Primjena vizijskog sustava za vođenje robota

Žmegač, Iva

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Iva Žmegač

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc. dr. sc. Marko Švaco, mag. ing.

Student:

Iva Žmegač

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentorima doc.dr.sc. Marku Švaci i dr.sc. Bojanu Šekoranji na ustupljenoj opremi, pomoći, savjetima i strpljenju koje su mi ustupili prilikom izrade ovog rada.

Također zahvaljujem mojoj obitelji, prijateljima i dečku koji su mi bili podrška tokom studiranja te prilikom izrade ovog rada.

U Zagrebu, 2022. godine

Iva Žmegač

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Iva Žmegač** JMBAG: **0035216881**

Naslov rada na hrvatskom jeziku: **Primjena vizijskog sustava za vođenje robota**

Naslov rada na engleskom jeziku: **Application of a vision system for robot guiding**

Opis zadatka:

U završnom radu potrebno je razviti programsku podršku za vizijski sustav koja omogućuje praćenje kretnji ljudske ruke. Razvijena programska podrška treba sadržavati model za preslikavanje vrijednosti koordinata iz realnog svijeta snimljenih vizijskim sustavom u koordinatni sustav robota. Dobivene podatke o kretanjima potrebno je prenijeti na robota u realnom vremenu. Razvijene algoritme potrebno je verificirati na postavu u Laboratoriju za računalnu inteligenciju.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2021.

Zadatak zadao:
Doc. dr. sc. Marko Švaco

Datum predaje rada:
1. rok: 24. 2. 2022.
2. rok (izvanredni): 6. 7. 2022.
3. rok: 22. 9. 2022.

Predviđeni datumi obrane:
1. rok: 28. 2. – 4. 3. 2022.
2. rok (izvanredni): 8. 7. 2022.
3. rok: 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	**Opis**
engl.	engleski
X,Y,Z	osi Kartezijevog koordinatnog sustava
RX,RY,RZ	osi rotacije
SDK	engl. *Software Development Kit* – komplet za razvoj softvera
API	engl. *Application Programming Interface* – sučelje za programiranje
LED	engl. *Light Emitting Diode* – dioda koja emitira svjetlo
USB	engl. *Universal Serial Bus* – univerzalna serijska sabirnica
VR	engl. *Virtual Reality* – virtualna stvarnost
VR/AR	engl. Virtual Reality/Augmented Reality – Virtualna stvarnost/Proširena stvarnost
TCP	engl. *Transmission Control Protocol* – protokol upravljanja prijenosom
RTDE	engl. *Real-Time Data Exchange* - razmjena podataka u stvarnom vremenu
XML-RPC	engl. *XML - Remote Procedure Call* - metoda udaljenog poziva procedure
GUI	engl. *graphical user interface* - grafičko korisničko sučelje

SAŽETAK

Rad donosi opis i pregled područja virtualne stvarnosti i robotike te je ovaj rad eksperimentalnog tipa jer integrira oba područja. U prvom dijelu rada korišten je *Leap Motion Kontroler* kao senzor koji registrira pokrete ruke, dok je u drugom dijelu korištena robotska ruka *UNIVERSAL ROBOTS 3* koja ponavlja pokrete registrirane *Leap Motion* Kontrolerom. Sva korištena oprema nalazi se u Regionalnom centru izvrsnosti za robotske tehnologije (CRTA) u sklopu Fakulteta strojarstva i brodogradnje. Opisan je postupak prikupljanja podataka sa senzora, slanja tih podataka robotu te čitanja i izvršavanja naredbi sa strane robota. Podaci koji se šalju robotu su koordinate pozicije ruke koje robot preko svoje upravljačke jedinice očitava i na koncu izvršava. Za povezivanje *Leap Motion* Kontrolera i robota korišteno je prijenosno računalo koje koristi Windows 64-bit operativni sustav te TCP/IP protokol. Također, opisana je problematika prijenosa ljudskih pokreta na pokrete robota.

Ključne riječi:

virtualna stvarnost, robotika, *Leap Motion, UNIVERSAL ROBOTS*, ruka, snimanje pokreta, *LeapC*

SUMMARY

The paper provides a description and overview of the fields of virtual reality and robotics, and this paper is of experimental type because it integrates both fields. In the first part of the paper, the *Leap Motion Controller* is used as a sensor which registers hand movements, while in the second part, the *UNIVERSAL ROBOTS 3* robotic arm is used, which repeats the movements registered by the *Leap Motion Controller*. All the equipment used is located in the Regional Center of Excellence for Robotic Technologies (CRTA) within the Faculty of Mechanical Engineering and Naval Architecture. It's describing the process of collecting data from the sensor, sending that data to the robot, and reading and executing the command by the robot. Data sent to the robot are the coordinates of the hand's position, which the robot reads through its control panel and finally executes. For connecting the *Leap Motion Controller* and the robot, a laptop is used that uses Windows 64-bit operating system and the TCP/IP protocol. Also, it's describing the issue of transferring human movements to robot movements.

Key words:

virtual reality, robotics, *Leap Motion*, *UNIVERSAL ROBOTS*, hand, motion capture, *LeapC*

1. UVOD

U digitalizaciji stvarnosti i stvaranju virtualne stvarnosti važnu ulogu ima pokret. Pokret određuje interakciju među ljudima, među predmetima rada, kao i među napravama ili strojevima s okolinom. U automatskim robotiziranim procesima, kako bi strojevi mogli surađivati u stvarnom svijetu, moraju znati svoj trenutni položaj, stanje okoline te moraju biti sposobni predviđati buduća stanja. Kako bi se navedeno realiziralo potrebno je kvalitetno snimati pokrete fizičkog rada i pretočiti ih u virtualnu stvarnost. [1]

Robotika je područje koje povezuje područja strojarstva, elektrotehnike i računarstva s matematičkom podlogom te pruža širok spektar mogućnosti u rješavanju različitih problema. U industriji robotika ima značajnu ulogu u povećanju produktivnosti procesa, kvalitete proizvoda, efikasnog iskorištavanja materijala te sigurnosti radnika. Glavni razlog za ove prednosti je velika preciznost i ponovljivost robota u radu naspram čovjeka. Rukovanje predmetima rada predstavlja težak zadatak kako za čovjeka, tako i za robota te se rukovanje često želi poboljšati i konstantno traži mjesto za napredak. Od velikog je značaja mogućnost upravljanja i navođenja robota s daljine te bi se to moglo postići integriranjem virtualne stvarnosti s robotikom.

U sljedećim poglavljima istražit će se mogućnost objedinjavanja navedenih područja te će se napraviti eksperiment s uređajem *Leap Motion*, koji će poslužiti kao senzor, povezanim na robotsku ruku *UR3*. Za povezivanje *Leap Motion Kontrolera* i robota korišteno je prijenosno računalo koje koristi Windows 64-bit operativni sustav te TCP/IP protokol.

2. *Leap Motion* Kontroler

2.1. Struktura *Leap Motion* Kontrolera

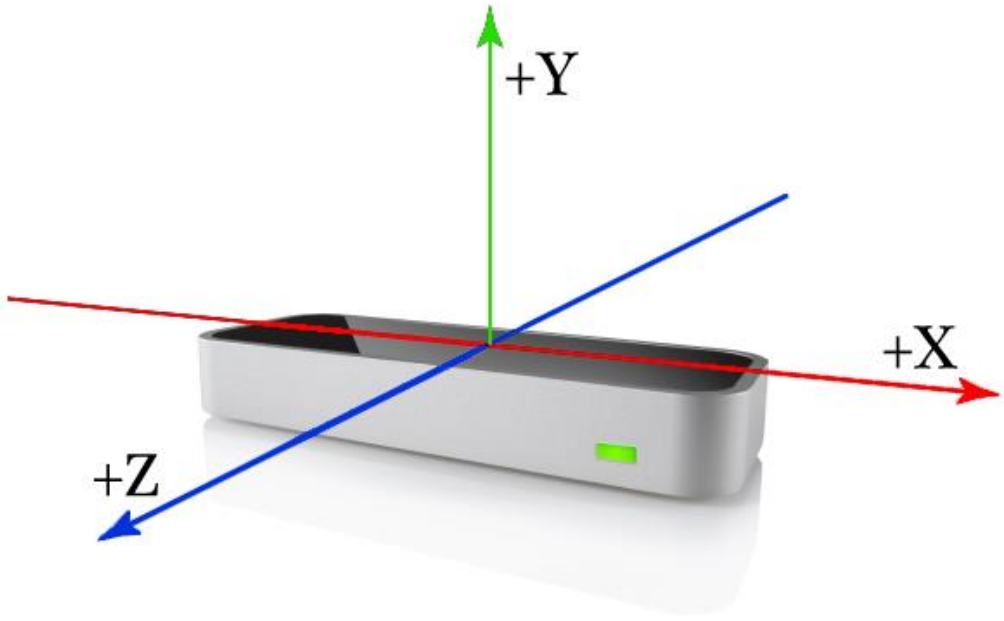
Leap Motion Kontroler proizvod je kompanije *Ultraleap*. To je optički uređaj koji snima i bilježi pokrete ruku i prstiju korisnika za interakciju s digitalnim sadržajem. Brz je i precizan, može se koristiti s računalima, pričvrstiti na VR naočale, biti integriran u hardverska rješenja te ima mnogo drugih načina upotrebe.

Slika 1. ***Leap Motion Controller*** **[2]**

Kontroler se sastoji od dvije infracrvene kamere i tri infracrvene LED diode koje prate infracrvenu svjetlost na valnoj duljini od 850 nanometara, što je izvan spektra vidljive svjetlosti. LED diode pulsiraju sinkronizirano s brzinom okvira kamere (engl. Camera framerate – broj sličica u sekundi kamere), što omogućuje znatno nižu potrošnju energije i povećani intenzitet. Brzina ažuriranja slika obično iznosi 120 Hz, a može generirati gotovo 200 slika u sekundi. Svaki put kada to učine, kamere šalju podatke računalu. [3] Slike koje snima kamera su sive stereo slike odvojene u lijevu i desnu kameru te su uglavnom vidljivi samo predmeti koji su izravno osvjetljeni LED diodama što su najčešće ruke, no dnevna svjetlost i žarulje također mogu biti vidljive u infracrvenom svjetlu. Efektivni domet Kontrolera proteže se od približno 2,5 cm do 60 cm iznad uređaja u vidnom polju od 140x120°. [4][5]

Slika 2. Struktura *Leap Motion* Kontrolera [6]

Koordinatni sustav kojeg koristi Kontroler je desnokretni kartezijski koordinatni sustav s ishodištem na vrhu uređaja gdje pozitivan smjer Y osi gleda prema korisniku. [7]



Slika 3. Prikaz koordinatnog sustava *Leap Motion* Kontrolera [7]

2.2. *Leap Motion* softver za praćenje ruku

Kontroler se spaja USB priključkom na računalo te prikuplja podatke sa senzora u vlastitu lokalnu memoriju i izvodi sve potrebne prilagodbe rezolucije te ih zatim prenosi putem USB-a u *Leap Motion* softver za praćenje ruku. Softver koristi slike za generiranje virtualnog modela pokreta ruku te se modeliraju ne samo dlan ili vrhovi prstiju, već i zglobovi i kosti unutar ruke što omogućava točno predviđanje položaja prstiju čak i ako su skriveni od pogleda.

Softver koristi napredne algoritme na neobrađenim podacima sa senzora. Nakon filtriranja pozadinskih objekata (kao što su glave) i ambijentalnog osvjetljenja, slike se analiziraju kako bi se rekonstruirao 3D prikaz onoga što uređaj „vidi“ te kako bi se dobile detaljne informacije o pokretima te o broju ruku i prstiju. Softver te informacije bilježi kao niz snimaka (*Frame*) koji sadrže sve podatke o praćenju te ih provodi kroz tzv. transportni protokol te komunicira s programom *Leap Motion Control Panel,* s izvornim sučeljem (putem TCP protokola) i internetskim klijentskim sučeljem (putem *WebSocket* komunikacije). [3]

Leap Motion SDK[1] (engl. *software development kit* – hrv. Pribor za razvijanje softvera) pruža dvije varijante *API*-ja[2] (engl. *Application Programming Interface* – hrv. Programersko sučelje aplikacije) za dobivanje podataka s *Leap Motion* softvera: izvorno sučelje i *WebSocket* sučelje. Izvorno sučelje je dinamička knjižnica nazvana *LeapC* koju možete koristiti za stvaranje novih aplikacija izravno u C programu. *WebSocket* sučelje i *JavaScript* klijentska knjižnica omogućuju vam stvaranje internetskih aplikacija. [8]

Softver nudi podršku za sve najraširenije operativne sustave osobnih računala (Windows, Mac i Linux) te ima više inačica koje su se mijenjale i nadograđivale tijekom godina. Postoji pet inačica softvera: V2 (koja više nije podržana), V3 pod nazivom Orion, V4 pod nazivom Orion druge generacije te najnovija verzija softvera V5 pod nazivom *Gemini*. [9]

[1] - SDK - Pribor za razvijanje softvera (engl. software development kit) je skup pribora za razvijanje softvera koje omogućuje stvaranje izvršnog softvera (aplikacija) za određeni softverski paket, softverski okvir, platformu sklopovlja, računalni sustav, igraću konzolu, operacijski sustav ili slično.[10]

[2] - API - Aplikacijsko programsko sučelje (engl. application programming interface) ili sučelje za programiranje aplikacija je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava ili nekog drugog složenog programa kao standardne knjižnice rutina (funkcija, procedura, metoda), struktura podataka, objekata i protokola. [11]

Posljednja verzija podržava rješenja virtualne i proširene stvarnosti (VR/AR) i nudi različita poboljšanja, a posebno u praćenju obje ruke kada su blizu jedna drugoj ili se preklapaju. Sadrži podršku samo za jedan jezik – C, te je napuštena podrška za ostale programske jezike koje su starije verzije sadržavale. Ova promjena uvedena je kako bi se ubrzalo vrijeme procesiranja budući da je jezik C onaj koji se najbolje može optimirati. Također, najnoviji softver podržava novi način orijentacije – s vrha zaslona (engl. *screentop*). [9]

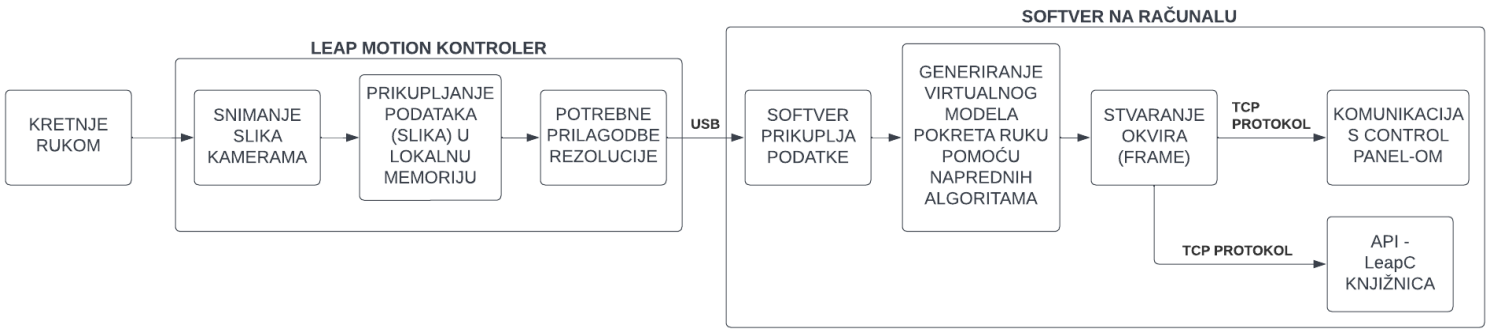
U tablici 1. prikazan je sustav mjernih jedinica za mjerenje fizikalnih veličina koje koristi *Leap Motion* API.

Tablica 1. Sustav mjernih jedinica *Leap Motion* Kontrolera [12]

Fizikalna veličina	Oznaka	Mjerna jedinica
Udaljenost	l	[mm]
Vrijeme	t	[μs]
Brzina	v	[mm/s]
Kut	α	[rad]

2.3. Blok dijagram *Leap Motion* sustava za praćenje ruke

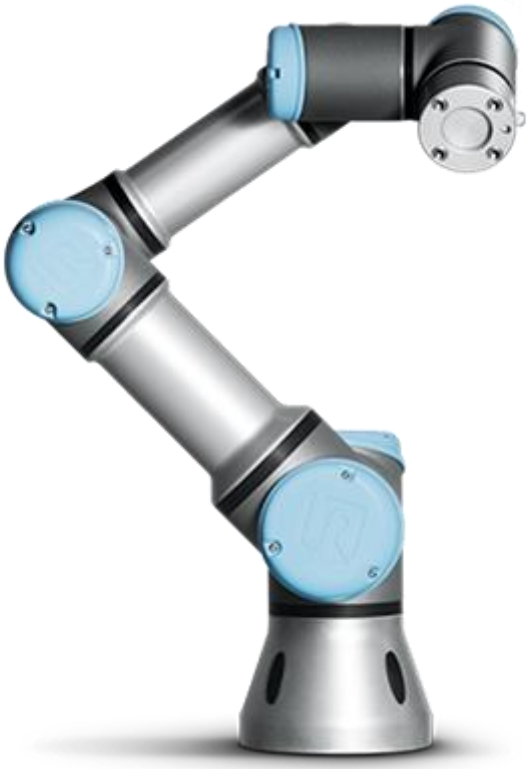
Pošto će se u ovom radu koristiti izvorno sučelje, tj. *LeapC* knjižnica, u blok dijagramu opisan je taj tip komunikacije.



Slika 4. Blok dijagram sustava *Leap Motion*

3. UNIVERSAL ROBOTS (UR) ROBOT

UR3 robotska je ruka poduzeća UNIVERSAL ROBOTS te je najmanji od robota UR e-serije. Ima kompaktan oblik te je zato koristan za rad u uskim radnim prostorima. Prikladan je za ugradnju na radne stolove ili izravno u strojeve. Težak je samo 11 kg, ali ima nosivost od 3 kg, rotaciju od ±360 stupnjeva na svim zglobovima zapešća i beskonačnu rotaciju na krajnjem zglobu. Ima raspon kretanja od 500 mm te radi s preciznošću od 0,1 mm. [13]

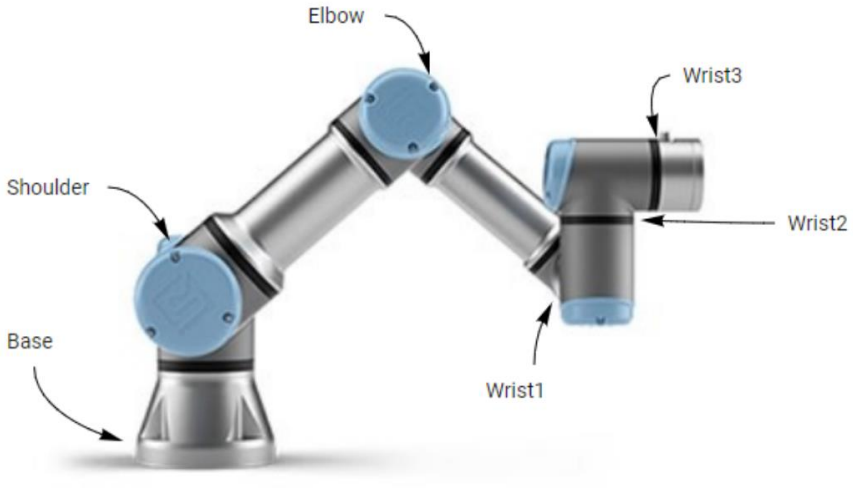


Slika 5. *UR3* robotska ruka [14]

Robot se nalazi u laboratoriju na postolju gdje je spojen na mrežu Ethernet priključkom. Upute za postavljanje robota dane su u [15].

3.1. Struktura UR robota

Robot se sastoji od šest zglobova koji mu omogućuju šest stupnjeva slobode gibanja i dvije aluminijske cijevi koje povezuju bazu s alatom robota. Nomenklatura glasi: baza (eng. base), rame (eng. shoulder), lakat (eng. elbow), zglob 1 (eng. wrist 1), zglob 2 (eng. wrist 2) i zglob 3 (eng. wrist 3).



Slika 6. Nomenklatura *UR3* robota [16]

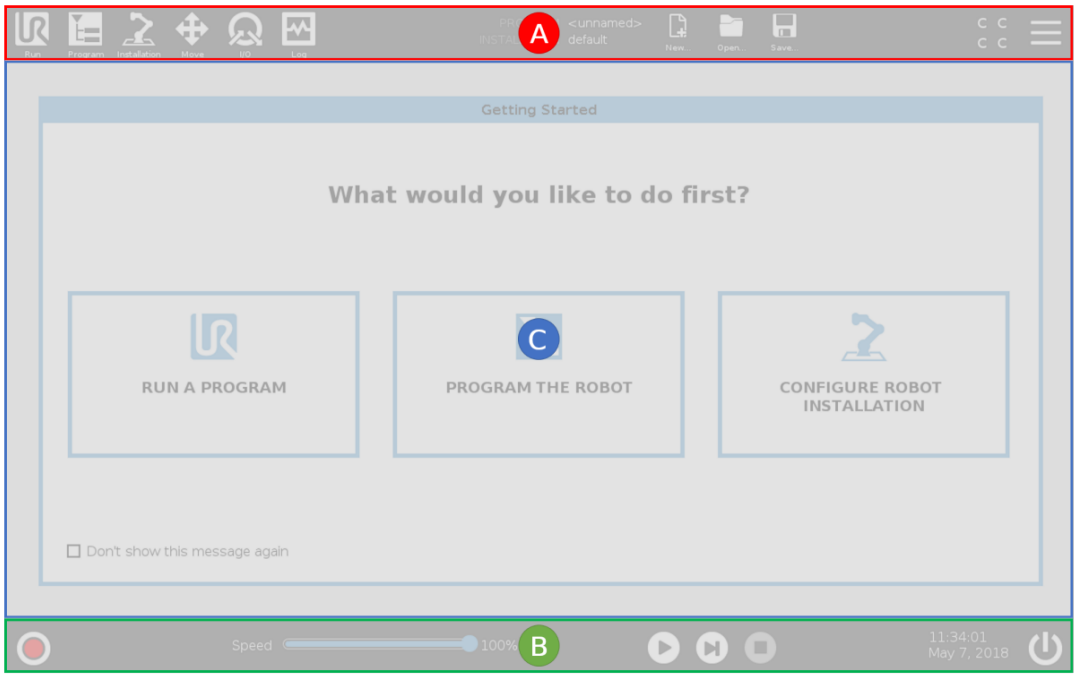
Može ga se ručno namjestiti u bilo koji položaj te dodatno namjestiti s interaktivnom jedinicom *Teach pendant.*

Teach pendant jedinica je naziv za uređaj koji dolazi uz sam robot te služi za upravljanje i pomicanje robota te za programiranje kretnji robota. UR roboti imaju dodirni zaslon koji se koristi kao korisničko sučelje (engl. *PolyScope*). Korisničko sučelje se sastoji od tri zone:

A: Zaglavlje s karticama/ikonama koje čine interaktivne zaslone dostupnima korisniku.

B: Podnožje s gumbima koji kontroliraju učitane programe.

C: Zaslon s poljima koja upravljaju i nadziru kretnje robota. [17]



Slika 7. Korisničko sučelje robota – *PolyScope* [17]

4. KORIŠTENJE LEAP MOTION CONTROLLER-A

4.1. Povezivanje s *Leap Motion* softverom

Za povezivanje vlastite aplikacije na *Leap Motion* softver potrebno je s njihove internetske stranice preuzeti jednu od inačica softvera te ju instalirati.

Inačice softvera V4 i V5 koriste programski jezik C. Inačice softvera V1, V2 i V3 podržavaju i druge programske jezike te imaju knjižnice napisane za jezike C++, Python i Java Script. [18] No, budući da inačica V3 podržava samo inačicu 2.7 programskog jezika Python koja više nije podržana, kako bi se *Leap Motion* softver koristio s programskim jezikom Python inačice 3.X potrebno je postaviti *Wrapper*[1] (engl. *Wrapper*) oko podržane knjižnice napisane u C jeziku. [19]

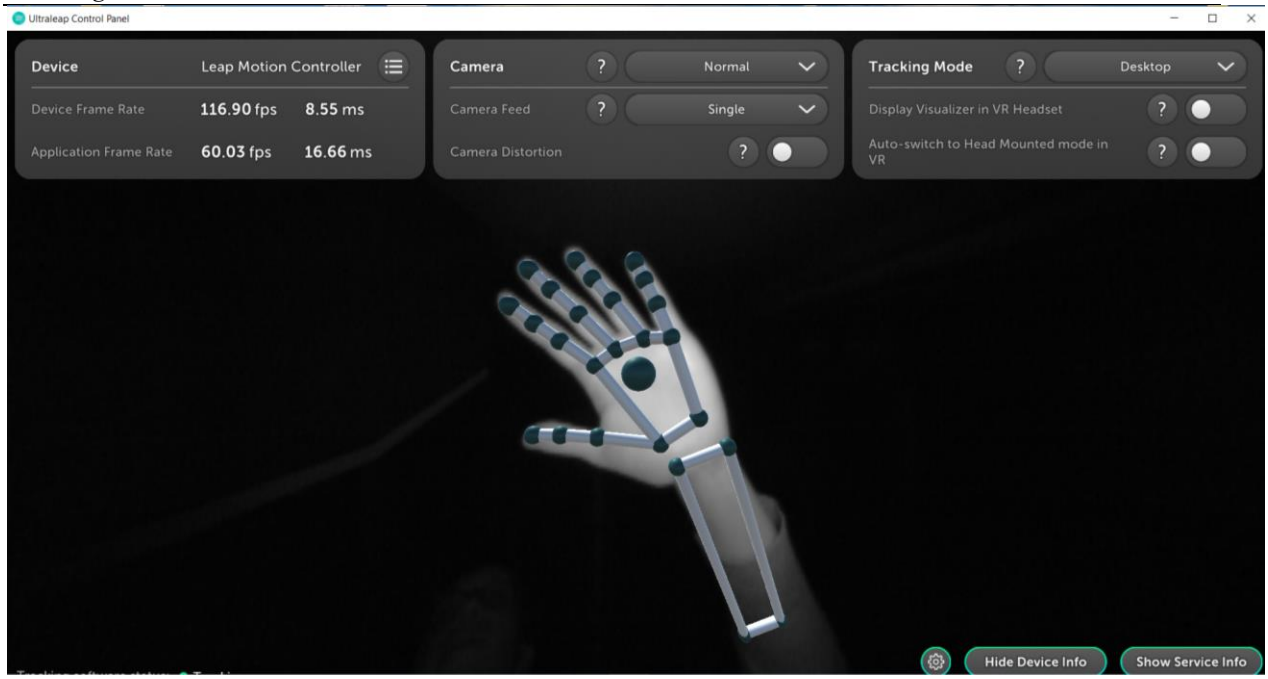
U ovom radu korištena je posljednja inačica softvera V5 - Gemini. Sučelje komunicira putem programskog jezika C te je zato vlastiti softver također programiran u C jeziku. Knjižnica *LeapC* (SDK) koja se automatski dohvaća prilikom instalacije također sadrži primjere (eng. *Samples*) koji su vrlo korisni i sadrže kod koji je potreban za izradu ovog rada. Kod je napisan prema primjeru *PollingSample.c*. [20]

4.2. Pokretanje *Leap Motion* softvera

Potrebno je pokrenuti *Leap Motion* softver kako bi se utvrdilo da softver funkcionira i snima pokrete ruku. Potrebno je *Leap Motion* Kontroler spojiti preko USB porta na računalo te pravilno odložiti na tvrdu podlogu, okrenuti ga tako da je vidljivo crveno svijetlo te ga obrisati kako bi kamera što preciznije pratila pokrete. Nakon toga je potrebno pokrenuti *Ultraleap Control Panel.exe* te se otvara aplikacija upravljačke ploče. Upravljačka ploča će prikazati ruke koje se prate, kao i infracrvene slike izravno iz modula kamere. Također, mogu se podesiti postavke za praćenje pokreta.

Leap Motion Service je softver koji na računalu obrađuje slike te filtrira pozadinske objekte (sve što nije šaka) i kompenzira osvjetljenja. Na ekranu računala je vidljivo da se približavanjem ruke uređaju okoliš zatamni dok se udaljavanjem od uređaja posvijetli.

[1] – *Wraper* – Funkcija je da programe ili kodove „omotava“ oko drugih komponenti programa. Često se koristi za osiguravanje kompatibilnosti između različitih softverskih struktura. [19]



Slika 8. Ultraleap upravljačka ploča

4.3. Stvaranje vlastitog projekta te pisanje koda za prikaz koordinata pozicije ruke

Za stvaranje vlastitog projekta korišten je softver *CMake* koji povezuje projekt na knjižnicu *Leap Motion*-a, te je za pisanje programskog koda korišten softver *Microsoft Visual Studio.* [21][22]

Programski kod u C jeziku napisan je prema primjeru *PollingSample.c* skripte preuzete iz knjižnice *Leap Motion*-a.

Za početak potrebno je uvesti knjižnicu *Leap Motion*-a te se u c jeziku za to koristi naredba *#include* na ovaj način:

```
#include "LeapC.h"
#include "ExampleConnection.h"
```

Zatim se potrebno spojiti na *Leap Motion* Kontroler koristeći naredbu *OpenConnection()*[23]:

```
int main(int argc, char** argv) {
  /************** connecting ultraleap **************/
  OpenConnection();
  while (!IsConnected)
    millisleep(100); //wait a bit to let the connection complete
```

Slijedi dio koda koji služi za dohvaćanje koordinata dlana ruke. Za ovaj rad nije bilo potrebno preuzimati koordinate prstiju. Koordinate koje se ispisuju na ekran prikazane su u milimetrima [mm]. Naredba *GetFrame()* koristi se za dohvaćanje okvira u kojem se nalazi ruka u tom trenu. Ovaj dio koda ispisuje okvir s koordinatama X,Y,Z ruku. Brzina ažuriranja je 120 Hz što iznosi 0.0083 sekundi. Pošto je to robotu nemoguće pratiti potrebno je podesiti da se ne dohvaća svaki okvir, a da se pritom pazi da se dohvaća dovoljan broj okvira kako bi se pratile kretnje ruke. To se postiže naredbom *(lastFrameID+20)* kojom se dohvaća svaki dvadeseti okvir. Ta brojka dobivena je eksperimentalno, isprobavajući s brojevima između 10 (podaci se šalju svakih 0.083 sekunde) i 100 (podaci se šalju svakih 0.83 sekunde). Svaki dvadeseti okvir dohvaća okvire u razmaku od 0.166 sekundi te se to pokazalo kao optimalno vrijeme za slanje podataka robotu.

Ako ruka nesta iz vidnog polja Kontrolera, koordinate se prestaju slati te su posljednje zabilježene koordinate na rubu vidnog polja Kontrolera. Kako bi se izbjeglo sudaranje robota u postolje ili u samog sebe, prostor za kretanje robota je sužen što je objašnjeno u 5.4.3. poglavlju.

```
int64_t lastFrameID = 0; //The last frame received

for (;;) {

    LEAP_TRACKING_EVENT* frame = GetFrame();
    if (frame && (frame->tracking_frame_id > (lastFrameID+40))) {
      lastFrameID = frame->tracking_frame_id;
```

Sljedeći dio koda opisuje koordinate X,Y,Z osi te kutove rotacija dlana ruku oko tih osi te ih ispisuje na ekran pomoću naredbe *printf()*. Koristit će se naredba *palm.position* za koordinate X,Y,Z osi te naredba *palm.orientation* za kutove rotacija X,Y,Z osi. Kontroler također prepoznaje radi li se o lijevoj ili desnoj ruci.

```
for (uint32_t h = 0; h < frame->nHands; h++) {
LEAP_HAND* hand = &frame->pHands[h];
  printf("   Hand id %i is a %s hand with position (%f, %f, %f, %f, %f, %f).\n",
          hand->id,
          (hand->type == eLeapHandType_Left ? "left" : "right"),
          hand->palm.position.x,
          hand->palm.position.y,
          hand->palm.position.z,
          hand->palm.orientation.x,
          hand->palm.orientation.y,
          hand->palm.orientation.z);
       }
```

5. SLANJE KOORDINATA UR ROBOTU

5.1. Vrste komunikacijskih sučelja UR robota

UR robot može komunicirati s vanjskim uređajima putem različitih vrsta komunikacijskih sučelja.

Primarna/Sekundarna sučelja – Servere za slanje podataka o stanju robota i primanje *URScript* naredbi osigurava UR kontroler. Primarno sučelje prenosi podatke o stanju robota i dodatne poruke s brzinom ažuriranja od 10 Hz. Sekundarno sučelje prenosi samo podatke o stanju robota s brzinom ažuriranja od 10 Hz.

Sučelja u stvarnom vremenu – Funkcionalnost je slična kao kod primarno/sekundarnih sučelja. Kontroler šalje podatke o stanju robota i prima *URScript* naredbe. Glavna razlika je brzina ažuriranja koja iznosi 125-500 Hz ovisno o seriji kontrolera.

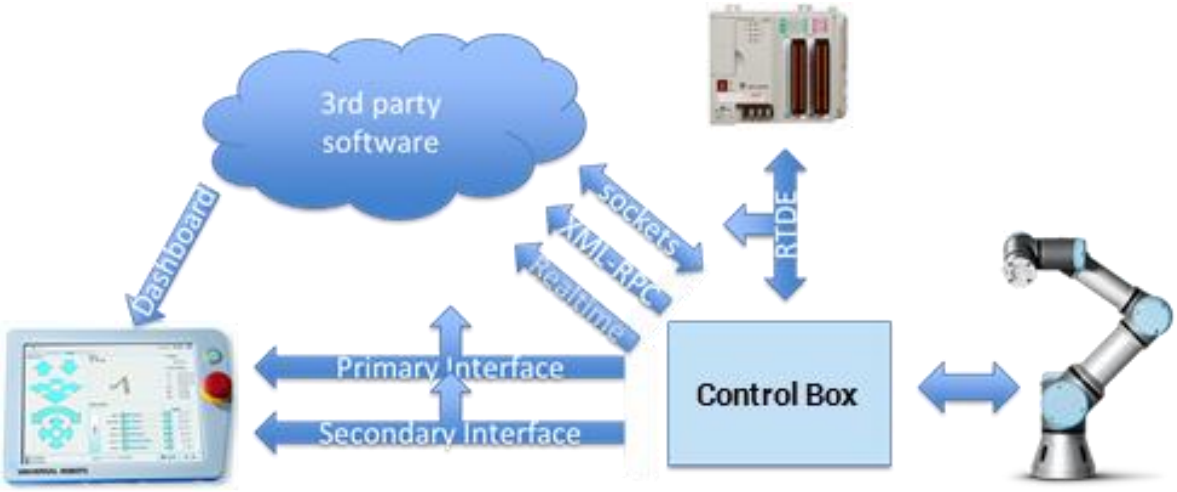
Poslužitelj nadzorne ploče - Robotom se može upravljati s daljine slanjem jednostavnih naredbi u grafičko korisničko sučelje (GUI) preko TCP/IP-a. Glavne funkcije poslužitelja su učitavanje, reprodukcija, pauziranje i zaustavljanje programa robota, postavljanje razine korisničkog pristupa i primanje povratnih informacija o stanju robota.

Socket komunikacija (robot je klijent) - UR robot komunicira s vanjskom opremom putem TCP/IP protokola. Podaci se prenose putem *socket* komunikacije između robota i drugog uređaja. Robot igra ulogu klijenta, a drugi uređaji igraju ulogu poslužitelja. *URScript* daje naredbe koje otvaraju i zatvaraju *socket*-e, te šalju i primaju različite podatke.

Razmjena podataka u stvarnom vremenu (RTDE - *Real-Time Data Exchange*) - RTDE je dizajniran kao robusna zamjena za sučelje u stvarnom vremenu što omogućuje da UR kontroler prenosi prilagođene podatke o stanju i prihvaća prilagođene zadane vrijednosti i podatke o registru.

Metoda udaljenog poziva procedure (XML-RPC – XML *Remote Procedure Call*) - Koristi XML za prijenos podataka između programa preko *socket*-a. Pomoću njega UR kontroler može pozvati funkcije (s parametrima) na udaljenom programu/poslužitelju i dobiti natrag podatke. [24]

U ovom radu korišten je tip *Socket* komunikacije.



Slika 9. Komunikacijska sučelja UR robota [25]

5.2. *Socket* komunikacija

5.2.1. Vrste Socket komunikacije

Socket omogućuje komunikaciju između dva različita procesa na istom ili različitim uređajima ili računalima. Preciznije, to je način da se razgovara s drugim računalima pomoću standardnih Unix deskriptora datoteka. *Unix socket* se koristi u okviru klijent-poslužitelj aplikacije gdje je poslužitelj proces koji izvršava neke funkcije na zahtjev klijenta.

Korisnicima su na raspolaganju četiri vrste *socket*-a. Pretpostavlja se da procesi komuniciraju samo između *socket*-a iste vrste, ali ne postoji ograničenje koje sprječava komunikaciju između *socket*-a različitih vrsta. Vrste su:

Stream Sockets – Isporuka podataka u umreženom okruženju je zajamčena. Ako se kroz *stream socket* pošalju tri stavke "A, B, C", one će stići tim istim redoslijedom − "A, B, C". Ovaj *socket* koristi TCP (*Transmission Control Protocol*) za prijenos podataka. Ako je slanje podataka nemoguće, pošiljatelj dobiva indikator pogreške. Zapisi podataka nemaju granica.

Datagram Sockets – Isporuka podataka u umreženom okruženju nije zajamčena. Klijent i server nisu povezani te nije potrebno imati otvorenu vezu kao kod *Stream Sockets*-a. Paket koji se šalje mora imati informacije o odredištu. Ovaj *socket* koristi UDP (*User Datagram Protocol*).

Postoje još *Raw Sockets* i *Sequenced Packet Sockets*, no oni se rijetko koriste. [26]

5.2.2. TCP/IP protokol

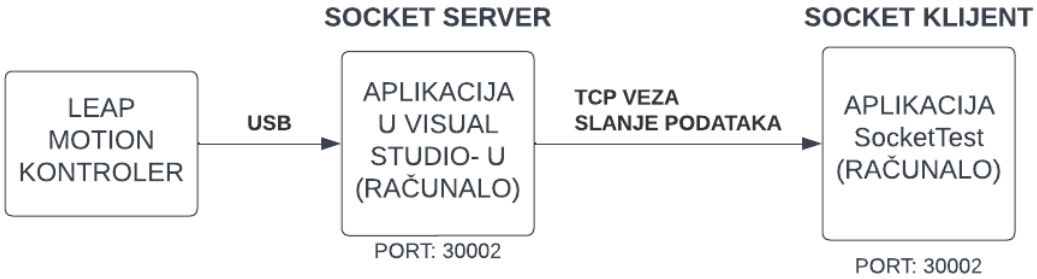
TCP/IP (*Transmission Control Protocol/Internet Protocol*) je skup komunikacijskih protokola koji se koriste za međusobno povezivanje mrežnih uređaja na internetu. TCP/IP se također koristi kao komunikacijski protokol u privatnoj računalnoj mreži (intranet ili ekstranet). Cijeli IP paket - skup pravila i procedura - obično se naziva TCP/IP. TCP/IP određuje kako se podaci razmjenjuju preko interneta pružajući end-to-end komunikaciju koja identificira kako ih treba razbiti u dijelove, adresirati, prenijeti, usmjeriti i primiti na odredištu.

TCP definira kako aplikacije mogu stvoriti kanale komunikacije preko mreže. Također upravlja načinom na koji se poruka sastavlja u manje pakete prije nego što se prenesu preko interneta te se onda ponovno sastave pravim redoslijedom na odredišnoj adresi.

IP definira kako adresirati i usmjeravati svaki paket kako bi bio siguran da će doći do pravog odredišta. [27]

5.2.3. Uspostavljanje Socket veze i korištenje SocketTest aplikacije

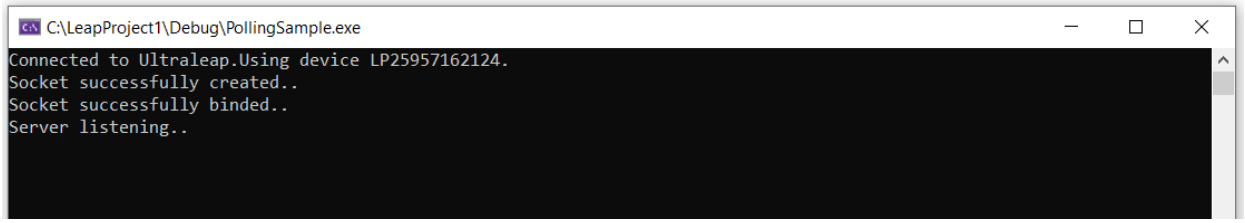
U ovom radu korišten je *Stream Socket* putem TCP/IP protokola. Kako bi se uspostavila veza potrebno je definirati servera i klijenta. Za početak, kako bi se potvrdilo da kod napisan u C jeziku šalje točne podatke koji su potrebni, potrebno je preuzeti i instalirati aplikaciju *SocketTest v 3.0.0*. Ova aplikacija će poslužiti za testiranje *socket* veze te će imati ulogu klijenta, dok će kod u *Visual Studio*-u biti server. [28]



Slika 10. Blok dijagram prikaz komunikacije

Za uspostavljanje *socket* veze postoji gotov kod koji se može preuzeti s interneta [29]. Pošto je korišteno prijenosno računalo s Windows operativnim sustavom potrebno je koristiti prikladnu *socket* vezu. Također, kako bi se *socket* veza uspostavila potrebno je da su server i klijent na istom PORT-u što se definira u kodu.

Ako pokrenemo kod na ekranu će ispisati da je *Leap Motion* Kontroler uspješno spojen, *socket* kreiran i spojen te čeka da se klijent spoji.

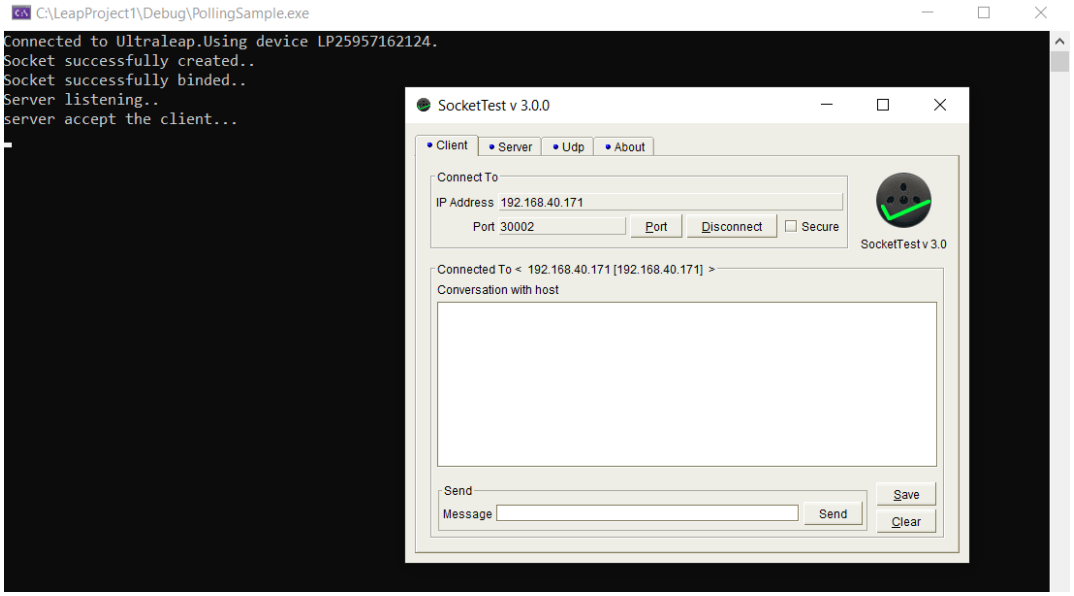


Slika 11. Prikaz ekrana nakon pokretanja koda

5.2.3.1. Pokretanje SocketTest aplikacije

Kako bi se testirao kod potrebno je pokrenuti SocketTest aplikaciju te u izborniku *Client* upisati IP adresu računala i PORT te zatim kliknuti na gumb *Connect.*

Na ekranu koda prikazat će se poruka da se klijent uspješno spojio na server.



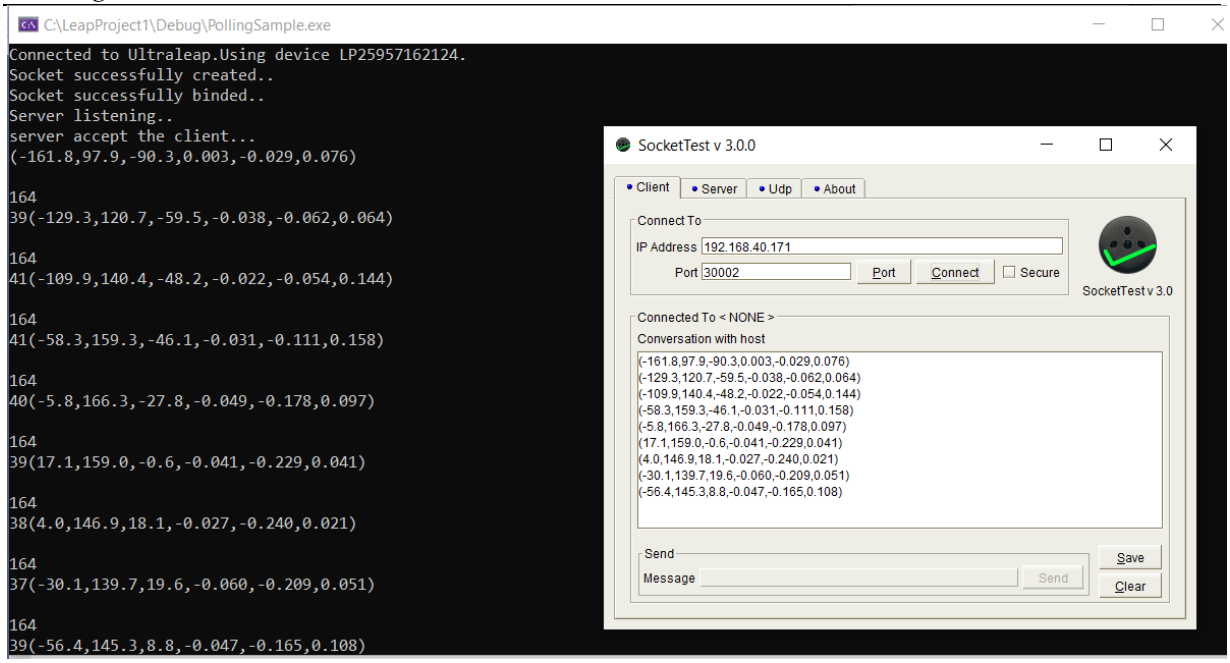
Slika 12. Prikaz ekrana nakon spajanja servera i klijenta

5.2.3.2. Pisanje koda za prijenos podataka između servera i klijenta

Slijedi pisanje funkcije unutar koje će se nalaziti već prije napisan kod za dohvaćanje koordinata i kutova rotacija dlana ruke s *Leap Motion* Kontrolera te dio kod koji sprema te podatke u međuspremnik (*buffer)* te ih zatim šalje klijentu. Međuspremnik je generalni izraz te će ovdje poslužiti kao mjesto za privremeno pohranjivanje podataka. Služi za ublažavanje razlika između ulazne i izlazne brzine. Koordinate X,Y,Z slat će se s jednom decimalom te kutevi rotacija tih osi s tri decimale pošto je Kontroler vrlo precizan te šalje podatke sa šest decimala, a robot je osjetljiv na svaku decimalu i svaku promjenu iznosa prihvaća kao novu točku u koju se treba pomaknuti.

```
// Function designed for chat between client and server.
void loopFunction(SOCKET sockfd)
{
   char buff[MAX];

   for (;;) {

            ...

           // copy message in the buffer
           sprintf(buff, "(%0.1f,%0.1f,%0.1f,%0.3f,%0.3f,%0.3f)\n",
              hand->palm.position.x,
              hand->palm.position.y,
              hand->palm.position.z
              hand->palm.orientation.x,
              hand->palm.orientation.y,
              hand->palm.orientation.z);

           printf(buff);
           printf("\n%i", sockfd);
           printf("\n%d", strlen(buff));

           // and send that buffer to client
           send(sockfd, buff, strlen(buff), 0);
        }
```

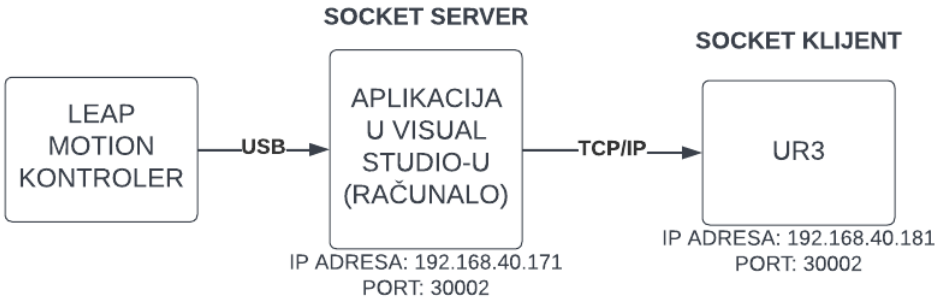
Ako se pokrene kod te ako se rukom pređe iznad *Leap Motion* Kontrolera, na ekranu će se ispisivati koordinate te će se odmah slati klijentu što se može vidjeti na *SocketTest* aplikaciji. Ako sve radi kako treba znači da je veza uspješno uspostavljena te je sljedeći korak slanje tih podataka robotu.



Slika 13. Prikaz ekrana nakon spajanja servera i klijenta i slanja podataka

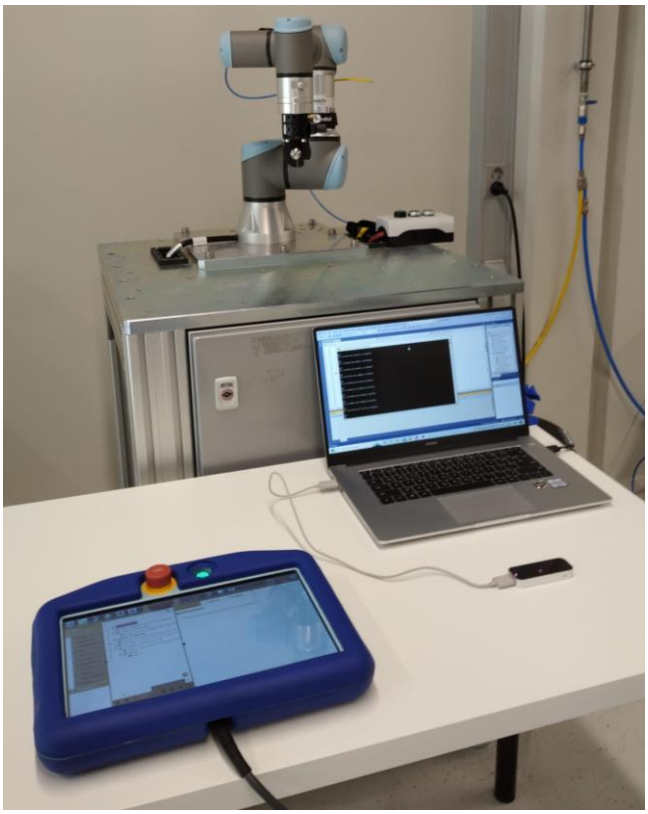
5.3. Spajanje računala na UR robota

Kako bi se podaci slali na robota potrebno je računali i robota spojiti na istu mrežu. U CRTI robot je već spojen na mrežu fakulteta putem Ethernet priključka te je potrebno računalo također spojiti na istu mrežu. Pošto prijenosno računalo nema utor za Ethernet kabel može se koristiti nastavak za USB priključak. Kada su računalo i robot spojeni na istu mrežu, IP adresa računala i IP adresa robota razlikuju se samo u posljednje tri znamenke.



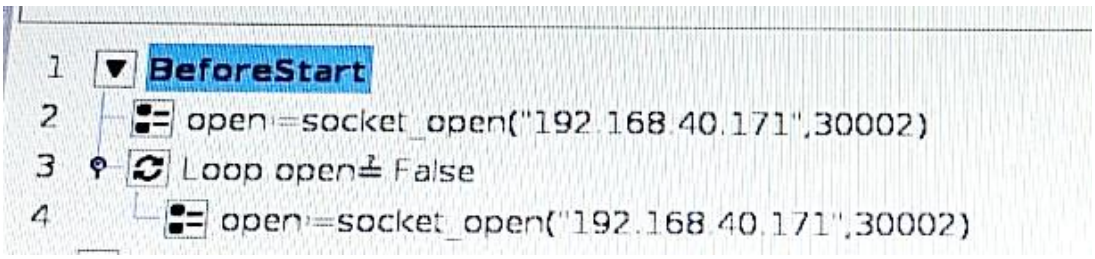
Slika 14. Blok dijagram prikaz komunikacije

Na slici 15. prikazani su svi potrebni uređaji i priključci koji su se koristili za ovaj rad.



Slika 15. Korišteni uređaji za rad

U korisničkom sučelju robota potrebno je upisati kod koji će služiti za povezivanje na računalo putem *socket* komunikacije. Na vrhu stranice potrebno je odabrati gumb *New* kako bi se otvorila nova *URScript* skripta. U skripti je s desne strane izbornik u kojem je potrebno označiti kućicu *BeforeStart* te se otvara nova naredba koja će pri pokretanju koda uvijek započinjati prije glavnog programa. U lijevom izborniku pod *Advanced* potrebno je odabrati *Assignment* te upisati naredbu *socket_open* koja omogućava povezivanje na *socket* komunikaciju. Potrebno je upisati IP adresu računala na koje se potrebno spojiti te PORT koji je već prije naveden u kodu. Kako bi komunikacija ostala otvorena potrebno je koristiti *Loop* funkciju. [30]



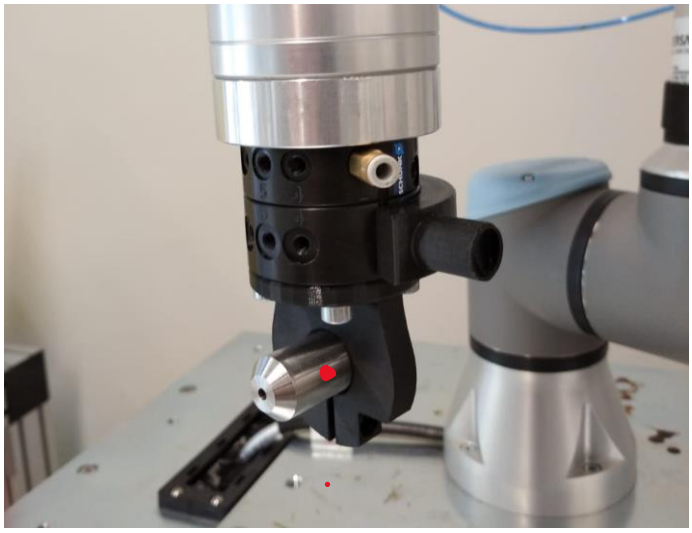
Slika 16. *PolyScope* kod za spajanje putem *socket* komunikacije

Ako pokrenemo ovu skriptu nakon što smo pokrenuli kod u *Visual Studio*-u pojavit će se isti ekran s porukom da se klijent uspješno spojio na server kao u slici 10.

5.4. Slanje koordinata robotu

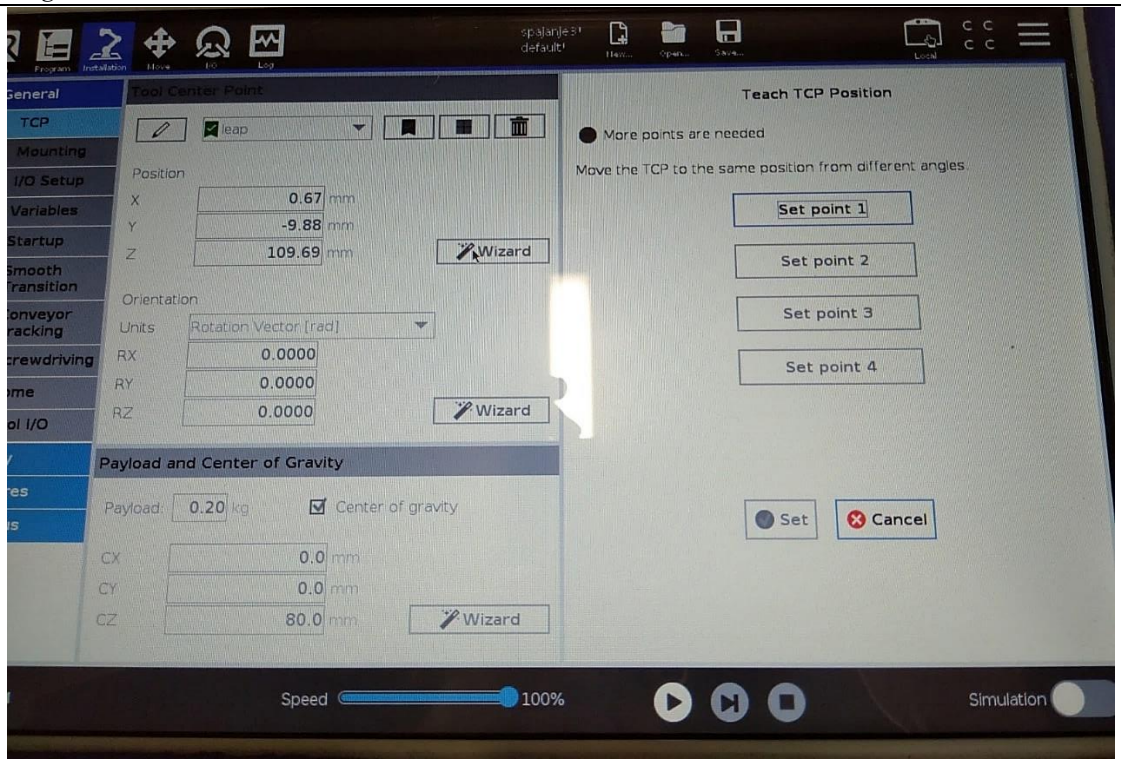
5.4.1. Pozicioniranje vrha alata

Robot će pratiti dobivene koordinate vrhom alata te je zato potrebno definirati udaljenost vrha alata od prirubnice. Na robot je već prikvačen alat te će se vrh alata definirati u središtu sivog valjka označenom crvenom točkom na slici 13.



Slika 17. Vrh alata na robotu

Kako bi se definirao vrh alata potrebno je pritisnuti gumb *Installation* te u lijevom izborniku pod *General* izabrati *TCP* (engl. Tool Control Point – kontrolna točka alata) te pritisnuti gumb *Wizard*. Sljedeći korak je definiranje tri točke pomicanjem robota kako bi se definirao vrh alata.



Slika 18. Definiranje vrha alata na robotu

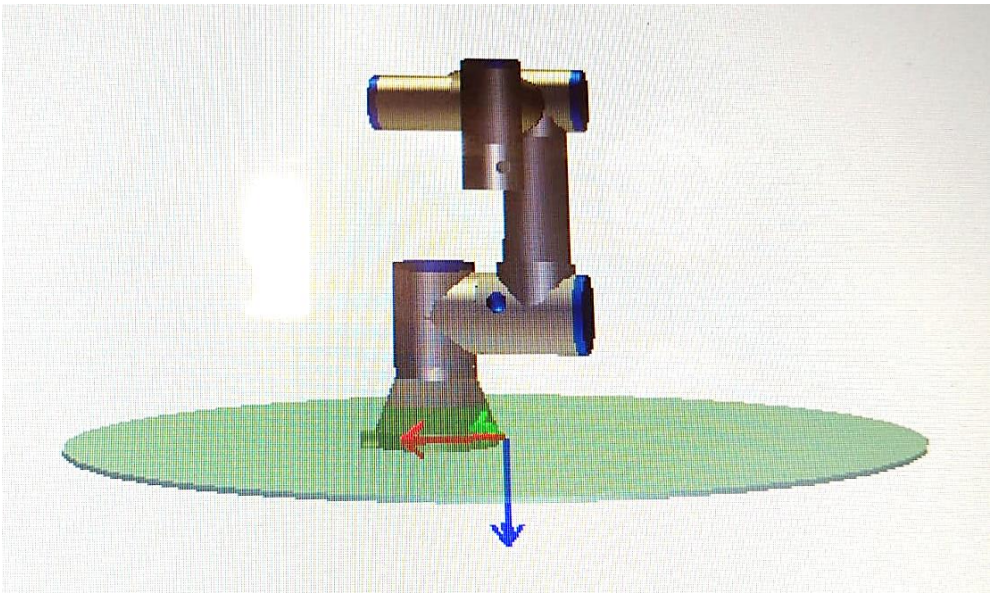
Kako bi se kretnje robota kretale u odnosu na vrh alata potrebno je u naredbi *MoveJ* u opciji *Set TCP* odabrati definirani vrh alata.

5.4.2. Definiranje ravnine na koju će se kretnje robota referencirati

Kako bi se robot pomicao u određenom dijelu svog raspona potrebno je definirati ravninu s ishodištem kako bi se vrh alata mogao pomicati sukladno s tom ravninom.

Potrebno je u *Installation* izborniku pod *Features* kliknuti na gumb *Plane*. Stvara se nova ravnina koju je potrebno definirati. Ravnina se definira s tri točke u prostoru. Praćenjem uputa na korisničkom sučelju te pomicanjem robota u svaku od tri točke dobiva se nova ravnina. Ravnina koja će se koristiti kako referenca za ovaj rad prikazana je na slici 16. Može se vidjeti da je X os (označena crvenom bojom) usmjerena ulijevo, a Y os (označena zelenom bojom) prema bazi robota te da Z os (označena plavom bojom) gleda u stol. U kodu napisanom u *URScript*-i prema ovim osima će se prilagoditi koordinate koje se dobivaju s *Leap Motion* Kontrolera.

Kako bi se kretnje robota referencirale na ovu ravninu potrebno je u naredbi *MoveJ* u opciji *Feature* odabrati definiranu ravninu.



Slika 19. Definirana nova ravnina

5.4.3. Pisanje koda u URScript

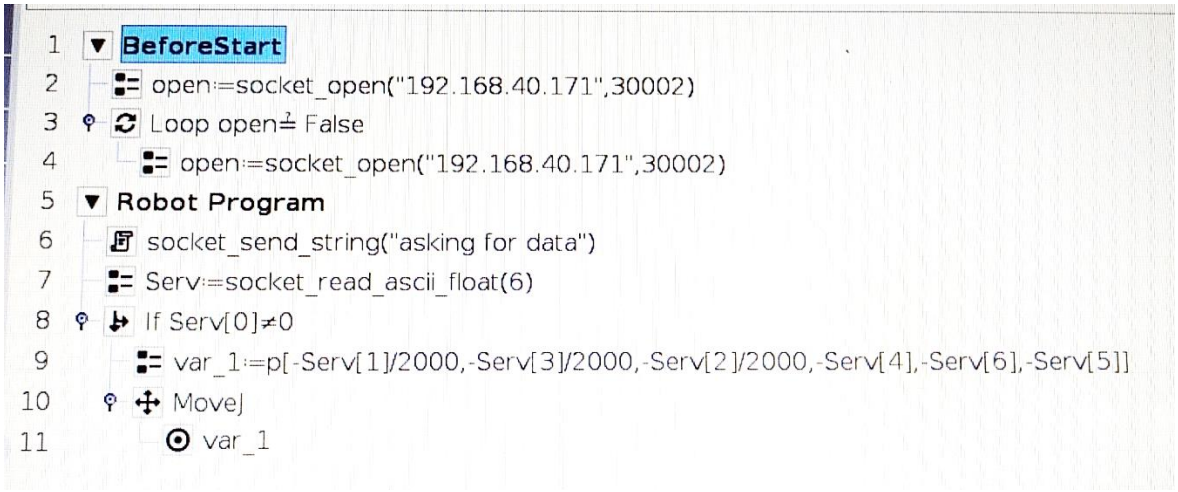
U skripti je potrebno raspisati kod koji će primati koordinate te pomicati robota sukladno s njima. Naredbom *socket_send_string* serveru se šalje poruka te služi za provjeru ako je moguće da server prima poruke od klijenta. Naredbom *socket_read_ascii_float()* robot će primati koordinate s računala. U zagradu je potrebno upisati broj primljenih podataka koji je u ovom slučaju 6: X,Y i Z os te RX,RY i RY kutovi rotacija. U varijablu *var_1* potrebno je upisati *p[]* što označava poziciju u prostoru. Unutar *p[]* upisuje se 6 znakova koji označavaju redom X,Y,Z koordinate te RX,RY,RZ kutovi nagiba. Pošto *Leap Motion* Kontroler ima različit koordinatni sustav od koordinatnog sustava robota potrebno je zamijeniti Z i Y os, te ispred X, Y i Z osi te ispred RX,RY i RZ kutova rotacija staviti minus kako bi se robot gibao u istom smjeru kao i ruka iznad Kontrolera.

Vrijednosti X,Y i Z koordinata potrebno je podijeliti s 1000 jer su vrijednosti koje Kontroler snima u [mm], a vrijednosti koje robot prati u [m]. Zatim, potrebno je smanjiti raspon kretanja robota (500 mm od baze robota) jer je manji od raspona vidokruga Kontrolera (600 mm od površine Kontrolera). Te kako bi osigurali da se robot neće sudariti u podlogu ili u samoga sebe

te kako ne bi došlo do zaustavljanja robota jer su dobivene koordinate izvan raspona kretanja robota, vrijednosti X,Y i Z podijeljene su još i s 2.

Kutovi RX,RY i RZ koje Kontroler šalje su u radijanima te ih robot također čita u radijanima. Naime, kod ponavljanja zakreta ruke, robot često zapne, te je radi sigurnosti namješteno da kad se ruka zakrene za 90°, da se robot zakrene za duplo manji kut, odnosno za 45°.

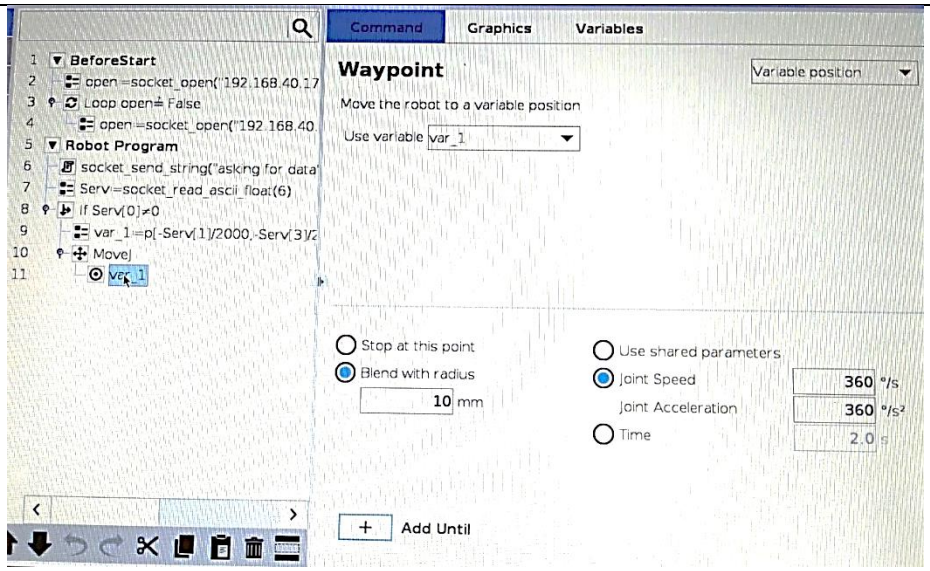
Naredbom *MoveJ* robot se pomiče u točku definiranu u varijabli *var_1*.



Slika 20. Kod napisan na korisničkom sučelju robota

Kako bi se postiglo fluidno kretanje robota potrebno je u varijabli *var_1* promijeniti postavke. U izborniku *Command* potrebno je umjesto *Stop at this point* označiti *Blend with radius* te upisati 10 [mm]. Time se postiže da umjesto da nakon svake točke robot stane pa krene u sljedeću točku, zapravo svaku točku zaobilazi u polumjeru od 10 mm te tako nastavlja kretnju bez zaustavljanja. Iznos od 10 mm dobiven je eksperimentalno, pokušavajući s iznosima između 5 i 50 mm.

Kako bi robot što uspješnije pratio dobivene koordinate potrebno je namjestiti brzinu kretanja na maksimum. Zato je odabrana opcija Joint Speed te je samostalno upisana maksimalna vrijednost od 360 °/s te opcija Joint Acceleration te je upisana maksimalna vrijednost od 360 °/s. [31]



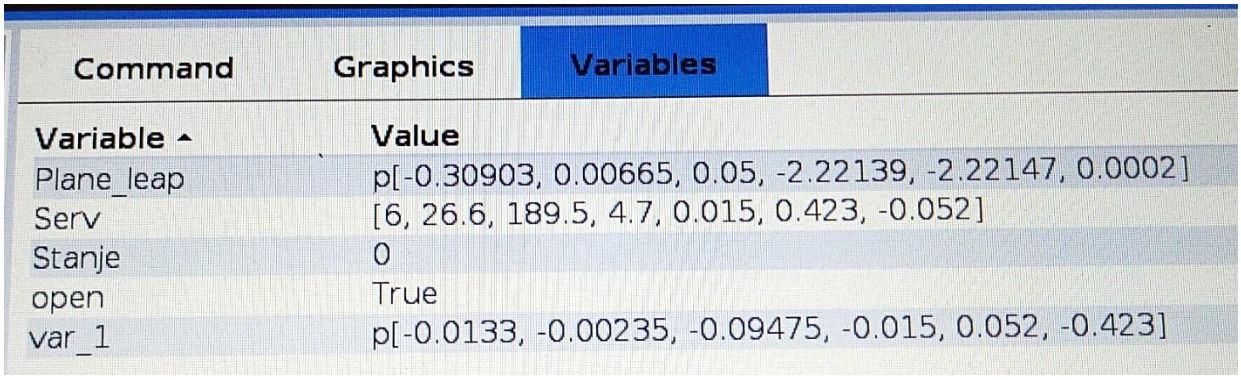
Slika 21. Postavke varijable *var_1*

5.5. Pokretanje robota

Potrebno je pokrenuti kod u C jeziku u Visual Studio-u, zatim pokrenuti kod na korisničkom sučelju robota. Nakon što se ispiše poruka da su server – računalo i klijent – robot povezani, pokrete rukom iznad *Leap Motion* Kontrolera robotska ruka ponavlja i pomiče se u prostoru.

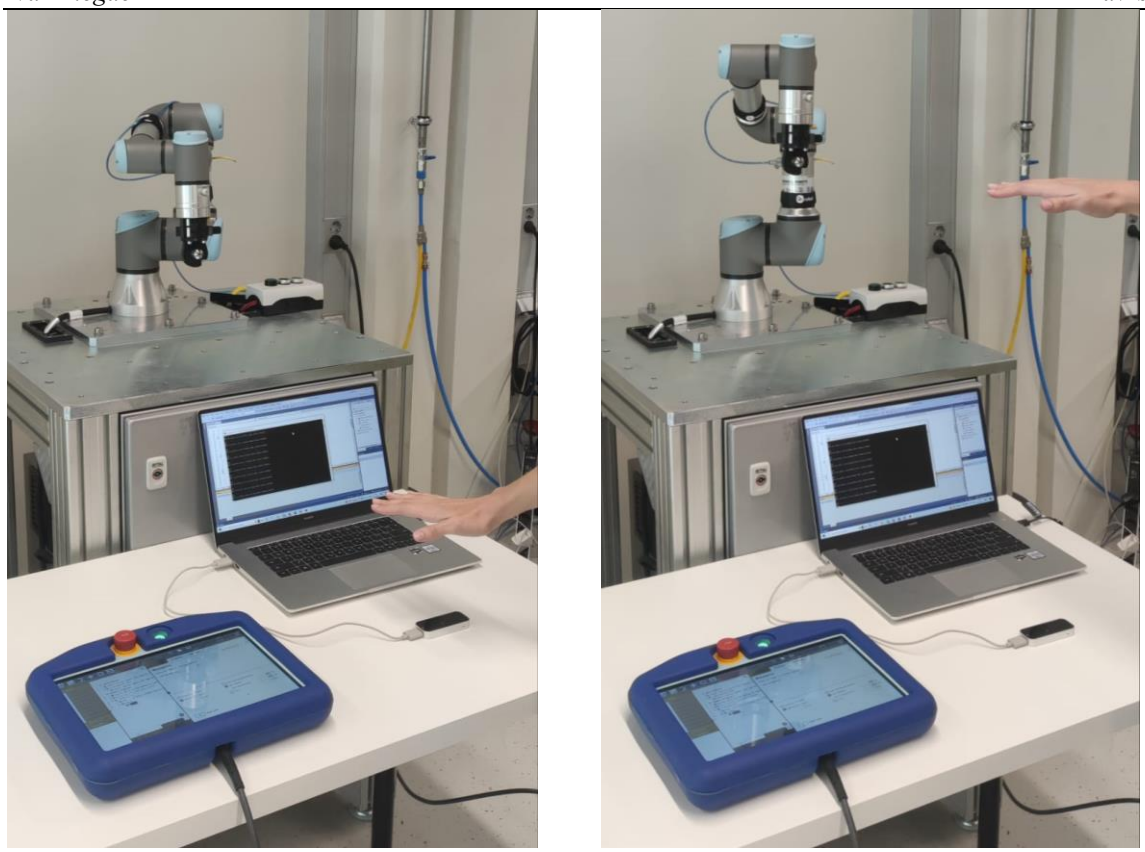
Koordinate u [mm] koje računalo šalje robotu, varijabla *var_1* preračunava te robotu šalje koordinate u [m] što se može vidjeti pod izbornikom *Variables* na korisničkom sučelju robota.

Varijabla *Serv* ima sedam podatka od kojih prvi podatak označava broj dobivenih koordinata te su ostalih šest podatka dobivene koordinate i kutovi rotacija s *Leap Motion* Kontrolera.

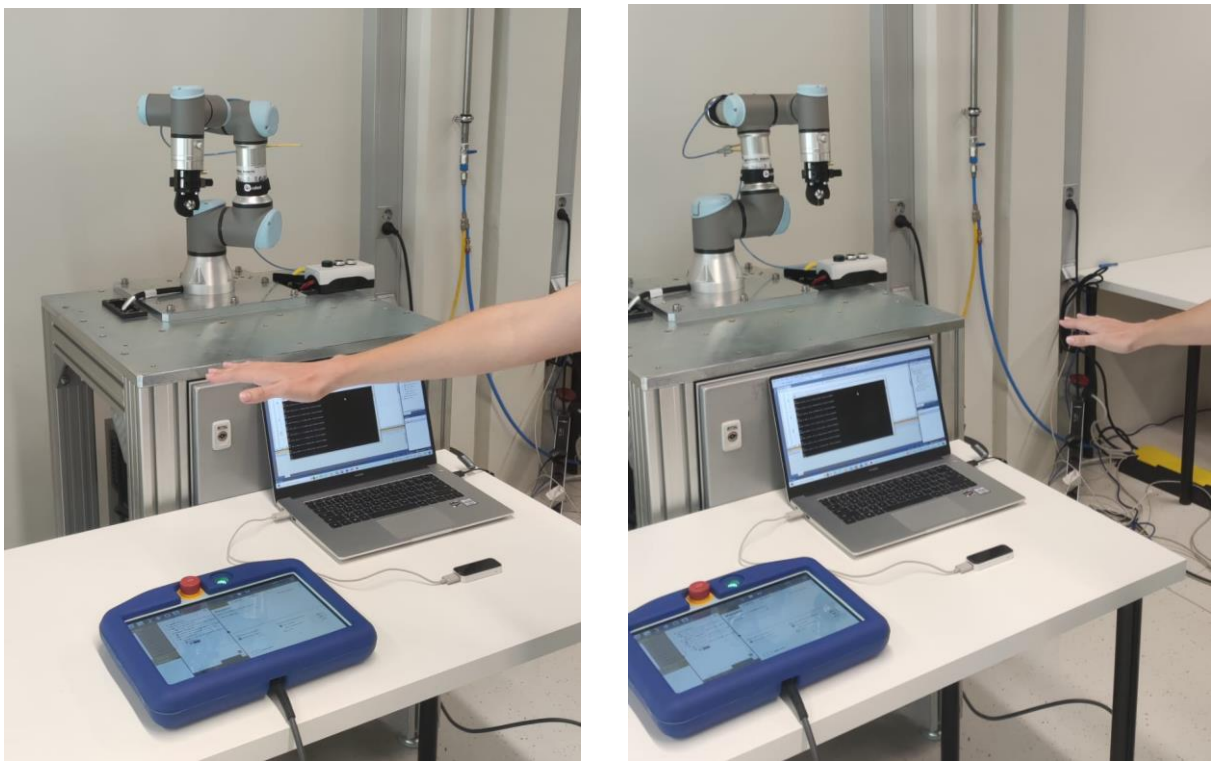


Slika 22. Varijable na korisničkom sučelju robota

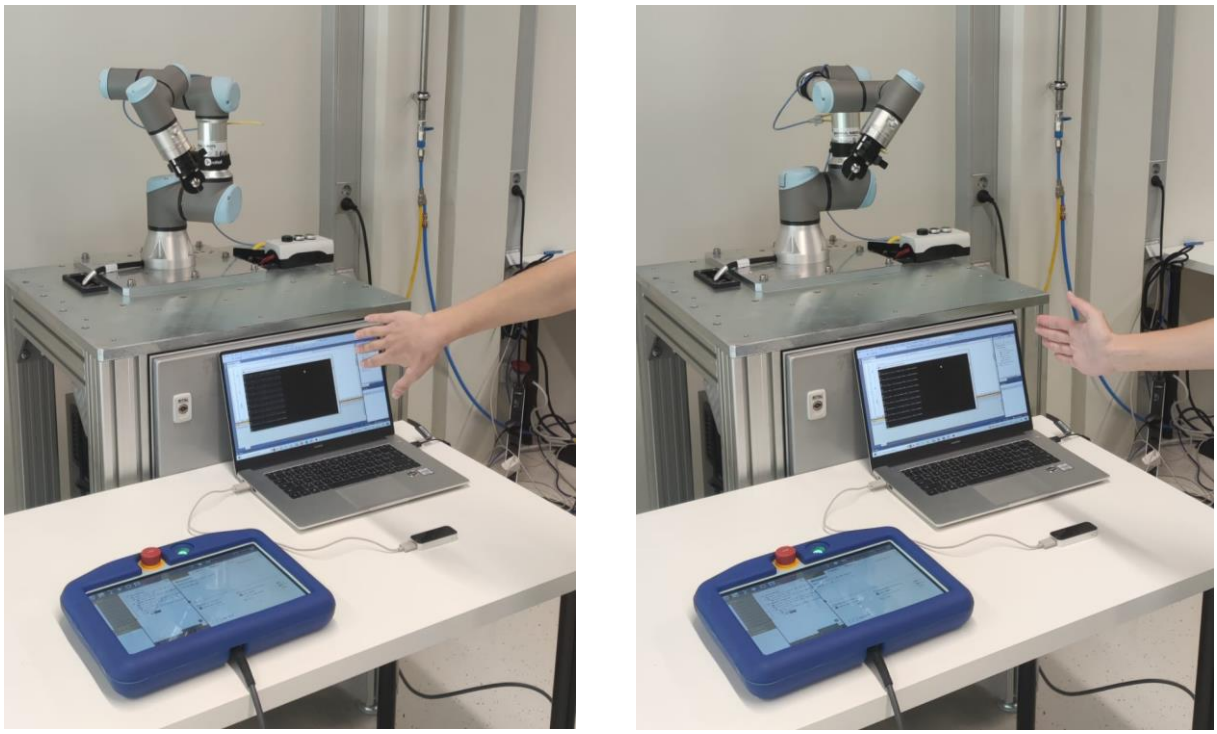
Slike 18., 19., 20., 21., 22., 23., 24., i 25. prikazuju pomicanje robota sukladno pomicanju ruke.



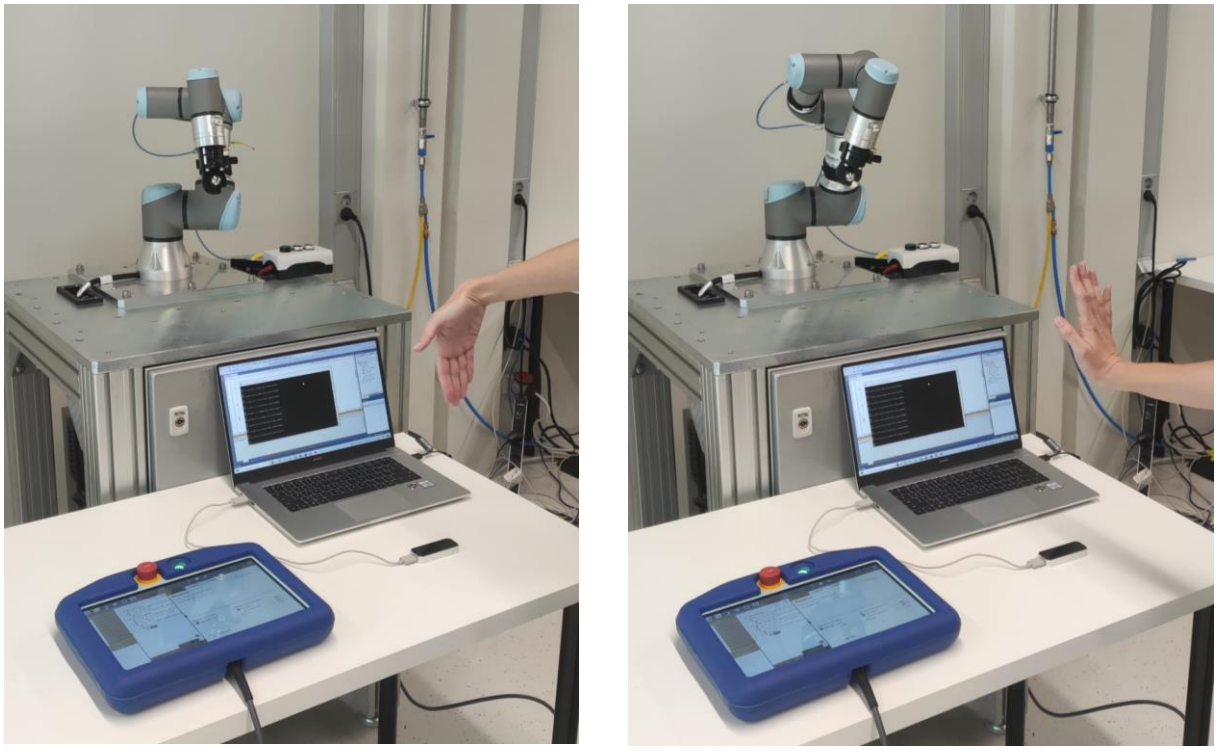
Slike 23. Pomicanje ruke i robota prema dolje (lijeva slika) i gore (desna slika)



Slike 24. Pomicanje ruke i robota ulijevo (lijeva slika) i udesno (desna slika)



Slike 25. Zakretanje ruke i robota oko Z osi *Leap Motion* Kontrolera



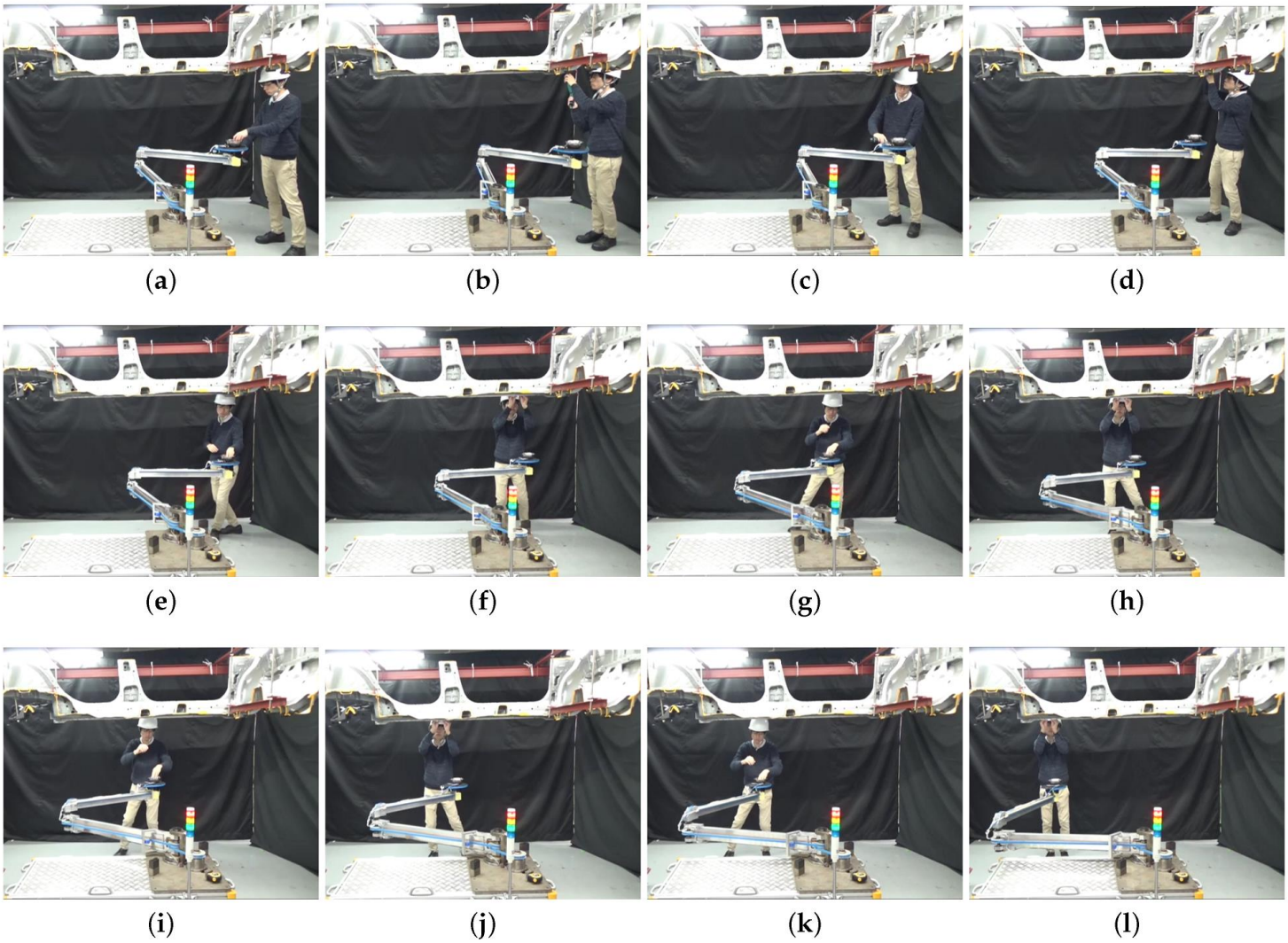
Slike 26. Zakretanje ruke i robota oko X osi *Leap Motion* Kontrolera

6. PRIMJENA VIZIJSKOG SUSTAVA ZA VOĐENJE ROBOTA

Kontroliranje robota korištenjem senzora Leap Motion moglo bi prvenstveno imati svrhu u područjima do kojih ljudi ne mogu doći. Stoga, robotska ruka izvršava zadatke koji su kontrolirani korištenjem Leap Motion Kontrolera. To omogućava izvođenje zadataka s prirodnim ljudskim pokretima kako bi se postigli željeni rezultati te služi kao alternativni način za obavljanje radnji. Robotska ruka mogla bi se koristiti za izvršavanje zadataka u područjima kao što su nuklearno pogođena područja, u dubokom moru, u područjima visokog tlaka i vlage, u ekstremno neprijateljskoj klimi i mjestima kao što su svemirske postaje itd. [32]

Na Tohoku sveučilištu u Japanu 2021. godine provedeno je istraživanje o kolaborativnom robotu koji pomaže radniku u obavljanju određene radnje. Predložena je shema u kojoj se prate kretnje čovjeka te se prema tome planiraju i upravljaju kretnje kolaborativnog robota koji opskrbljuje radnika potrebnim dijelovima i alatima u procesu sklapanja u tvornici. U predloženoj shemi koristi se 3D senzorski sustav za mjerenje kosturnih podataka radnika. U svakom trenutku snimanja senzorima procjenjuje se optimalna pozicija isporuke. Istovremeno se predviđaju buduće moguće pozicije radnika. Model prediktivne kontrole (engl. Model Predictive Control – MPC) korišten je za izračun putanje robota i predviđanje budućih položaja radnika kako bi se osigurao glatki prijelaz između prethodne i nove putanje.. Instalirali su predloženu shemu u kolaborativni robotski sustav s planarnim manipulatorom s dva DOF-a. Eksperimenti su provedeni u stvarnom okruženju gdje je radnik uz pomoć robota izvodio proces sklapanja automobila. Rezultati pokusa potvrdili su predloženu shemu koja pruža bolju pomoć radniku, poboljšava radnu učinkovitost te osigurava sigurnost i udobnost radnika. Pretpostavili su da je ljudski radnik u tvornici dobro obučen i da je dinamika njegovog/njezinog ponašanja prema kretanju robota zanemariva. [33]

Praćenje pokreta ljudi te sposobnost robota da pruži pomoć u bilo kojem trenutku ima veliki potencijal i vrijedno je provođenja daljnjih istraživanja.



Slike 27. Eksperiment koji prikazuje potpuni radni ciklus u kojem se izvodi šest zadataka.

(a) Zasun i alat su podignuti; (b) Zadatak 1 je izvršen; (c) Alat se vraća i uzimaju se 3 ušice; (d) Zadatak 2 je izvršen; (e) čahura je podignuta; (f) Zadatak 3 je izvršen; (g) Podignut je prsten; (h) Zadatak 4 je obavljen; (i) Podignut je prsten; (j) Zadatak 5 je izvršen; (k) Podignut je prsten; (l) Izvršava se zadatak 6.

7. ZAKLJUČAK

Tehnologija se svojim napretkom sve više usmjeruje na integraciju strojeva i računala u život čovjeka te u virtualnu stvarnost. Integracija se odnosi na profesionalni aspekt čovjekova života, ali i na svakidašnji aspekt. Skoro svaki posao koji čovjek obavlja se pokušava digitalizirati. Virtualna stvarnost je pojam koji se sve više širi u svim industrijama. Virtualna stvarnost omogućava prisutnost s daljine, kontrolu, simulaciju stvarnosti… što može doprinijeti povećanju produktivnosti, smanjenju troškova, unaprjeđenje analize te poboljšanju zabave.

Moderne tehnologije i digitalni alati omogućavaju snimanje pokreta lakim i vrlo korisnim. Virtualna stvarnost glavna je tehnologija koja prati, snima i analizira pokrete. Integracija virtualne stvarnosti u ostale tehnologije mogla bi puno doprinijeti olakšavanju procesa i smanjenju ljudskog rada što na kraju teži postizanju veće produktivnosti i profita.

Leap Motion Kontroler je malen, brz i precizan uređaj koji služi za snimanje i analiziranje pokreta ruku i prstiju čovjeka te njihovu daljnju upotrebu u virtualnoj stvarnosti. Zbog malih dimenzija i niske cijene široko je upotrebljiv u raznim industrijama. Kontroler se može integrirati s VR naočalama te ima i novu funkciju – upravljanje zaslonom bez dodira.

Leap Motion Kontroler koristi napredne algoritme i tehnike filtriranja te pruža detaljne informacije o pokretima ruku i prstiju čovjeka.

U radu je proveden eksperiment korištenjem *Leap Motion* Kontrolera s robotskom rukom *UNIVERSAL ROBOTS 3* te je imao za cilj ispitati praktičnost korištenja alata za snimanje pokreta te praktičnost integracije s robotom. Snimanje pokreta ruku te robotom ponavljani pokreti pokazali su se uspješnim te vrlo zanimljivim područjem uporabe. Ovakva upotreba snimanja i prijenosa pokreta mogla bi biti od značajne vrijednosti u radu gdje je potrebna fizička udaljenost za upravljanje strojevima ili pogonom. Kad bi se poslovi koji zahtijevaju fizičku prisutnost za upravljanje strojevima mogli obavljati na daljinu, to bi uvelike olakšalo i ubrzalo obavljanje posla.

Povezivanje *Leap Motion* Kontrolera s robotom pokazalo se uspješno, no također ima puno mjesta za napredak. Vrijeme potrebno robotu da primi koordinate i pomakne se je ipak nedovoljno brzo te se pri bržim pomicanjem ruke iznad Kontrolera, robot ne stigne dovoljno brzo pomaknuti. Ovo bi se moglo poboljšati korištenjem neke druge vrste prijenosa podataka između računala i robota, npr. RTDE – razmjenom podataka u stvarnom vremenu ili nadograđivanjem koda i korištenjem različitih funkcija koje nudi *URScript UR3* robota.

Također, u ovom radu nije se obraćala pažnja na preciznost, a dokazano je da je Kontroler vrlo precizan što bi moglo biti od velikog značaja za provođenje daljnjih eksperimenata kada bi se koordinatni sustav robota precizno podesio u odnosu na koordinatni sustav Kontrolera.

Nadalje, zanimljiva je nova mogućnost postavljanja Kontrolera – s vrha zaslona kada je okrenut prema dolje (engl. *screentop*). Korisno bi bilo isprobati ovu mogućnost jer omogućava šire vidno polje koje bi moglo obuhvatiti predmete koji su odloženi na radnoj površini te bi se mogla isprobati mogućnost da robot prati kretnje ruke koje barataju različitim predmetima te ih hvataju i odlažu (engl. *Pick and place*).

Ovaj eksperiment mogao bi se dodatno nadograditi VR naočalama jer bi tada korisnik imao više opcija te bi se naredbe mogle jasnije slati robotu. Ovakav sustav bi na kraju mogao pomoći ljudima da nadziru robote iz daljine što u konačnici vodi prema sve višim stupnjevima automatizacije.

LITERATURA

[1] Šare N., Snimanje i analiza pokreta u virtualnoj stvarnosti [Diplomski rad]. Zagreb: Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2020. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:270575

[2] https://learn.sparkfun.com/tutorials/leap-motion-teardown/all

[3] https://www.ultraleap.com/company/news/blog/how-hand-tracking-works/

[4] https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf

[5] https://blog.leapmotion.com/leapuvc/

[6] https://developer-archive.leapmotion.com/documentation/v2/csharp/devguide/Leap_Overview.html

[7] https://www.ultraleap.com/tracking/#how-it-works

[8] https://developer-archive.leapmotion.com/documentation/cpp/devguide/Leap_Architecture.html

[9] https://docs.ultraleap.com/tracking-api/leapc-guide/using-leapc.html

[10] https://hr.wikipedia.org/wiki/Pribor_za_razvijanje_softvera

[11] https://hr.wikipedia.org/wiki/API

[12] https://blog.leapmotion.com/vr-universities/

[13] https://www.universal-robots.com/products/ur3-robot/

[14] https://wiredworkers.io/product/ur3e/

[15] https://s3-eu-west-1.amazonaws.com/ur-support-site/43872/UR3e_User_Manual_en_Global.pdf

[16] https://erepo.uef.fi/bitstream/handle/123456789/27164/urn_nbn_fi_uef-20220213.pdf?sequence=1&isAllowed=y

[17] https://s3euwest1.amazonaws.com/ursupportsite/41166/UR3e_User_Manual_en_Global.pdf

[18] https://developer-archive.leapmotion.com/documentation/python/devguide/Leap_SDK_Overview.html

[19] https://support.leapmotion.com/hc/en-us/articles/360004362237-Generating-a-Python-3-3-0-Wrapper-with-SWIG-2-0-9

[20] https://docs.ultraleap.com/tracking-api/examples/polling-example.html

[21] https://en.wikipedia.org/wiki/CMake

[22] https://en.wikipedia.org/wiki/Visual_Studio

[23] https://docs.ultraleap.com/tracking-api/group/group___structs.html

[24] Šuligoj, Filip, Industrijske računalne mreže, Računalo – robot TCP/IP socket komunikacija, prezentacija, 2021.

[25] https://www.universal-robots.com/articles/ur/interface-communication/overview-of-client-interfaces/

[26] https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm

[27] https://www.techtarget.com/searchnetworking/definition/TCP-IP

[28] https://sourceforge.net/projects/sockettest/

[29] https://www.binarytides.com/winsock-socket-programming-tutorial/

[30] UNIVERSAL ROBOTS, The URScript Programming Language, Verzija 3.5.4, Travanj 12, 2018

[31] https://www.youtube.com/watch?v=lwzcp-sM4qk&ab_channel=CrossCompany

[32] https://www.ijert.org/research/leap-motion-controlled-robotic-arm-IJERTCONV2IS04015.pdf

[33] https://www.mdpi.com/1424-8220/21/24/8229/htm

PRILOZI

Prilog 1. Projekt koji sadrži programski kod - https://github.com/mwik20/Leap_Project