Virtualni uporabniški vmesnik za krmiljenje nanorobotske celice shaptično napravo

Simon Zapušek
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova ulica 17, 2000 Maribor
E-pošta: simon.zapusek@gmail.com
Mentor: Riko Šafarič

Virtual user interface for control of a nanorobotic cell with a haptic device

The development of man–machine interface for communication between the user and a nanorobot is presented in the paper. The interface allows the control of a real nanorobot cell with a haptic device, control with the programmed buttons of the interface and control of point to point movements. The communication is done on the basis of the UDP protocol. The user interface has been developed in C++ and VRML languages.

1 Uvod

Veliko se govori o nanotehnologiji, ki je danes zelo aktualna tema v svetu. Njen razvoj se je močno razmahnil in vanj se vlagajo velika sredstva [3]. Tudi naše delo, ki bo predstavljeno v nadaljevanju predstavlja eno izmed področij nanotehnologije. Naš cilj je izdelati virtualni uporabniški vmesnik, ki bo omogočal kmiljenje nanorobotske celice s haptično napravo. Glede na dejstvo, da pomikov reda nanometra praktično ne vidimo s prostim očesom in si nanosvet le stežka predstavljamo, smo se odločili, da za krmiljenje nanorobota uporabimo napravo, ki nam bo poleg čutila vida, dala še občutek dotika oziroma občutek delovanja na okolico. Takšno napravo imenujemo haptična naprava. V našem primeru smo uporabili haptično napravo Phantom OMNI, proizvajalca SensAble Technologies.

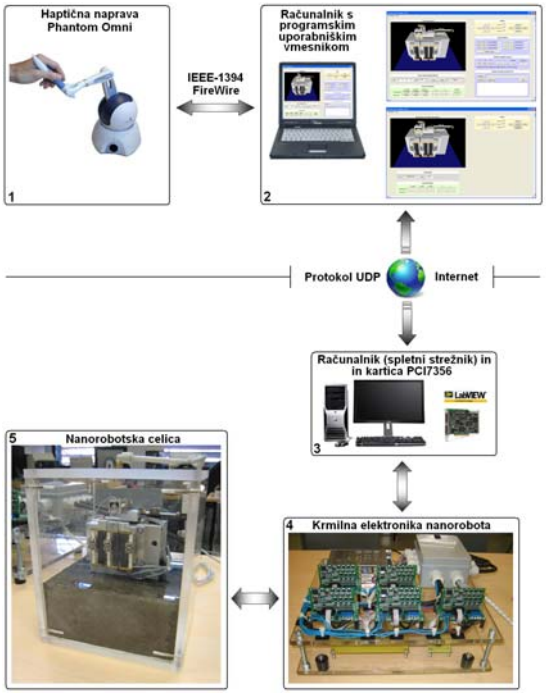
Posamezni sestavni deli realnega nanorobota so občutljivi in imajo visoko ceno. Zato želimo preprečiti kolizije med posameznimi osmi nanorobota in s tem preprečiti njihovo uničenje. To želimo doseči z uporabo haptične naprave, ki omogoča generiranje sile na njenem peresu. Ta sila nam lahko preprečuje premikanje v določeni smeri. Proizvajalec haptične naprave ponuja tudi razvojno orodje OpenHaptics, ki omogoča da razvijemo v programskem jeziku C ali C++ lastni uporabniški vmesnik.

V našem primeru bo potrebno izvesti komunikacijo med haptično napravo in uporabniškim vmesnikom ter uporabniškim vmesnikom in realno nanorobotsko celico. Ker želimo krmiliti nanorobotsko celico na daljavo, se pojavi težava, kako spremljati njeno dejansko stanje (dejanski položaj posameznih osi). Tukaj imamo možnost, da bi dejanski položaj spremljali preko kamere. Težava, ki se tu pojavi je da bi za to potrebovali hiter prenos podatkov, saj lahko drugače prihaja do izgube slike in njenega časovnega zamika. Pri prenosu slike s kamere vemo, da je lahko že samo ena slika velikosti megabajta, prikazovati pa jih je potrebno vsaj nekje 25 na sekundo. V ta namen želimo izdelati uporabniški vmesnik, ki bo omogočal prikaz dejanskega položaja, kar na virtualnem modelu nanorobota. Ta model mora čim bolj posnemati realnega nanorobota. Prednost prikazovanja položaja na virtualnem modelu pa je, da za prenašanje informacij oziroma podatkov z realnega nanorobota ne potrebujemo hitre povezave, saj so lahko podatki velikosti le nekaj bajtov.

2 Opis sistema

Celoten sistem lahko razdelimo na dva nivoja (slika 2.1). Spodnji nivo predstavljajo nanorobotska celica (5), krmilna elektronika nanorobota (4) in računalnik z vgrajeno vhodno/izhodno kartico PCI7356 (3). Zgornji nivo pa predstavlja računalnik, na katerem je nameščen uporabniški vmesnik (2), ki omogoča kumunikacijo s haptično napravo (1) in spodnjim nivojem (nanorobotsko celico).

Nanorobotska celica (5) je zgrajena tako, da vsebuje pet linearnih motorjev s petimi pripadajočimi merilnimi sistemi. Linearni motorji so pritrjeni na kovinsko konstrukcijo z vodilom za pomik v smeri osi Y. Prva dva motorja služita kot aktuatorja za pomikanje po X in Y osi, ostali trije motorji pa služijo kot aktuatorji za pomikanje podajalnih miz po Z osi. Podajalne mize, ki se pomikajo po osi Z smo v nadaljevanju poimenovali s kraticami PM1, PM2 in PM3. Nanorobot vsebuje tudi prijemalo, ki omogoča manipulacijo predmetov v smeri X in Y osi. Motorji nanorobota so linearni piezoelektrični motorji PizoLEGS proizvajalca PizoMotor Upssala AB, ki so se na področju mikro in nano robotike dobro uveljavili. Odlikuje jih dobra dinamika, zmožnost ustvarjanja majhnih pomikov in premagovanja relativno velikih sil. Kot slabost pa velja omeniti histerezo piezoelektrika, ki povzroča težave pri pozicioniranju. Linearni piezoelektrični motor proizvede linearno gibanje z ustreznim prekrmiljenjem tako imenovanih piezo nog. Piezo nogo sestavljata dva piezoelektrična aktuatorja, spojena eden na drugega, ki v odvisnosti od napajalne napetosti proizvedeta romboidno gibanje vrha teh nog.



Slika 2.1: Sistem za krmiljenje nanorobotske celice s haptično napravo

Motorji omogočajo pomike v korakih od 2nm do 8µm, s hitrostjo do 12 mm/s. Za merjenje položaja je uporabljen linearni enkoder proizvajalca NANOS Instruments, ki omogoča nanometersko resolucijo. Deluje po elektromagnetnem principu, z njim pa je možno doseči resolucijo do 61 nm, s predpisanim dovoljenim odstopanjem +/- 0,15%. Posebno prijemalo, ki je vgrajeno na nanorobotu je zgrajeno iz mikrostrukturiranega stekla debeline 1 mm in izdelano na osnovi UV liftografskega postopka jedkanja. Prijemalo so razvili na inštitutu Fraunhofer Institute of Reliability and Microintegration v Nemčiji. Osnovna oblika prijemala je omogočala pomik čeljusti približno +/-1 µm. Z naknadno spremembo osnovne oblike samega prijemala in drugačno namestitvijo piezoelektričnega aktuatorja pa je bil dosežen pomik čeljusti približno +/-10 µm. Tako sedaj prijemalo omogoča prijemanje objektov velikosti 200 µm.

Krmilna elektronika nanorobota (4) služi za krmiljenje posameznih osi oziroma linearnih piezoelektričnih motorjev nanorobota (PiezoLEGS) in prejemanje povratnih informacij z linearnih enkoderjev (NANOS-Instruments). Omogoča krmiljenje prijemala, vsebuje pa tudi tipalo za merjenje temperature okolice. Krmilne signale za vodenje linearnih piezoelektričnih motorjev prejema iz vhodo/izhodne kartice PCI7356 (National Instrumants), nameščene v računalniku (3). Na to vhodno/izhodno kartico pa pošilja nazaj tudi povratne informacije iz linearnih enkoderjev.

Računalnik (3) je namenjen izvajanju krmilnih algoritmov v realnem času, ima vgrajen dvojederni procesor. Deluje na osnovi LabVIEW Real-Time operacijskega sistema (RealTime Desktop Target). Ta omogoča med drugim tudi, da lahko računalnik deluje kot spletni strežnik. Vgrajeno ima kartico PCI7356 (krmilnik gibanja) proizvajalca National Instruments, ki služi kot vmesnik med uporabniškim vmesnikom zgrajenim v LabView okolju (krmilnim algoritmom) in krmilnikom nanorobota. Krmilniku pošilja krmilne signale za krmiljenje posameznih osi nanorobota in z njega sprejema povratne informacije o njihovem dejanskem položaju [2].

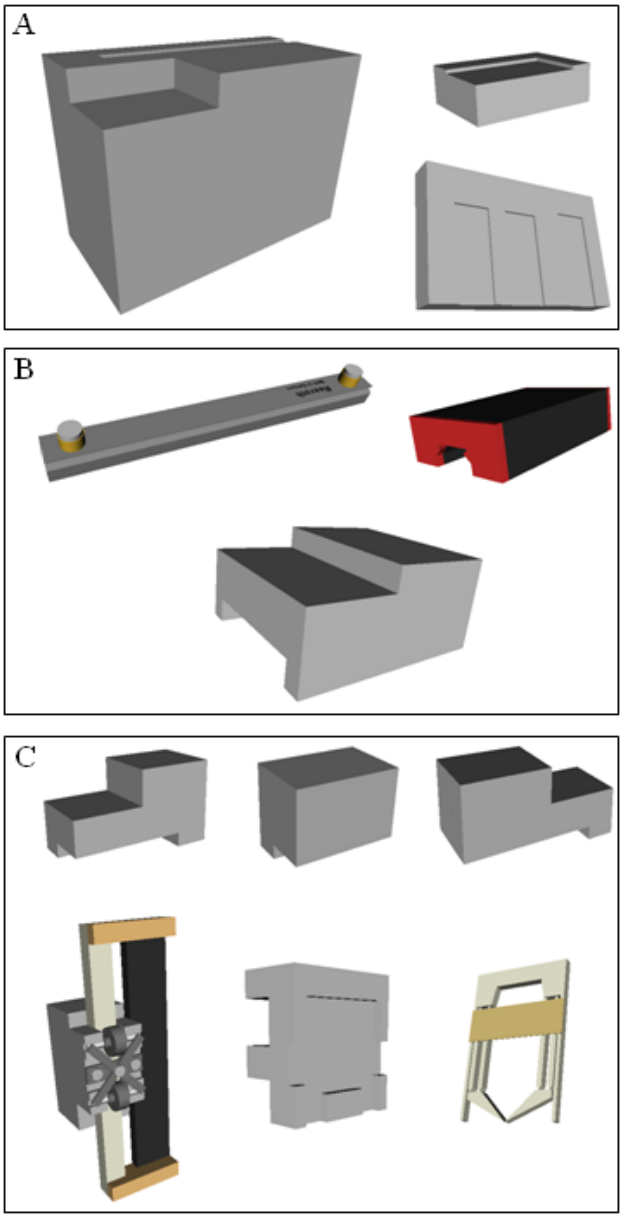
Na računalniku (2), ki deluje na osnovi operacijskega sistema Microsoft Windows XP, smo izdelali virtualni uporabniški vmesnik. Haptična naprava Phantom Omni (1) je priključena na računalnik preko vmesnika Firewire IEEE-1394. Phantom Omni je najosnovnejši model haptične naprave proizvajalca SensAble Technologies in je namenjen v akademske oziroma raziskovalne namene. Je zelo kompaktne izvedbe, ima dve funkcijski tipki, njen konec pa drži uporabnik z roko. Vgrajeni motorji služijo za generiranje sile po treh oseh (X, Y, Z), vgrajeni digitalni inkrementalni dajalniki in linearni potenciometri pa vračajo informacijo o položaju v šestih prostostnih stopnjah.

3 Virtualni uporabniški vmesnik

Uporabniški vmesnik smo razvijali s pomočjo razvojnega okolja Microsoft Visual C++ 6.0 in pripadajoče MSDN knjižnice. Programirali smo v programskem jeziku C++. Uporabili smo tudi programsko orodje OpenHaptics (Sensable Technologies), ki med drugim ponuja knjižnice, ki jih lahko uporabimo za razvoj uporabniškega vmesnika za delo s haptično napravo Phantom. Uporabili smo tudi knjižnico VRaniML (Great Hill Corporation), ki v programskem jeziku C++ omogoča vključitev VRML modelov v 3D grafične aplikacije. Vključimo lahko zunanje VRML datoteke in operiramo z virtualnimi objekti v njej. Knjižnica podpira standard VRML 2.0.

3.1 Razvoj virtualnega modela nanorobota

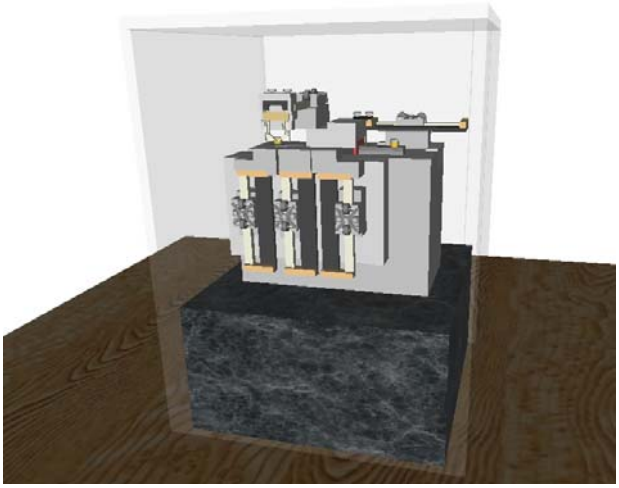
Virtualni model nanorobotske celice, ki je vključena v uporabniški vmesnik, smo izdelali s pomočjo jezika VRML. Želeli smo izdelati model nanorobota, ki naj bi čim bolj posnemal realnega nanorobota. Mere smo pridobili iz načrtov (risb, skic in podatkovnih listov) na osnovi katerih je bila izdelana in sestavljena realna nanorobotska celica. Na osnovi teh smo v urejevalnku besedil (*Notepad*), izdelali virtualni model nanorobotske celice (slika 3.1). Najprej smo izdelali podporno konstrukcijo nanorobota, ki je zgrajena iz treh komponent (slika 3.1 A). Nato smo izdelali model linearnega vodila z voziČkom, ki služi za translacijo po X osi (slika 3.1 B). Izdelali smo še model piezoelektričnega motorja PiezoLEGS, skupaj z linearnim enkoderjem in povezovalnim členom. Na koncu smo izdelali še podajalne mizice, držalo prijemala in model prijemala s piezoelektričnim aktuatorjem (slika 3.1 C).



Slika 3.1: Razvoj VRML modela nanorobota

Komponente modela nanorobotske celice, ki so zgrajene v več datotekah, smo povezali v glavni datoteki Nanorobot.wrl. V njej smo definirali začetni pogled in poti do ostalih VRML datotek. Ločili smo statične komponente in komponente, za katere želimo da se

premikajo. Končno podobo modela nanorobotske celice prikazuje slika 3.2, na kateri smo dodali še model ohišja nanorobotske celice.

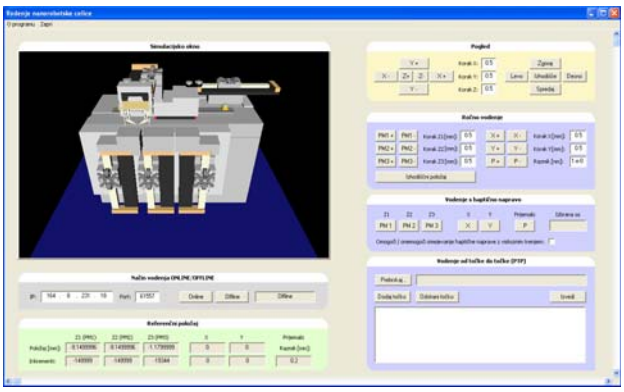


Slika 3.2: Virtualni model nanorobotske celice

3.2 Razvoj uporabniškega vmesnika

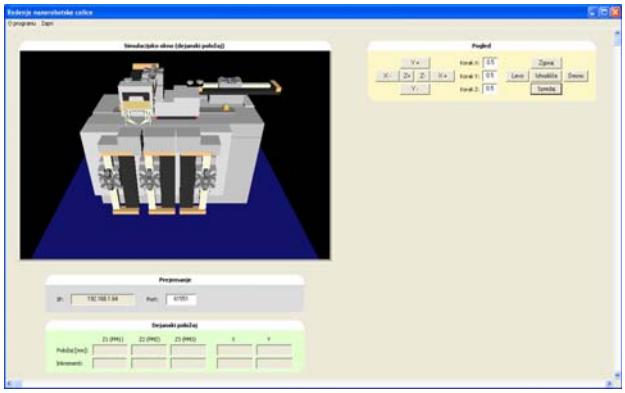
Na začetku je bilo potrebno pripraviti razvojno okolje, namestiti potrebne knjižnice in gonilnike. Nato smo začeli v izdelavo grafičnih oken našega uporabniškega vmesnika. V programskem orodju Microsoft Visual C++ smo kreirali novo Resource Script datoteko (*.rc). To je vrsta datoteke, ki omogoča gradnjo grafičnih oken (grafično programiranje). Funkcionalnost takšnega okna lahko kasneje usposobimo s kodo v ozadju (v našem primeru programski jezik C++). Izdelali smo dve glavni grafični okni, ki sta namenjeni dvema ločenima programoma. Prvo grafično okno (slika 3.3) bo namenjeno programu za krmiljenje nanorobotske celice, drugo grafično okno (slika 3.4) pa spremljanju dejanskega položaja nanorobota. Program za krmiljenje nanorobotske celice smo poimenovali Nanorobot.exe, program za spremljanje dejanskega položaja nanorobota pa Nanorobot R.exe.

V prvo grafično okno (slika 3.3) smo dodali simulacijsko podokno, kjer bomo prikazovali želeni položaj pri krmiljenju nanorobota in sicer na njegovem virtualnem modelu. Za spreminjanje pogleda na model nanorobota v simulacijskem oknu, smo dodali skupino kontrol (na rumeni podlagi, slika 3.3). Ker želimo krmiliti nanorobota na tri načine, smo izdelali tri skupine kontrol, ki bodo to omogočale. Prva skupina kontrol (na zgornji modri podlagi, slika 3.3) bo služila krmiljenju preko gumbov uporabniškega vmesnika. Druga skupina kontrol (na srednji modri podlagi, slika 3.3) bo služila krmiljenju s pomočjo haptične naprave. Tretja skupina kontrol (na spodnji modri podlagi, slika 3.3) pa bo služila za vodenje od točke do točke (*Point-to-Point*). Za nastavitve pri pošiljanju podatkov (IP naslov, port) preko omrežja smo dodali skupino kontrol na sivi podlagi na sliki 3.3. Za spremljanje vrednosti želenega položaja posameznih osi nanorobota in prijemala, smo dodali okna za izpis (na zeleni podlagi, slika 3.3).



Slika 3.3: Grafično okno programa za krmiljenje nanorobota

Drugo grafično okno (slika 3.4) je zelo podobno prvemu, z razliko da tu nimamo kontrol namenjenih krmiljenju nanorobota. Namesto kontrol za nastavitev pošiljanja, imamo tukaj nastavitve za prejemanje podatkov z nanorobota preko omrežja (na sivi podlagi, slika 3.4). Namesto oken za izpis želenega položaja pa imamo tukaj okna za izpis dejanskega položaja (na zeleni podlagi, slika 3.4). V simulacijskem podoknu bomo na virtualnem modelu nanorobota prikazovali dejanski položaj z realnega nanorobota. Enako kot pri prvem grafičnem oknu smo tudi tu dodali skupino kontrol za spreminjanje pogleda na virtualni model nanorobota.



Slika 3.4: Grafično okno programa za spremljanje dejanskega položaja nanorobota

Kodo za dodajanje funkcionalnosti grafičnemu oknu oziroma njegovih kontrol, smo izdelali v programskem jeziku C++.

V nadaljevanju bo opisan potek razvoja prvega programa (program za krmiljenje nanorobota). Pri programiranju smo najprej vključili potrebne knjižnice in opravili definicijo spremenljivk. Nato smo opravili definicijo in inicalizacijo okna za prikaz modela nanorobota ter kreirali glavno okno. Dodali smo funkcionalnost kontrolam (gumbom) glavnega grafičnega okna. Znotraj glavne funkcije programa smo zapisali kodo, za zagon novega procesa (programa Nanorobot R.exe). Nato smo dodali inicializacijo haptične naprave. Izdelali smo kodo, ki poveže VRML datoteko našega virtualnega modela nanorobota z uporabniškim vmesnikom. Znotraj glavne funkcije se nahaja zanka, ki se izvaja ves čas delovanja programa. Tukaj povežemo vrednosti, ki nam jih daje haptična naprava s spremenljivkami. Vsaka sprememba želenega položaja povzroči premik osi virtualnega nanorobota. Na koncu smo izdelali še kodo, ki omogoča delo s haptično napravo. Zapisali smo funkcijo, ki omogoča, da pri izhodu iz programa ali v primeru napake onemogoči delovanje haptične naprave. Nato smo zapisali funkcijo, ki jo imenujemo klicna zanka (*callback*) in se izvaja, kot visoko prioritetna nit programa na asinhroni način. Naš program se bo tako izvajal večnitno. S tem bomo zagotovili ustrezno oziroma stabilno delovanje. Znotraj klicne zanke se izvaja tako imenovana servo zanka, ki se mora osveževati s frekvenco vsaj 1kHz [1]. Znotraj servo zanke smo dodali funkcionalnost funkcijskima tipkama na haptični napravi. Prvo funkcijsko tipko smo uporabili za to, da z njo zgrabimo os nanorobota, ki jo želimo krmiliti. Drugo funkcijsko tipko pa smo uporabili za preklop med osmi nanorobota pri krmiljenju. Nato smo zapisali kodo, ki bo namenjena preprečevanju kolizije med posameznimi osmi nanorobota. S kodo za izračun in generiranje sile, pa dosežemo občutek dotika ob koliziji ali v končnem položaju osi (motorja). Dodali pa smo tudi kodo, ki omogoča generiranje omejevanja peresa haptične naprave v obliki viskoznega trenja. Vse to smo zapisali znotraj servo zanke. Nato smo dodali še funkcijo, ki omogoča izvedbo in načrtovanje vodenja nanorobota od točke do točke (*Point-to-Point*). Koda je zapisana tako, da bere ukaze iz tekstovne datoteke in jih izvaja na nanorobotu. Ukaze lahko zapišemo direktno v tekstovno datoteko ali pa jih dodajamo in brišemo z uporabo uporabniškega vmesnika. Na koncu smo zapisali še funkcijo za pošiljanje podatkov iz uporabniškega vmesnika preko omrežja, realnemu nanorobotu. Za to smo izbrali protokol UDP, ki je sicer nepovezovalni protokol za prenašanje paketov. Odjemalec in strežnik tukaj ne vzpostavita povezave, ampak strežnik pošilja pakete odjemalcu in ne preverja, če je odjemalec pakete dobil.

Razvoj drugega programa (program za spremljanje dejanskega položaja nanorobota), smo opravili na osnovi prvega. Vendar pa je bilo potrebno tukaj dosti manj kode. Za razliko od prvega programa tukaj nimamo kontrol za krmiljenje nanorobota. Namesto pošiljanja podatkov, pa smo tu morali realizirati sprejem podatkov (podatki o dejanskem položaju). Znotraj funkcije za sprejem smo dodali kodo, ki preverja ali je sprejet podatek v ustrezni obliki. Vsak sprejet podatek, ki je v ustrezni obliki povzroči prikaz na virtualnem modelu in oknih za izpis dejanskega položaja. Tudi sprejem podatkov smo izdelali na osnovi protokola UDP.

Okno uporabniškega vmesnika smo izdelali v dveh jezikih in sicer slovenskem ter angleškem. Uporabnik lahko tako pri zagonu uporabniškega vmesnika izbere, v katerem jeziku ga želi imeti prikazanega.

Na sliki 3.5 je prikazan celotni sistem in delovno okolje za krmiljenje nanorobotske celice s haptično napravo.

Slika 3.5: Aplikacija za krmiljenje nanorobotske celice s haptično napravo

4 Zaključek

V članku smo v začetku opisali namen, zakaj smo se odločili za krmiljenje nanorobotske celice s haptično napravo. Nato smo opisali že obstoječi sistem, nanorobotsko celico, krmilno elektroniko, računalnik (spletni strežnik) z vgrajeno kartico PCI7356 in haptično napravo Phantom OMNI. V nadaljevanju smo predstavili razvoj uporabniškega vmesnika. Tukaj smo predstavili najprej razvoj virtualnega modela nanorobota v jeziku VRML. Nato smo predstavili še razvoj obeh programov uporabniškega vmesnika, v katera smo vključili naš virtualni model nanorobota.

Prednosti sistema, za katerega smo izdelali uporabniški vmesnik je, da dobimo pri delu s haptično napravo občutek delovanja na okolico oziroma občutek dotika. Torej občutimo lahko kdaj smo posamezno os nanorobota pripeljali v končni položaj, kdaj pride do kolizije med posameznimi osmi. V tem primeru nam haptična naprava generira proti silo, kar občutimo kot nek končni položaj. Tako lahko ob dodatnem programskem preprečevanju kolizije obvarujemo realnega nanorobota pred poškodbami. Haptična naprava pa omogoča tudi generiranje raznih trenj ali vibracij, s katerimi lahko še izpopolnimo vmesnik in se s tem še bolj približamo realnemu občutku. Prednost, ki jo prinaša vmesnik je tudi ta, da lahko izdelamo program izvajanja nanorobota (izvedba PTP gibov), ki ga najprej načrtujemo lokalno (*off-line*) in ko smo s tem programom zadovoljni ga prenesemo na realnega nanorobota (*on-line* način krmiljenja).

Pomankljivosti sistema so predvsem te, da lahko iz uporabniškega vmesnika krmilimo nanorobota hitreje, kot pa je njegova sposobnost (omejen je z dinamiko motorjev). Težavo smo sicer rešili za direktno krmiljenje s haptično napravo, kjer smo dodali omejevanje hitrosti gibanja peresa haptične naprave. Zmožnost omejevanja hitrosti smo dosegli z generiranjem viskoznega trenja. Pri samem krmiljenju preko gumbov uporabniškega vmesnika ali vodenju s programom z ukazi PTP gibov, pa lahko krmilimo hitreje in motorji nanorobota niso sposobni slediti. V ta namen bi bilo dobro prilagoditi še ta dva načina krmiljenja. Pri krmiljenju z gumbi uporabniškega vmesnika bi bila rešitev ta, da bi po pritisku na gumb za premik posamezne osi za en korak, preverili kakšen je ta korak (sprememba položaja) in nato na podlagi tega izračunali čas, kako dolgo potrebuje realni nanorobot, da se premakne za ta nastavljeni korak. Za ta čas bi nato skrili ali onemogočili ta gumb. Pri krmiljenju s programom s PTP gibi pa bi bilo potrebno izdelati tako imenovani PTP preoblikovalnik s $\sin^2$ pospeškovnim profilom. Ta bi dele programa, ki jih realni nanorobot ni sposoben izvesti, preoblikoval v obliko primerno za izvedbo.

5 Literatura

[1] Sensable technologies: 3D Touch SDK Open Haptics Toolkit, različica 1.02, Programmers Guide.

[2] G. Škorc, J. Čas, S. Brezovnik, R. Šafarič: Simulacija nano robotske celice s podporo za 6D - HID, Zbornik sedemnajste mednarodne Elektrotehniške in računalniške konference ERK 2008, 29. september – 1. oktober 2008, Portorož, Slovenija, zv. A, str. 245-248, 2008.

[3] ftp://ftp.cordis.europa.eu/pub/nanotechnology/docs/nano_brochure_sl.pdf/.