructoin\_without\_rotatingplate.py***.

## A.6 Snímanie objektu s využitím rotačnej podložky

Druhý spôsob automatickej 3D rekonštrukcie objektu využíva pre získanie datasetu aj rotačnú podložku na ktorej je umiestnení snímací objekt.

Samotné robotické rameno vykonáva menej polôh a pohybuje sa iba vertikálnym smerom. Vďaka tomuto riešeniu získame všetky možné uhly objektu pričom nie sме limitovaný dĺžkou ramena. Môžeme snímať aj väčšie objekty, ktoré by neboli možné zosnímať pri prvom riešení, ktoré podmieňuje získavanie datasetu snímkov krúžením robotického ramena okolo snímacieho objektu.



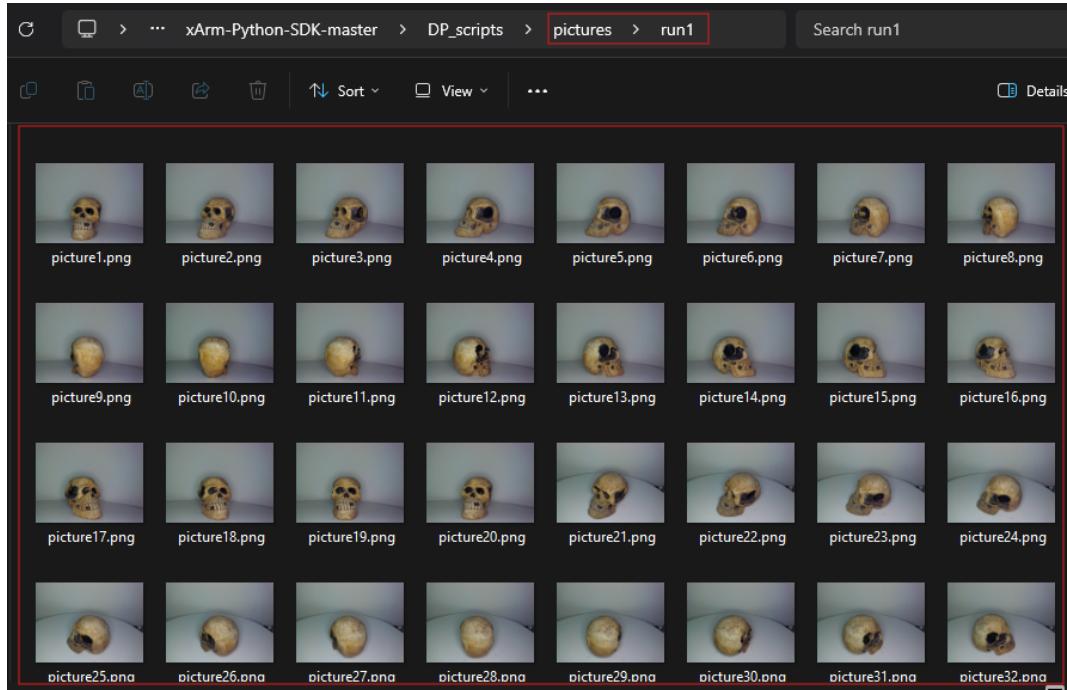
Obr. A.13: Snímanie objektu s využitím rotačnej podložky

Tento proces je možné realizovať spustením skriptu ***automatic\_3Dreconstructoin.py***.

Spustením procesu automatizovanej 3D rekonštrukcie sa začne inicializovať krobot a kamera, následne sa aktivuje rotačná podložka, po ktorom sa začnú vykonávať snímky. Po dosnímaní objektu sa robot s kamerou deinicializujú a nastáva proces 3D rekonštrukcie z datasetu, ktorej výsledkom je 3D objekt.

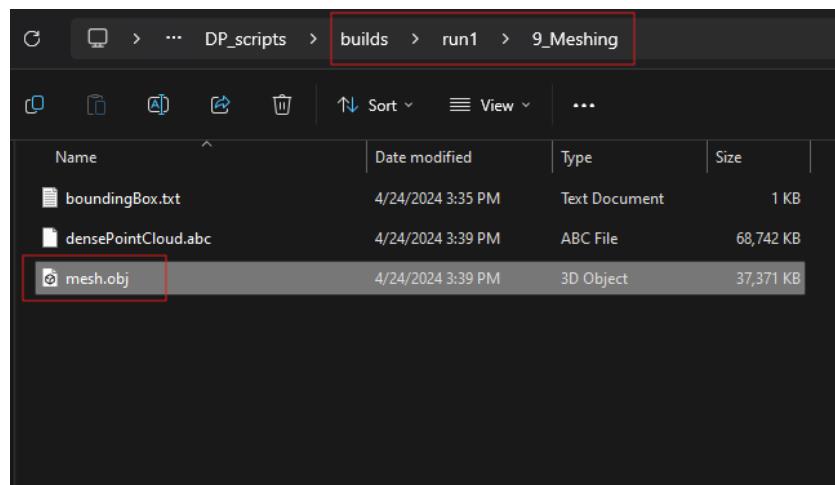
## A.7 Zobrazenie výsledkov

Výsledky procesu sú rozdelené do dvoch priečinkov. Prvý z nich je priečinok **pictures**, ktorý obsahuje celý dataset vytvorených snímkov rozdelený podľa čísla behu.



Obr. A.14: Súbor obsahujúci snímky z procesu

Druhý priečinok **builds** obsahuje údaje vytvorené softvérom Meshroom, ktoré sú taktiež oddelené priečinkom s číslom behu. Nachádzajú sa tu projektové súbory, ktoré obsahujú informácie o postupe fotogrammetrie, vrátane nastavení, použitých obrázkov a ďalších parametrov. Vrámci tejto práce je najdôležitejší podpriečinok **9\_Meshing**, ktorý obsahuje výsledný 3D model.



Obr. A.15: Súbor obsahujúci vytvorený 3D objekt z procesu

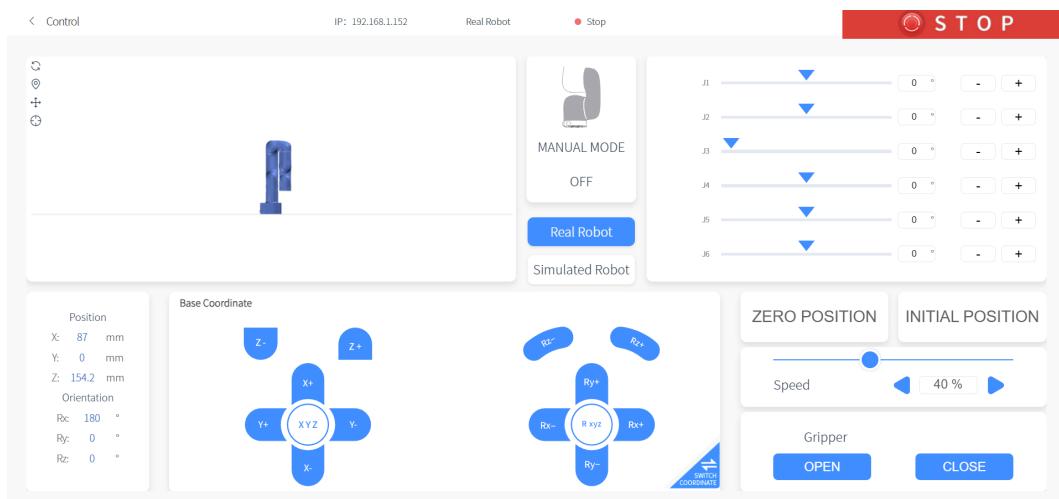
## A.8 Zmena snímacích pozícii

Definované snímacie pozície sú uložené v hlavnom skripte *automatic\_3Dreconstructoin.py* ako pole, ktorého prvky reprezentujú hodnoty pohybu jednotlivých klíbov a značia ich stupeň otočenia: [J1, J2, J3, J4, J5, J6]. Toto pole má názov *sequences*:

```
1 sequences = [  
2     [23.8, 62.9, 100.9, 0.5, 38.7, 105.8],  
3     [4.4, 37.2, 89.8, 39.6, 57.7, 63.3],  
4     [-5, 34.7, 95.5, 46.5, 73.4, 58],  
5     [1.3, 45.1, 84.8, 34, 45.4, 58.1],  
6     [7.5, 52.9, 86.1, 14.4, 33.8, 77.8]  
7 ]
```

Listing A.2: Pole snímacích pozícii

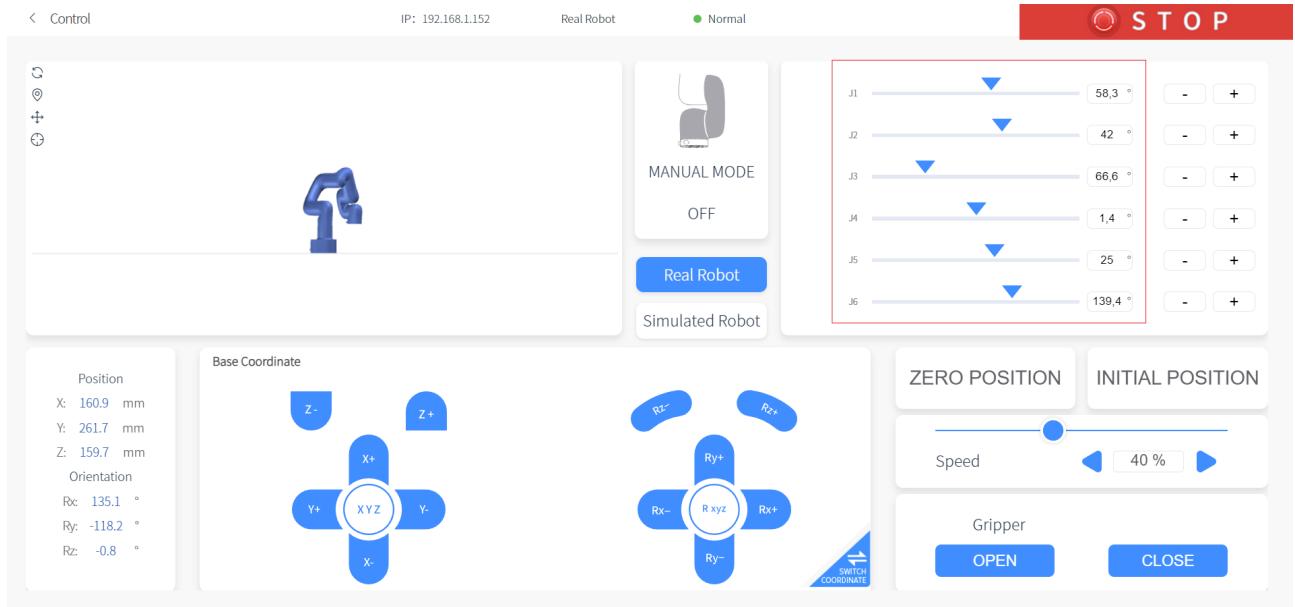
Pre zmenu týchto hodnôt je odporúčané spustiť skript *\xArm-Python-SDK-master\ids\_camera\camera\_live\main.py*, ktorý otvorí okno s živým zobrazením obrazu z kamery. Vďaka tomuto skriptu je možné nastaviť nové požadované pozície v závislosti od typu objektu. Po spustení tohto skriptu a zobrazení skutočného obrazu z kamery je možné pomocou Ufactory studia, spusteného cez odkaz <http://192.168.1.152:18333/>, využiť rozhranie Live Control, a pomocou jednoduchého ovládania ramena v podobe smerových tlačidiel umiestniť rameno do želanej polohy.



Obr. A.16: Live Control rozhranie

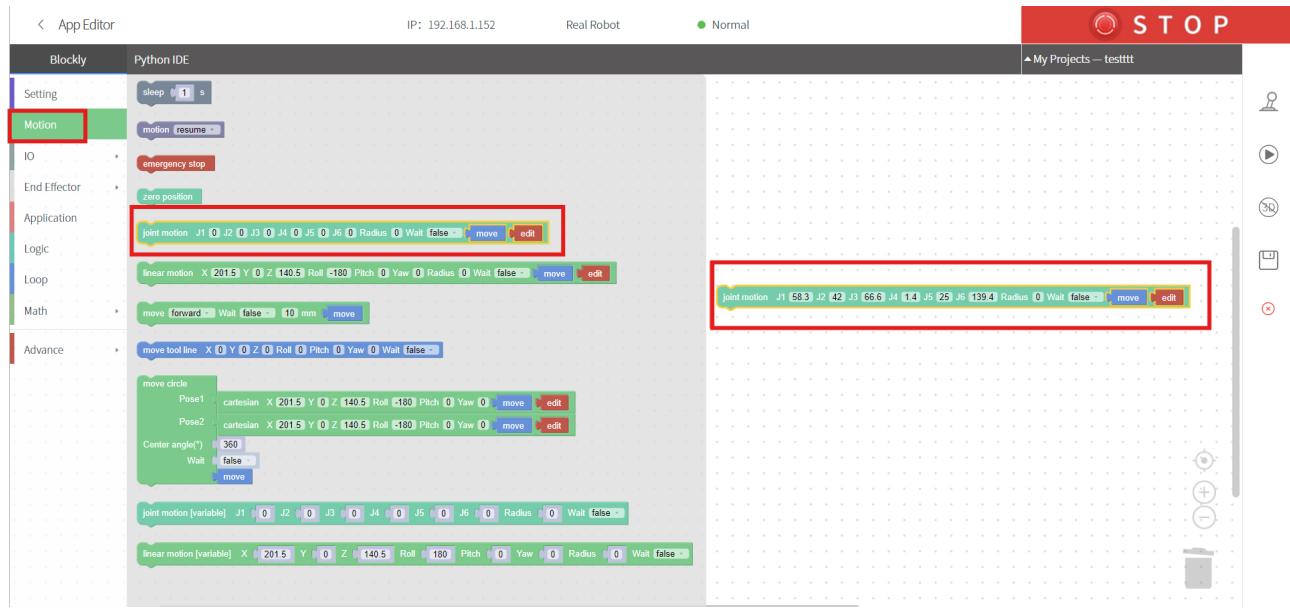
Ked je rameno v požadovanej pozícii, jeho umiestnenie, teda naše želané hodnoty do pola sequences je možné zistíť troma spôsobmi:

- Priamo v rozhraní Live Control na pravej strane, kde sú označené stupne otočenia [J1, J2, J3, J4, J5, J6] :



Obr. A.17: Live Control rozhranie

- V časti Blockly na ľavej strane rozklikneme možnosť "Motion", pod ktorou je umiestnený príkaz "join motion". Tento príkaz môžeme preniesť na voľnú plochu v pravo a automaticky sa nám priradia aktuálne hodnoty umiestnenia ramena:



Obr. A.18: Blockly rozhranie

- Sekcia Python IDE odzrkadluje obsah Blockly rozhrania. Ak je použitý druhý spôsob zobrazenia pozície, je ju možné zobraziť aj tretím spôsobom cez Python IDE pod funkciou *set\_servo\_angle*:

This screenshot shows a software interface with a code editor and a terminal window. The code editor displays Python code for a robot arm, specifically for setting servo angles. A red box highlights a line of code: `code = self._arm.set_servo_angle(angle=[58.3, 42.0, 66.6, 1.4, 25.0, 139.4], speed=self._angle_speed, mvacc=self._angle_acc, wait=False, radius=0)`. The terminal window at the bottom shows the output of the code execution.

```
IP: 192.168.1.152 Real Robot Normal STOP
```

```
This module function is not yet stable (can be tried, but not used in production environments)
```

```
BlockyToPython testhttp x
camera_points_22.jar BlockyToPython/testhttp.py
    print(*args, **kwargs)

@property
def _is_alive(self):
    if self._alive and self._arm.connected and self._arm.error_code == 0:
        if self._arm.state == 5:
            cnt = 0
            while self._arm.state == 5 and cnt < 5:
                cnt += 1
                time.sleep(0.1)
            return self._arm.state < 4
    else:
        return False

# Robot Main Run
def run(self):
    try:
        code = self._arm.set_servo_angle(angle=[58.3, 42.0, 66.6, 1.4, 25.0, 139.4], speed=self._angle_speed, mvacc=self._angle_acc, wait=False, radius=0)
        if not self._check_code(code, "set_servo_angle"):
            return
    except Exception as e:
        self._print("MainException: {}".format(e))
    self._alive = False
    self._arm.release_error_warn_changed_callback(self._error_warn_changed_callback)
    self._arm.release_state_changed_callback(self._state_changed_callback)
    if hasattr(self._arm, 'release_count_changed_callback'):
        self._arm.release_count_changed_callback(self._count_changed_callback)

if __name__ == '__main__':
    RobotMain.print('XArm-Python-SDK Version:{}'.format(version.__version__))
    arm = XArmAPI('192.168.1.152', baud_checkset=False)
    robot_main = RobotMain(arm)
    robot_main.run()
```

Obr. A.19: Python IDE rozhranie

Týmto spôsobom je možné určiť nové snímacie pozície, z ktorých sa bude realizovať výsledný dataset pre 3D rekonštrukciu objektu.