Interakcija s humanoidnim robotom

Knežević, Tara

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Tara Knežević

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Marko Švaco, mag. ing. mech.

Student:

Tara Knežević

Zagreb, 2022.

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

Zahvaljujem doc. dr. sc. Marku Švaci na zadavanju teme završnog rada i prof. dr. sc. Bojanu Jerbiću na pruženim savjetima pri izradi završnog rada.

Veliko hvala obitelji na bezuvjetnoj potpori, strpljenju i motivaciji.

Hvala prijateljima i kolegama što su mi uljepšali i olakšali studentske dane.

Tara Knežević

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: **Tara Knežević** JMBAG: **0035223365**

Naslov rada na hrvatskom jeziku: **Interakcija s humanoidnim robotom**

Naslov rada na engleskom jeziku: **Interaction with a humanoid robot**

Opis zadatka:

Humanoidna robotika intenzivno se istražuje i razvija kroz znanstvene ali sve više i komercijalne projekte. Robot Pepper jedna je od komercijalno dostupnih humanoidnih robotskih platformi te je u svijetu jedan od najviše korištenih autonomnih humanoidnih robota. Humanoidni robot Pepper sadrži sve osnovne mehaničke (glava, torzo, funkcionalne ruke i pojednostavljene šake), senzorske (mikrofoni, ultrazvučni senzori, vizijski sustav, enkoderi) i upravljačke komponente koje mu omogućavaju postizanje visoke razine autonomije.

U završnom radu potrebno je upoznati se s radom i mogućnostima robota Pepper te napraviti sljedeće:

- Ispitati i istražiti osnovne mogućnosti robota Pepper,
- Ispitati i usporediti različite načine programiranja robota,
- Ispitati mogućnosti sustava usmjeravanja pažnje,
- Istražiti mogućnosti algoritama za kretanje i prostornu navigaciju po zadanim putanjama,
- Istražiti sustav za prostorno prepoznavanje i praćenje markera,
- Ispitati sustav za prepoznavanje govora i eksperimentalno validirati scenarij verbalne interakcije robota s čovjekom 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

SAŽETAK

Povijesno gledano, zadaća robota oduvijek je bila rad na radnim mjestima koja su zamorna, opasna ili teška ljudima. Međutim, pojavom razvijenijih humanoidnih robota, mogućnosti rada robota su se proširile. Primjerice, na uslužne i njegovateljske djelatnosti. Ono što predstavlja najveći izazov u tim djelatnostima je mogućnost jednoznačne i precizne interakcije s čovjekom.

Završni rad se temelji na upoznavanju sa softverom *Choregraphe* te implementaciji programa na robotu *Pepper*. *Pepper* je humanoidni robot koji je na tržištu dostupan od 2015. godine, a proizvod je japanskog *SoftBank*-a i francuskog *Aldebaran Robotics*-a.

Cilj rada bio je istražiti i evaluirati načine programiranja robota te se upoznati s njegovim gotovim funkcijama. Također, ispitati njegove mogućnosti (govor, senzori, vizijski sustav, kretanje u prostoru) te osmisliti rješenje u kojem robot ima ulogu vodiča kroz Laboratorij računalne inteligencije u CRTA-i (Regionalni centar izvrsnosti za robotske tehnologije).

Ključne riječi: *Pepper*, *SoftBank*, humanoidni robot, *Choregraphe*, interakcija

SUMMARY

Through history, robots were designed to work at jobs that are tiresome, dangerous or hard for humans. With the emergence of more advanced humanoid robots the possibilities expanded. For example, there are robots in service and nursing workplaces. The biggest challenge that emerged from those workplaces is to create unambiguous and precise interaction with a human.

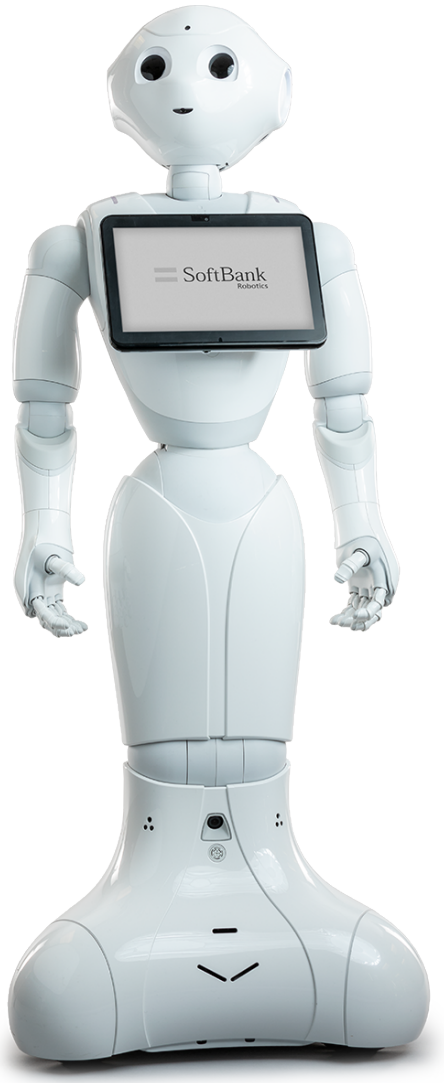
Bachelor's thesis is based on acquainting with software Choregraphe and implementing it on robot Pepper. Pepper is a humanoid robot released in 2015. He is a product of Japanese SoftBanks and French Aldebaran Robotics.

The main goal of the thesis was to explore and evaluate different ways to program the robot and to get acquainted with its embedded functions. Also, the goal was to test what the robot is capable of (speech, sensors, vision system, motion in space) and to come up with a solution for a task in which Pepper is a guide through the Laboratory of computer intelligence in CRTA (Regional Center of Excellence for Robotic Technology).

Key words: *Pepper*, *SoftBank*, humanoid robot, *Choregraphe*, interaction

1. UVOD

Pepper je robot iza kojeg je ideja bila napraviti komercijalnog humanoidnog robota dostupnog javnosti (raznim ustanovama, firmama, školama). Da je to postignuto potvrđuje činjenica da je prvih tisuću jedinki rasprodano unutar minute nakon što je izašao u prodaju. Dakle, od samih početaka, *Pepper* je privukao veliku pažnju i interes. Što se tiče tehničkog opisa, *Pepper* je opremljen mikrofonima, kamerama, senzorima (žiroskop, ultrazvučni, laserski, taktilni) te ima dvadeset stupnjeva slobode. Baterija mu traje do 12 sati što ga čini odličnim izborom za primjenu na recepcijama hotela, restorana i slično [1]. Primjerice, postoje istraživanja u kojima je *Pepper* radio na recepciji [2] i brinuo za osobe starije dobi [3].



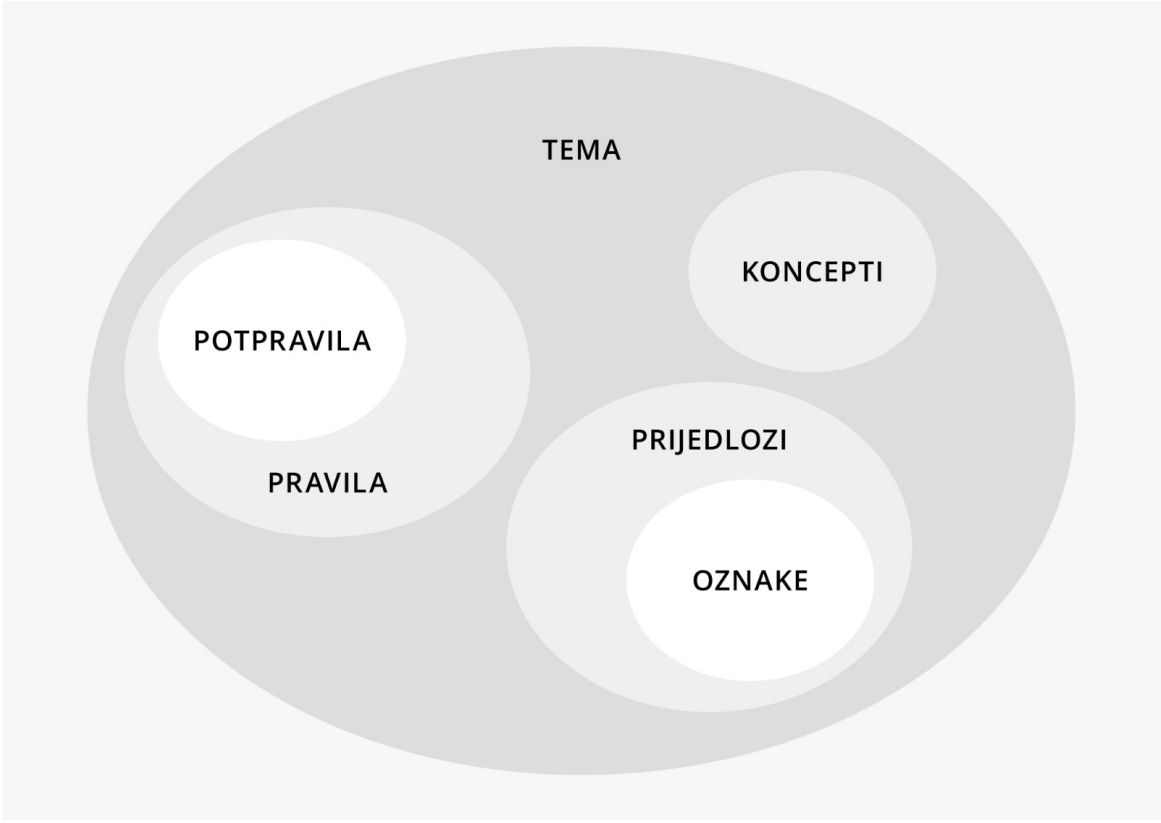
Slika 1. Robot *Pepper* [4]

U *Choregraphe*-u se programira koristeći blokove. Unutar svakog bloka piše se kod, a zatim se više blokova međusobno spoji te u konačnici čine program. Ovaj softver nudi veliki broj već gotovih opcija, od jednostavnijih do kompliciranijih koje u sebi imaju implementirane neuronske mreže. Vrste prijenosa dijele se na prasak (engl. *Bang*) i podatke (engl. *Data*). Prasak je prijenos kod kojeg nema prijenosa informacija, nego samo dolazi do aktivacije bloka. Može se zamisliti kao nisko i visoko logično stanje. Prije aktivacije bloka, on je u niskom stanju te je neaktivan. Zatim dolazi do praska koji prebacuje blok u visoko, tj. aktivno stanje. Kod podatkovnog prijenosa dolazi do prijenosa nekog broja, teksta ili dinamičkog podatka (ne mora se unaprijed znati radi li se o broju ili tekstu). Unutar *Choregraphe*-a ne postoji modul hrvatskog jezika, pa je sve programirano na engleskom jeziku.

U samom radu prvo će biti objašnjen sustav govora robota, zašto je bitan, načini na koje se može koristiti te će biti navedeno i objašnjeno zašto su neki načini programiranja bolji od drugih. Na isti način će biti i objašnjena treća tema, što je sustav kretnji i kretanja robota. Kod kretanja robota biti će provedena validacija određenih načina kretanja robota po proizvoljno odabranim parametrima. Zatim će kratkim primjerima biti objašnjen osjetilni sustav, tj. funkcije vezane za vizijski sustav i taktilne senzore. Nakon toga biti će opisano na koji način su se koristile programske funkcije i koji je njihov značaj te korištenje tableta robota. Na kraju će biti pokazan i objašnjen primjer koji je sinteza svih navedenih sustava. U tom primjeru, *Pepper* ima ulogu vodiča te mu je zadatak provesti čovjeka kroz Laboratorij računalne inteligencije te mu predstaviti tri robota unutar laboratorija. Pri vođenju, mora voditi konverzaciju, kretati se u prostoru, vršiti razne tjelesne kretnje, koristiti vizualni sustav i senzore kojima je opremljen.

2. SUSTAV GOVORA

Naglasak završnog rada stavljen je na govor i povećanje robusnosti razgovora između korisnika i robota. Najlakši pristup realizaciji govora robota je korištenje gotove opcije izgovori (*engl. Say*). Ova opcija dozvoljava korisniku da upiše određeni tekst i da ga Pepper, nakon aktivacije tog bloka (engl. *Box*) izgovori. Međutim, ova opcija donosi veliku nesigurnost u pravovaljanom izvođenju programa te mu povećava kompleksnost. Također, ne nudi opciju oblikovanja govora, primjerice mijenjanjem brzine govora, dodavanjem stanki između rečenica, varijacijom glasnoće i slično. Ovu opciju opravdano je koristiti na onim mjestima unutar programa gdje je cilj da *Pepper* izgovori nešto, ali ne i tamo gdje se očekuje uspostavljanje konverzacije s korisnikom. Kako bi se uspostavila konverzacija, najbolje je koristiti dijalog (engl. *Dialog*) blok. On se programira pomoću *QiChat*-a, skriptnog programskog jezika. Unutar svakog dijalog bloka možemo definirati temu (engl. *Topic*), pravila (engl. *Rules*), prijedloge (engl. *Propositions*), oznake (engl. *Tags*) i koncepte (engl. *Concepts*) (Slika 2).



Slika 2. Podjela komponenata unutar *QiChat*-a

Tema je objekt unutar kojeg se nalaze pravila, prijedlozi, oznake i koncepti. Pravila su poveznica između korisnika i robota. Ona su definirana na sljedeći način – „u“ označava da se radi o pravilu, sve unutar zagrada je korisnički unos, a ono što slijedi iza njih je izlaz robota. Postoje i potpravila, koja se mogu zamisliti kao uvjetna naredba, a funkcija im je da povećaju

opseg razgovora davanjem opcija korisniku. Potpravila se numeriraju, te ih najviše može biti tri („u1“, „u2“, „u3“), a svako je potpitanje prethodnog. Dodatna opcija unutar pravila je koristiti određene događaje (engl. *Events*). Ako događaje definiramo unutar dijela pravila koje se odnosi na korisnikov unos, „e“ označava događaj. Događaj se može aktivirati i preko dijela pravila koji se odnosi na izlaz robota. Aktivacija događaja je korisna jer omogućuje povezanost više sustava robota (primjerice kretanja i govora), ali i reakciju robota na podražaje (bilo da su oni vizualnog ili senzorskog porijekla). Prijedlozi (Slika 3) su dio razgovora koji je uvijek iniciran od strane robota. Aktiviraju se određenom funkcijom kako bi se ostvarila progresija razgovora, a identificiraju se pomoću oznaka.

```
proposal : % IME_PRIJEDLOGA     robot_pitanje_1
    u1 : (čovjek_odgovor_A)     robot_odgovor_A
    u1 : (čovjek_odgovor_B)     robot_pitanje_2
        u2 : (čovjek_odgovor_B1)     robot_odgovor_B1
        u2 : (čovjek_odgovor_B2)     robot_pitanje_3
                u3 : (čovjek_odgovor_B3)     robot_odgovor_B3
```

Slika 3. Sintaksa prijedloga unutar *QiChat*-a

Koncept (Slika 4) je način na koji se ostvaruje kompaktnost programiranja unutar *QiChat*-a. Ideja je grupirati sinonime i fraze koje imaju slično značenje kojim se opisuje neki pojam te ih povezati s riječima koje bi se mogle nalaziti uz njih. Koncepti se uvijek definiraju iznad koda koji se izvodi, a mogu se koristiti i za ljudski unos i robotski izlaz. Pri unosu korisnika, koncepti omogućavaju robotu da prepozna razne varijacije govora. Pri izlazu robota prvo bude generirana nasumična riječ iz koncepta, a nakon toga se izmjenjuju sekvencijalno, što doprinosi raznolikosti govora robota.

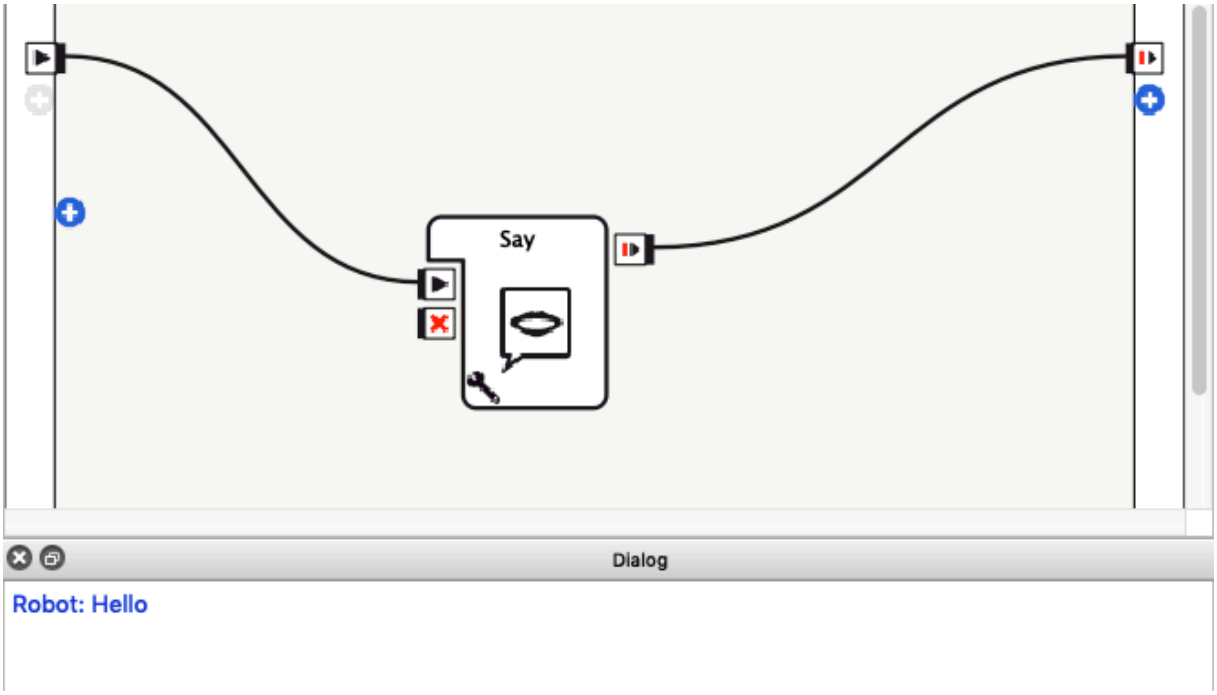
```
concept: (ime_koncepta) {opcionalna_riječ_1}
    {opcionalna_riječ_2} · · · {opcionalna_riječ_n}
    [sinonim_1    sinonim_2 · · · sinonim_n]
    {opcionalna_riječ_n+1} · · ·
```

Slika 4. Sintaksa koncepata unutar *QiChat*-a

Slijede primjeri u kojima će podrobnije biti objašnjeno korištenje navedenih opcija.

2.1. Naredba izgovori

Ova je naredba (`'Say'`) već navedena kao najjednostavniji način uspostavljanja realizacije govora. Odabire se gotov blok unutar kojeg se definira tekst, u ovom primjeru to je pozdrav (engl. *Hello*). Program je izveden na robotskom simulatoru te je, nakon aktivacije bloka u dijalog prozoru (engl. *Dialog*) prikazan rezultat.



Slika 5. Naredba izgovori

2.2. Izgovori blok

Slika 6 prikazuje programiranje unutar QiChat-a (`'Dialog Box'`). U osmoj liniji koda je predviđeno da, čim se pokrene blok unutar dijagrama tijeka programa, robot pokrene konverzaciju s korisnikom. Nadalje, u desetoj liniji prikazana je aktivacija događaja vezana za izlaz robota. Kako bi se taj događaj ostvario, potrebno je dodati izlaz bloku pod nazivom „outputLookAt“ (u ovom primjeru). Također, u tom dijelu koda, izlazu se pridodaje vrijednost jedan, što označava pobuđeno, tj. aktivno stanje. Interpretacija osme linije koda je sljedeća – nakon aktivacije bloka, robot sam pokreće prijedlog definiran oznakom „LIKE“ te pokreće konverzaciju. Unutar većine linija prikazanog koda, nalaze se naredbe za oblikovanje glasa koje se pišu unutar kosih linija. Primjerice, u desetoj liniji koda naredba za oblikovanje brzine govora (`'rspd'`) postavlja ju na 75%, a naredba za oblikovanje dodavanjem stanke označava stanku (`'pau'`) od 250 milisekundi prije izvršenja izlaza.

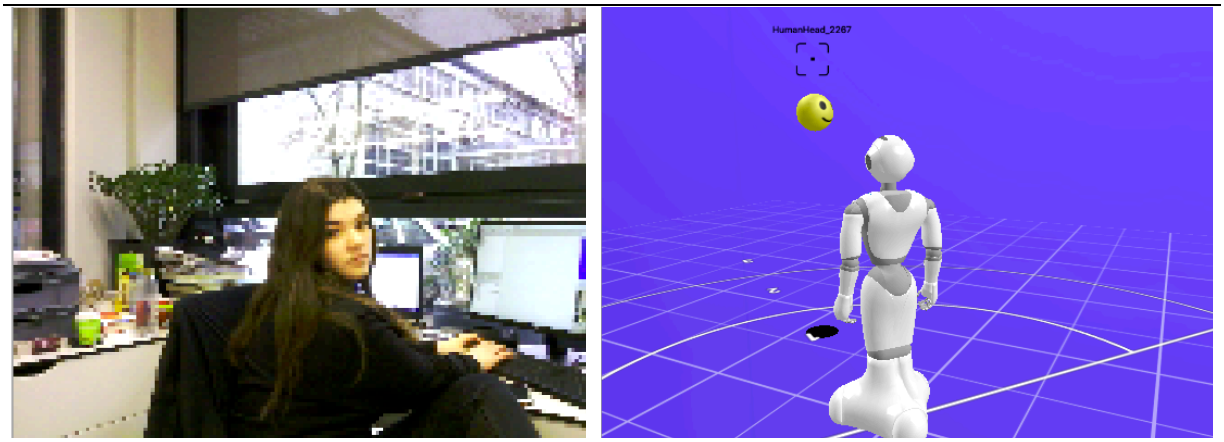
```
topic: ~Next()
language: enu

concept: (yes) {I} {changed} {my} [mind yes ok certainly definitely "of course" gladly indeed yeah yup sure "you bet" totally alright] {"I would"} {like} {that} {to} {meet} {see} {you} {the} {dance} {show} {me}
concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" "not really"] {sorry} {I} {do not} {don't} {want} {to} {wanna}


u:(e:onStart) \rspd=75\  ha ha \pau=250\ ^gotoReactivate(LIKE)

proposal: %LIKE \rspd=75\ Did you like that \pau=250\
    u1:(~yes) thank you! \pau=500\ I would like to show you something else ^gotoReactivate(DANCE)
    u1:(~no) \pau=250\ ok. \pau=150\ I can dance too. \pau=250\ Do you want to see it?
        u2:(~yes) \pau=250\ great! \pau=200\ Look at these moves! $onStopped=1
        u2:(~no) \pau=250\ Ok. Tell me if you change your mind. ^stayInScope

u:(~yes) yay! $onStopped=1

proposal: %DANCE \pau=500\ This is a dance I know $onStopped=1
```

Slika 6. Dijalog blok

U ovom primjeru definirana su dva koncepta. Jedan za načine kako reći ne, a drugi kako reći da. Unutar uglatih zagrada nalaze se sinonimi i fraze koji odgovaraju određenom konceptu, a unutar vitičastih zagrada su riječi i fraze koji se mogu, ali i ne moraju nalaziti uz riječi iz uglatih zagrada. Važno je za napomenuti kako je mogućnost da se više opcionalnih riječi (vitičaste zagrade) koristi odjednom, ali glavna riječ (uglate zagrade) može biti samo jedna. Što se više izbora doda, to je razgovor robusniji.

2.3. Ljudska percepcija i zone

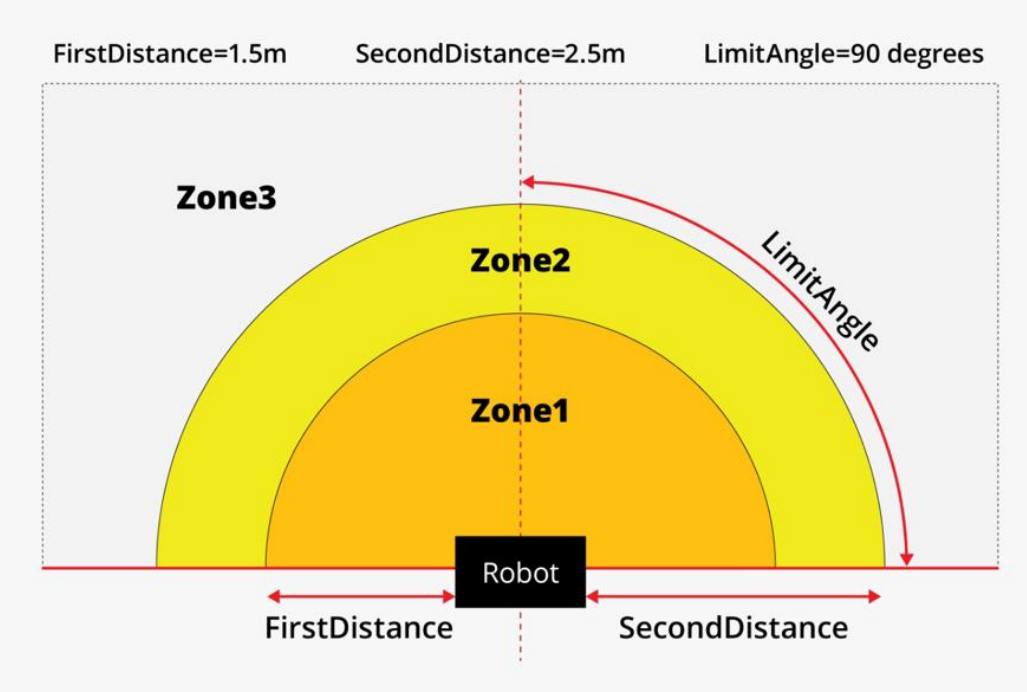
Pepper je napravljen tako da njegov vizijski sustav automatski prepozna ljudsko lice preko modula ljudska percepcija (engl. *People perception*). Zanimljivo je za napomenuti kako je isto isprobano ako čovjek nosi masku za lice i u tom slučaju *Pepper* ne prepoznaje ljudsko lice. Slika 7 prikazuje što je vidljivo na kameri te kako *Pepper* prikazuje ljudsko lice u svom okruženju. Bitno je za napomenuti da istovremeno može prepoznati više lica, ali razgovor voditi samo s jednom osobom.



(a) (b)

Slika 7. (a) Prikaz s kamere (b) Prikaz *Pepper*-ove simulacije okruženja

Područje u kojem robot promatra svoju okolinu je ograničeno udaljenošću i kutom te je podijeljeno na dvije zone (Slika 8).



Slika 8. Zone ograničenja promatranja robota [5]

Kada se spoje ljudska percepcija i zone, može se promatrati kretanje korisnika unutar tih zona. Primjerice, može se uočiti da je korisnik ušao u zonu jedan iz zone dva te je zaključak da se korisnik približava robotu (`'EngagementZones/PersonApproached'`).

Slika 9 daje primjer u kojem je upravo pomoću ovog koncepta u 22. liniji koda definirano da kada robot uoči da mu korisnik prilazi, inicira govor pozdravom. To je primjer korištenja podizanja događaja za aktivaciju robota. Međutim, postoji opcija da korisnik prebrzo priđe robotu te da on ne detektira promjenu zona. U tom slučaju, dodane su još dvije opcije, da se

robot aktivira pozdravom korisnika (21. linija koda) ili aktivacijom taktilnog senzora na glavi (23. linija koda, `'inputHead'`).

```
13
14 concept: (good) {"I am"} {"I'm"} {feeling} {doing} {pretty} [good excellent outstanding "all right" great okay ok fine
   "not bad" well awesome] {"Thank you"} {Thanks} {"How are you"} {"How about you"} {feeling} {doing} {you}
15 concept: (bad) {"I am"} {"I'm"} {feeling} {doing} {pretty} [bad terrible horrible awful "not [great good well]"] {"Thank
   you"} {Thanks} {"How are you"} {"How about you"} {feeling} {doing} {you}
16 concept: (hello1) [hi hello hey "hi there" greetings "good day"] {Pero} {Pepper}
17 concept: (hello2) [hi hello "hi there" greetings]
18 concept: (yes) [yes certainly definitely "of course" gladly indeed yeah yup sure "you bet" totally alright] {"I would"}
   {like} {that} {to} {meet}
19 concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" ] {sorry} {I} {do not} {don't}
   {want} {to} {wanna}
20
21 #u:(~hello1) ~hello2 $outputapproach=1 \pau=500\  ^gotoReactivate(WELCOME)
22 u:(e:EngagementZones/PersonApproached) ~hello2 \pau=500\ $outputapproach=1 ^gotoReactivate(WELCOME)
23 u:(e:inputHead) ~hello2 $outputapproach=1 \pau=500\ ^gotoReactivate(WELCOME)
24
25
26 proposal: %WELCOME $outputWelcome=1  Welcome to the regional center of excellence for robotic techology. \pau=250\
   $outputLookAt=1 \rst=100\ \rspd=75\ How are you doing? $outputLookAt=1
27     u1:(~good) \pau=500\ Great!^gotoReactivate(NAME)
28     u1:(~bad) \pau=250\ I am so sorry to hear that!^gotoReactivate(NAME)
29
30 proposal: %NAME \pau=500\ Can you tell me your name? $outputQuestion=1 $outputLookAt=1
31     u1:(~no) It is ok that you don't want to share it. \pau=500\ My name is Pepper $outputRobots=1
32     u1:(e:Dialog/NotUnderstood) \pau=1000\ I like that name! My name is \rspd=75\ Pero\pau=250\ $outputRobots=1
33
```

Slika 9. Aktivacija robota prilaskom korisnika i definicija proizvoljnih izlaza

2.4. Proizvoljni izlazi

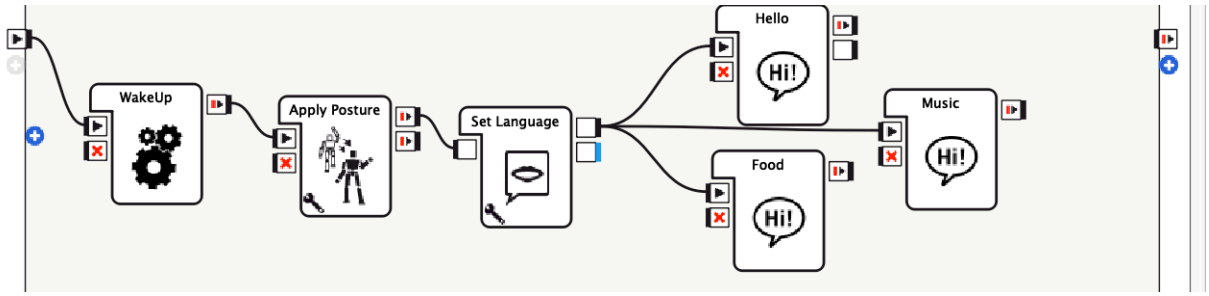
Već je spomenuto da, ako se želi pokrenuti neku aktivnost unutar dijalog bloka, tj. povezati govor s nekim ponašanjima robota, treba se dodati izlaze za aktivaciju te ih definirati unutar koda u QiChat-u. U 26. liniji koda (Slika 9) vidljivo jer da su aktivirana tri izlaza, ali ne istovremeno. Prvi izlaz je definiran prije nego što robot išta izgovori, drugi nakon prve rečenice, a treći nakon druge rečenice. Drugi i treći izlazi zapravo aktiviraju istu akciju, pa nije potrebno definirati poseban izlaz za svaki, nego jedan koji će oboje aktivirati. Također, na taj izlaz iz bloka potrebno je spojiti samo jedan blok s naredbom pogledaj u smjeru (engl. *Look at*).

U 32. liniji za aktivaciju robotskog odgovora iskorišten je događaj neprepoznavanja govora (`'Dialog not understood'`). To je dobar način da se izbjegne greška prilikom razgovora, jer većina imena s kojima je *Pepper* komunicirao su bila hrvatska te ih on ne bi razumio pa bi zaustavio program. Na ovaj način, on odgovara na bilo što izrečenog. Stavljena je i stanka od jedne sekunde prije nego što robot odgovori na ime, kako bi se osiguralo da ga je korisnik izgovorio do kraja.

2.5. Scenarij verbalne interakcije

Sljedeći primjer formiran je tako da se s *Pepper*-om može voditi razgovor o više tema paralelno, s mogućnošću vraćanja na teme, tj. blokove koji su već bili aktivirani. Kako bi to bilo moguće, blok „izgovori“ nema izlaz jer bi njegovom aktivacijom on postao neaktivan. Svi blokovi

moraju biti aktivni, a jedino što se između njih mijenjati je fokus (engl. Focus). Dakle, potrebno je definirati riječi i rečenice koje će imati funkciju okidača (engl. *Trigger*) nekog bloka. Osim toga, slijedi i paralelni prikaz dvije verbalne interakcije (Slika 11) kako bi se prikazalo da isti kod može imati različite ishode ako je programiran koristeći potpravila.



Slika 10. Konverzacija o više tema paralelno

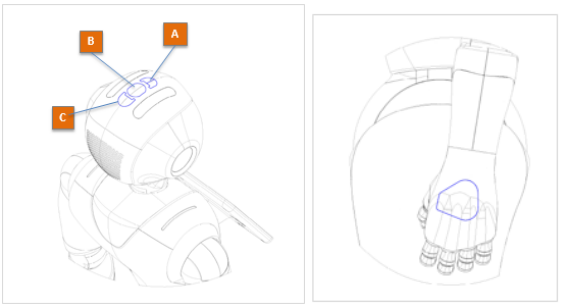
Napomena: robot započinje verbalnu interakciju tako što je aktiviran detekcijom prilaska korisnika.

Slika 11. (a) i (b) Primjer verbalne komunikacije

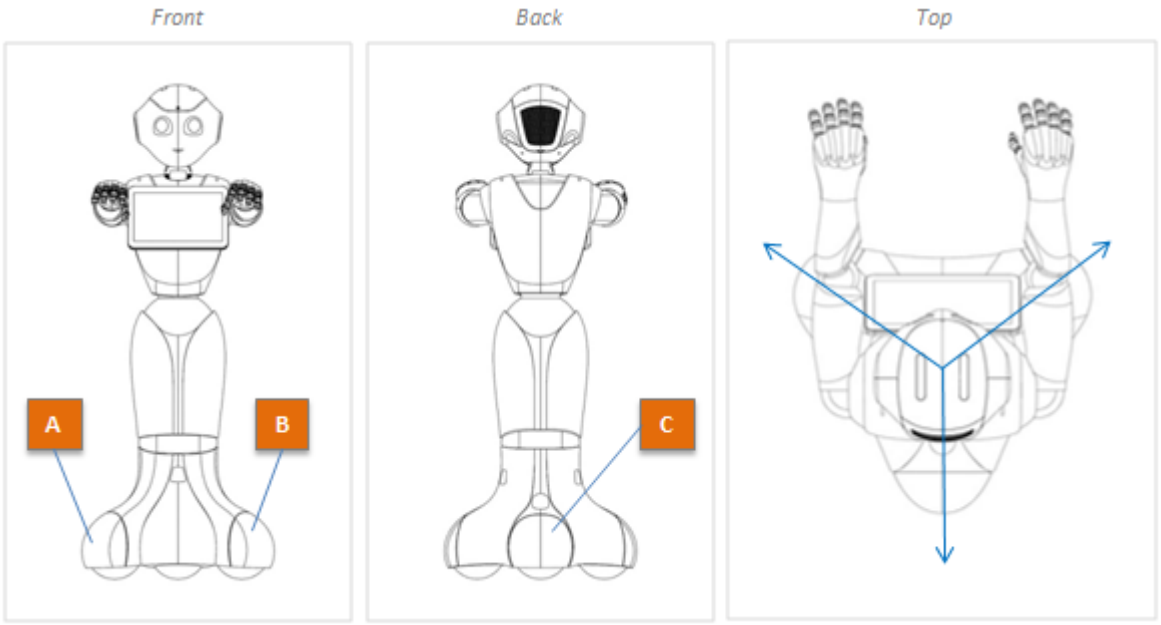
3. PERCEPCIJA OKOLINE

3.1. Taktilni senzori

Taktilni senzori na robotu mogu se podijeliti u tri grupe. Prva grupa su taktilni senzori glave te postoje tri (prednji, srednji i stražnji). Druga grupa su taktilni senzori na rukama, na svakoj postoji jedan i to sa stražnje strane ruke. Treća grupa su taktilni senzori na branicima, po jedan na svakom od tri branika. Slijede slike položaja sva tri senzora, a nakon toga objašnjenje u kojem slučaju se oni koriste te primjeri.



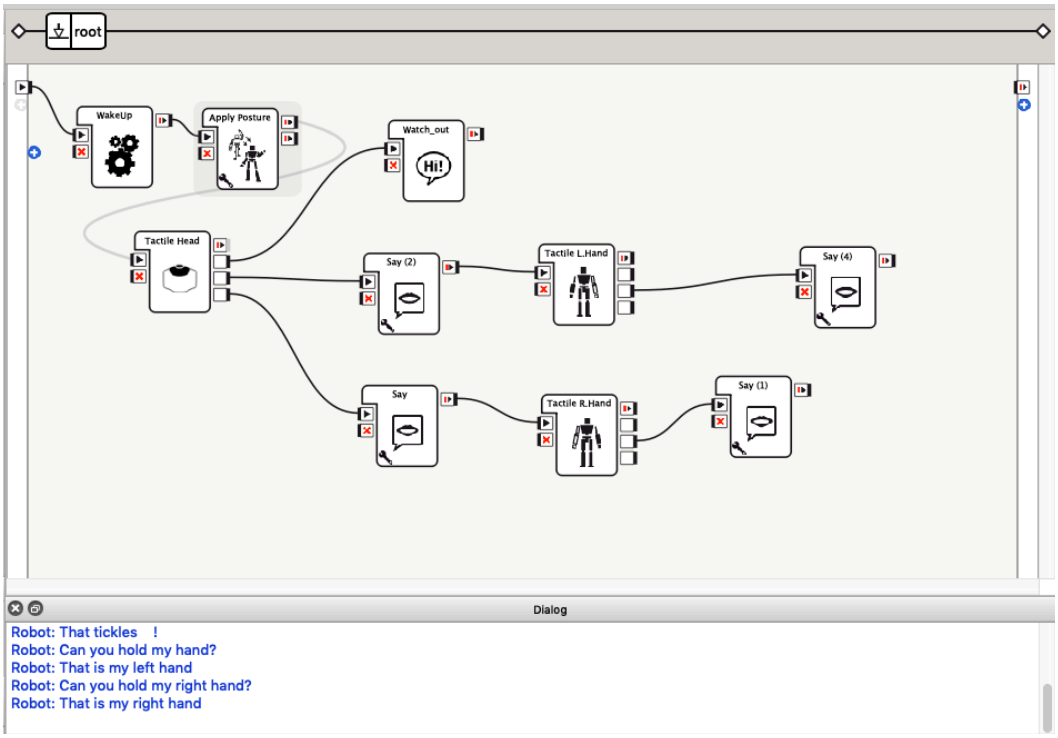
Slika 12. Pozicija taktilnih senzora: na glavi (lijevo), na ruci (desno) [9]



Slika 13. Pozicija taktilnih senzora na branicima [9]

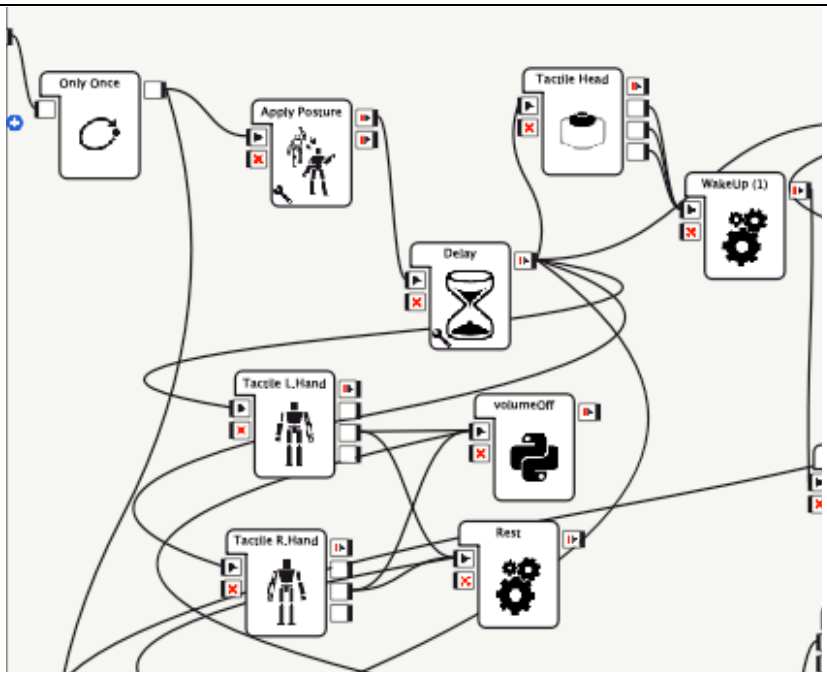
Senzori na branicima su tamo iz sigurnosnih razloga. Ako bi se oni koristili za neke druge primjene, primjerice za aktiviranje govora, izgledalo bi kao da korisnik udara robota nogom kako bi dobio odgovor što nije poželjno. Ali, zato su ti senzori neophodni za sigurnost robota u

slučaju da zakažu već spomenuti ultrazvučni senzori u bazi robota. Pošto su branici najširi dio tijela robota, oni će u koliziji prvi dotaknuti prepreku te će poslati signal robotu da se zaustavi. Taktilni senzori glave i na rukama se koriste kako bi aktivirali određenja ponašanja robota ili kao događaji za aktivaciju unutar dijalog bloka. Slijedi primjer u kojem je prednji senzor glave spojen na dijalog blok te kada se aktivira, robot izgovori sljedeće: „*That tickles*!“ Središnji senzor spojen je na izgovori blok nakon kojeg robot pita korisnika može li ga primiti za ruku. Ako korisnik primi robota za desnu ruku ništa se neće desiti, ali ako ga primi za lijevu ruku, robot će ga obavijestiti da je to njegova lijeva ruka. Naposljetku, aktivacijom stražnjeg senzora na glavi, aktivira se izgovori blok nakon kojeg robot pita korisnika može li ga primiti za desnu ruku. Nakon toga ga robot obavijesti da to stvarno je njegova ruka. Slika 14 osim programa prikazuje i rezultat simulacije. U ovom primjeru cilj je bio pokazati da svi senzori rade i da su dobro razlučivi, pa čak i oni na glavi koji nemaju fizički vidljivu razdiobu.



Slika 14. Program simulacije taktilnih senzora

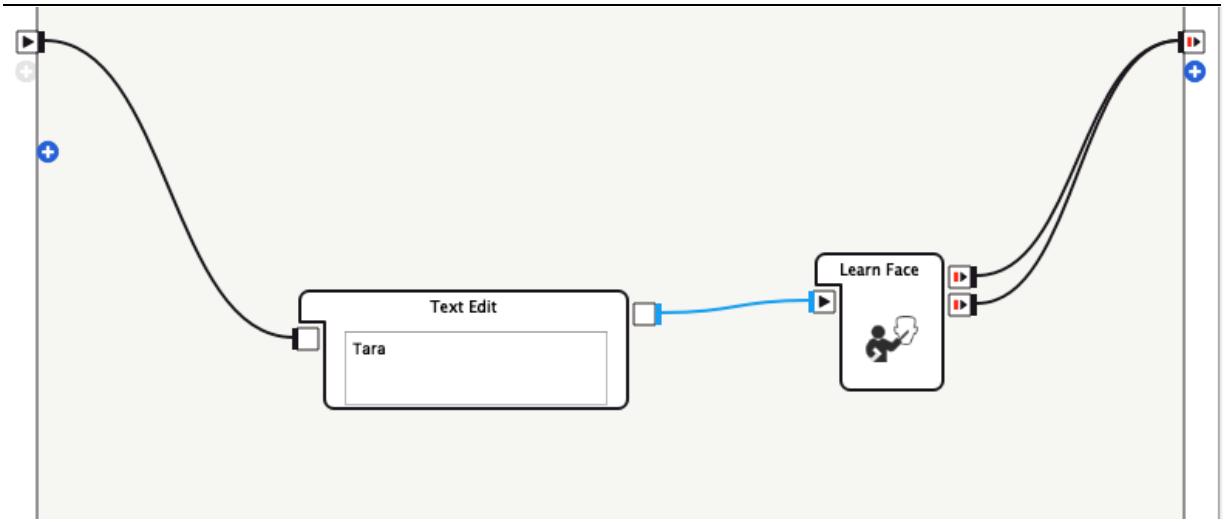
Slika 15 prikazuje program kod kojeg taktilni senzori imaju funkciju. Prilikom dodira bilo kojeg od triju taktilnih senzora glave robota, robot će podići motore iz ugašenog autonomnog stanja tako da mu autonomni sustav ostane ugašen. Razlog zašto se ovaj korak je uveo je kako bi se robot aktivirao tek kada to korisnik želi. A prilikom dodira senzora na lijevoj ili desnoj ruci, robot se stiša i gasi svoje motore te odlazi u pognutu, ravnotežnu poziciju. To je dobar sigurnosni korak i način da se robota isključi bez da se to mora fizički napraviti ako bilo koji dio programa ne pođe po planu i želi se prekinuti rad robota.



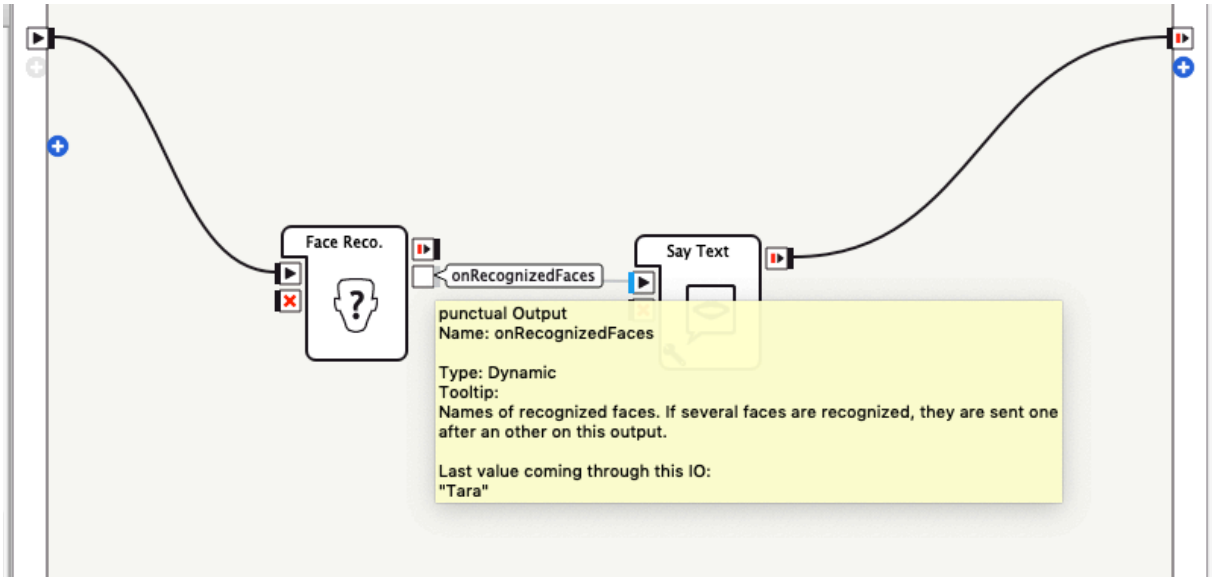
Slika 15. Program primjene taktilnih senzora

3.2. Vizijski sustav

Prvi primjer korištenja *Pepper*-ovog vizijskog sustava je prepoznavanje lica (engl. *Face recognition*). Ono što se mora učiniti prvo je robotu spremiti neko lice u njegovu bazu podataka za vizualno prepoznavanje te paralelno s time uz to lice spremiti i ime te osobe. Nakon što se ime i lice povežu, te se potvrdi uspjeh spremanja lica u bazu podataka, unutar drugog dokumenta pokreće se blok za prepoznavanje lica koji se spaja s blokom izgovori tekst (engl. *Say text*). Svaki put kad se ovaj dokument pokrene, ako robot prepozna neku osobu, reći će njeno ime. Ovaj je program isproban na desetak osoba različitih fizičkih karakteristika i spolova te je u svakom slučaju *Pepper* uspješno zapamtio i prepoznao lice, pa čak i u slučajevima kada se pred njime našlo više ljudi istovremeno. Slika 16 u žutom okviru prikazuje da je program uspješno izveden jer je kao zadnje lice koje je uspješno prepoznato vezano za ime Tara, što je i bio cilj.

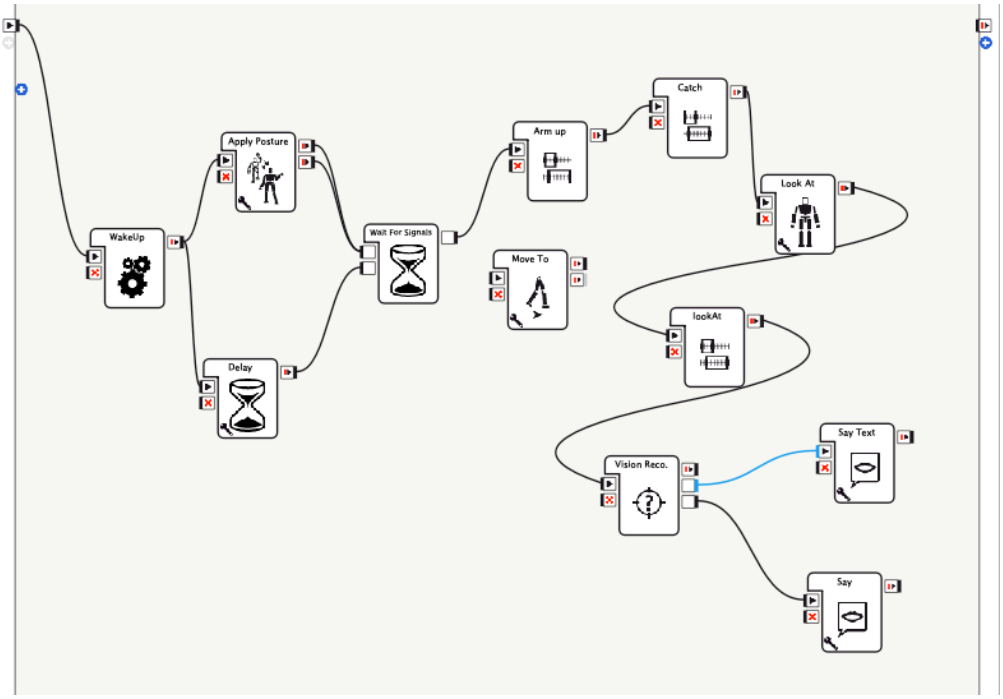


Slika 16. Blok nauči lice



Slika 17. Blok prepoznavanje lica

Prema principu sličnom prepoznavanju lica radi i prepoznavanje objekata (engl. *Object recognition*). Prvo se u bazu podataka za vizualno prepoznavanje spoji predmet i njegov naziv, a zatim se kroz ispituje poznaje li robot stvarno taj objekt (Slika 18). Za prepoznavanje objekata nema automatski gotova opcija kao za lice, nego se treba unositi veliki broj ulaznih podataka kako bi robot nešto naučio. U primjeru koji slijedi robotu je bio zadatak spojiti sustav kretnji i prepoznavanje objekata. Točnije, robot je morao biti sposoban primiti, namjestiti u poziciju da kamera dobro vidi te prepoznati smeđu plišanu igračku ježa (Slika 19). U njegovu bazu spremljeno je 60-ak fotografija ježa u različitim položajima, s različitim po i pod različitim osvjetljenjima (Slika 20). Unatoč velikom broju uzoraka, robot i dalje nije sposoban uvijek prepoznati ježa te bi uspješnost rasla dodavanjem veće količine ulaznih podataka.



Slika 18. Program primanja i prepoznavanja predmeta

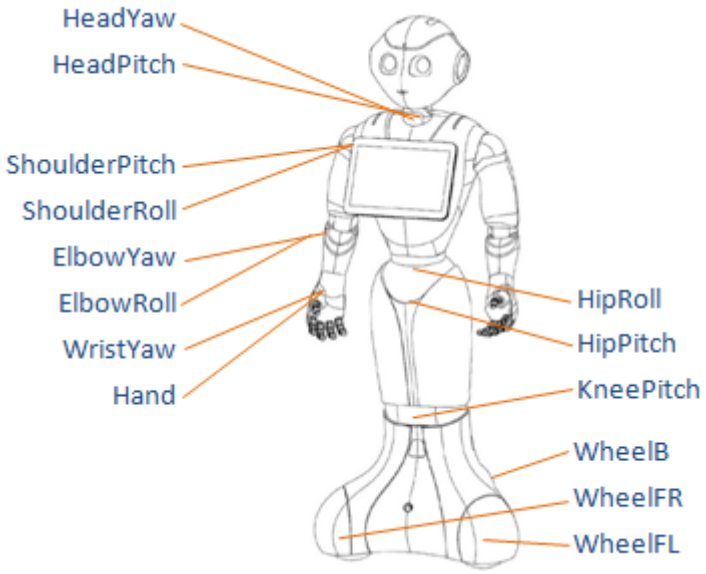
(a) (b)

Slika 19. (a) Primanje objekta (b) Provedba prepoznavanja objekta

Slika 20. Primjeri označavanja fotografija ježa za prepoznavanje objekata

4. SUSTAV KRETANJA

Pepper ima jedan stupanj slobode gibanja u koljenu, po dva stupnja slobode u glavi, ramenu, laktu, ručnom zglobu i kuku te po jedan u svakom kotaču. Ima tri kotača (dva prednja i jedan stražnji). To mu omogućuje veliki opseg kretnji i zauzimanje različitih položaja. Pošto nema noge, nego bazu u obliku trokuta, gotovo ga je nemoguće izbaciti iz ravnotežnog stanja što ga čini vrlo praktičnim za rad.



Slika 21. Lokacija motora [6]

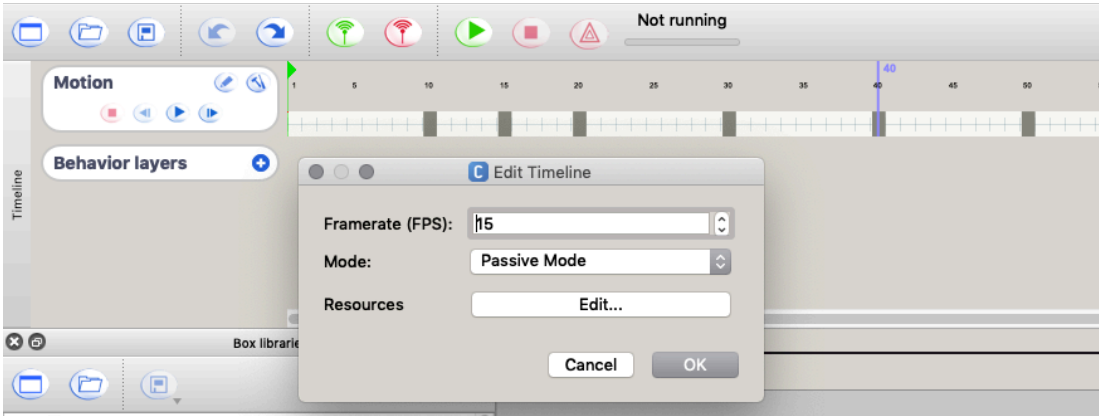
Unutar *Choregraphe*-a postoji više gotovih naredbi vezanih za kretanje, a dijele se na statične i dinamične. One statične su zauzimanje određene pozicije (engl. *Posture*) ili da se usmjeri kamo robot gleda (engl. *Look at*). Dinamične naredbe su najčešće zabavnog karaktera, gdje je cilj da robot zadobije interes korisnika. Primjerice, postoji kategorija životinja (engl. *Animals*) unutar koje se nalaze opcije da se *Pepper* ponaša kao slon, miš ili gorila. Također, postoji kategorija plesovi (engl. *Dances*) koja nudi tri različita plesa s popratnom muzikom.

Osim gotovih opcija, postoje opcije koje se moraju postaviti ručno. One, za razliku od gotovih opcija, osim kretnji vezanih za tijelo robota, dopuštaju i prostorno kretanje. Slijedi opis takvih načina programiranja kretnji i kretanja robota.

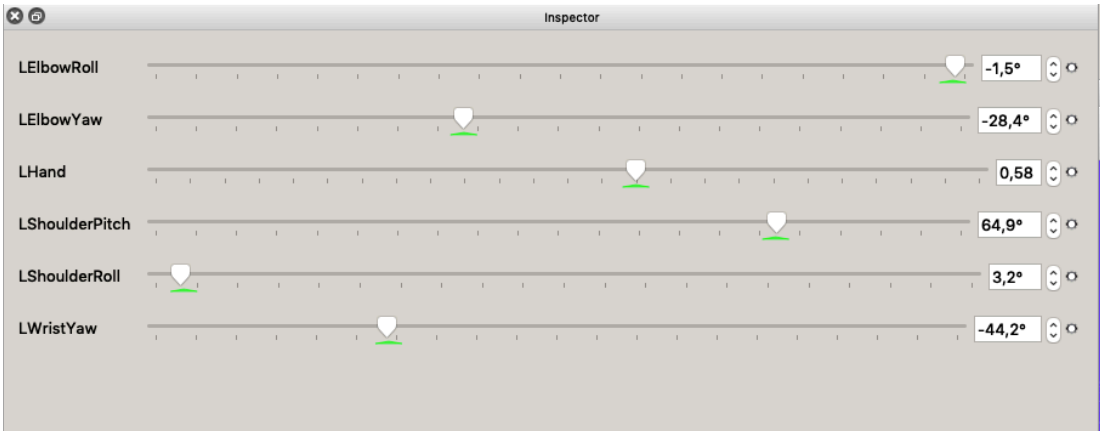
4.1. Blok vremenska linija

Slika 22 prikazuje jednu vremensku liniju u kojoj je napravljeno jednostavno mahanje rukom robota. Svaka siva traka na vremenskoj liniji je jedan položaj koji robot zauzme. Za svaki takav položaj definira se kut ili udaljenost nekog članka (Slika 23). Brzina izvođenja kretnji definira se preko broja kadrova koji se izvode u sekundi (engl. *Frame per second*). Cijela vremenska

linija stavlja se na jednu brzinu, ali ako se neki dijelovi trebaju izvesti brže, između njih se ostavi manje praznih kadrova. Također, bitno je za prepoznati kako će se neke pozicije sporije ostvariti, ovisno o koordinatama, tako da je preporučeno ostaviti prazne kadrove između položaja da robot ima vremena doći u zadane točke. Korisna opcija unutar samog bloka je zrcaljenje (engl. *Mirror*). Ona omogućuje da se sve naredbe kretanja za jednu stranu tijela preslikaju na drugu. Primjerice, vremensku liniju napravljenu za lijevu ruku prebaci i na desnu te ih izvodi istovremeno. To doprinosi velikoj uštedi vremena pošto je stvaranje navedenog bloka vremenski iscrpan proces.



Slika 22. Vremenska linija



Slika 23. Definicija položaja ruke

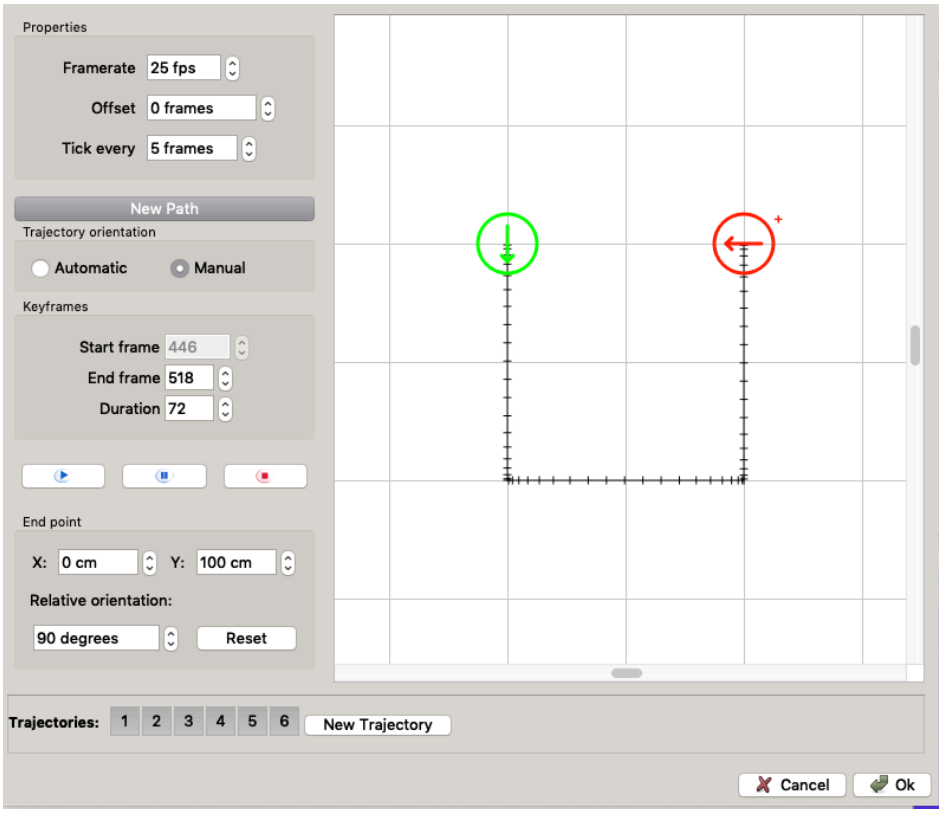
Ukupni broj kadrova također definira i finoću izvedbe kretnji. Težnja je postići što prirodnije i finije pokrete robota pa je poželjno da točke koje se definiraju budu blizu jedna drugoj. Tako se i izbjegava problem vremena potrebnog da robot dođe iz točke u točku jer, ako su blizu, sve pozicije će se ostvariti u relativno jednolikom vremenskom periodu.

4.2. Kretanje u prostoru

Kretanje u prostoru može se realizirati na više načina. Prvi od njih je zadavanje položaja u koji robot mora doći. Preko opcije kretanje prema (engl. *Move to*) zadaje se točka u koju robot mora

doći s obzirom na svoj trenutni položaj. Moguće je zadati x i y koordinate te kut pod kojim se robot kreće. Ishodište koordinatnog sustava je u bazi robota, a promatrajući iz stajališta samog robota, smjer pozitivne x koordinatne osi gleda ravno naprijed, a smjer pozitivne y koordinatne osi gleda prema lijevo. Ovaj način zadavanja kretanja pokazao se netočnim, često robot nije došao u zadanu točku. Također, tako se može zadati pomak samo u jednu točku, što nije praktično.

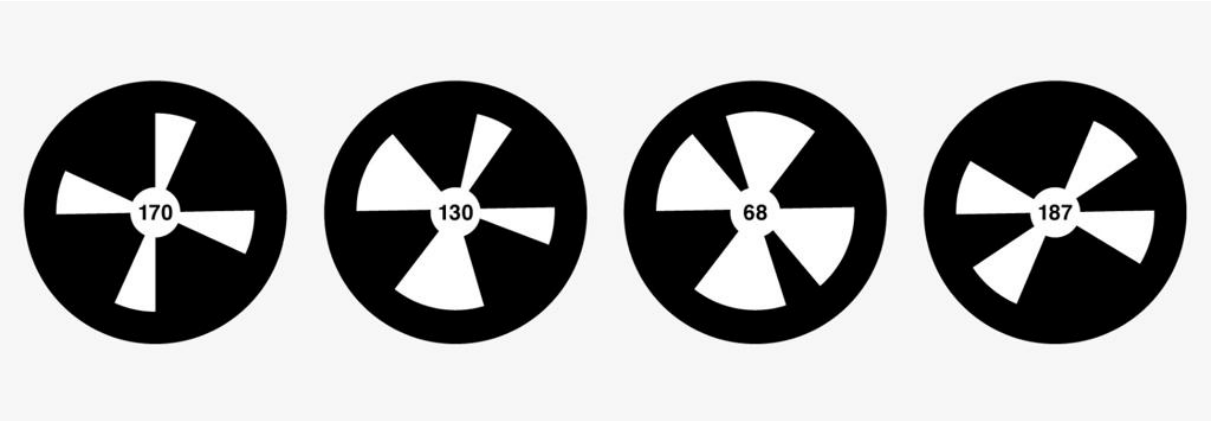
Puno točniji način je zadavanje trajektorija kroz opciju ravninsko kretanje (engl. *Planar move*). Trajektorije se zadaju unutar .pml dokumenta (Slika 24). Zatim se taj dokument učiota u blok (`'Move Along'`) koji zatim izvede to kretanje.Trajektorije se definirane kao putanja u vremenu [7]. Kao i kod vremenskih linija, može se zadati broj kadrova koji su u primjeru vidljivi kao zarezi na liniji. Promatranjem tih zareza uočljivo je da se robot najsporije kreće na početku i kraju, a najbrže na sredini zbog potrebnog vremena za ubrzanje i usporenje kretanja robota. Početni položaj prikazuje se zelenom bojom, a konačni crvenom.



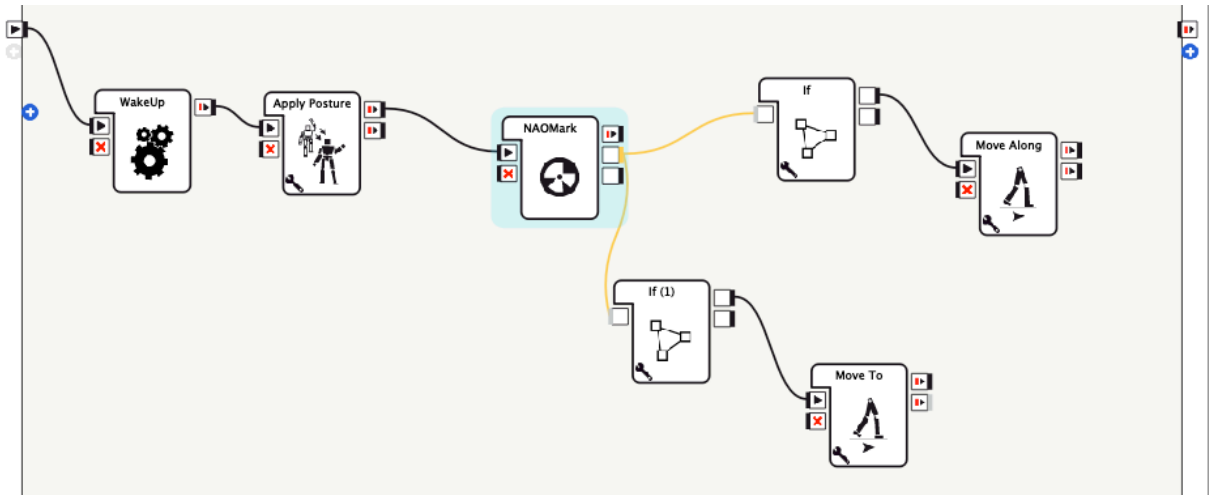
Slika 24. Definiranje trajektorija

Najnapredniji i najtočniji način kretanja u prostoru je pomoću markera. Ideja je da robot pomoću prepoznavanja objekata (engl. *Object recognition*) prepozna o kojem se markeru radi. Svaki marker ima određen identifikacijski broj. Ovisno o tome koji je marker očitao, robot

zatim izvršava radnju zadanu za njega. Slijedi primjer programa u kojem su se koristili *NAOMarks*, gotovi markeri skinuti sa stranice *SoftBank Robotics* [8] (Slika 25). Prva dva bloka koriste se za podizanje robota nakon gašenja autonomnog načina života (više o tome u poglavlju 5. Programiranje). Nakon toga slijedi blok *NAOMark* koji prepoznaje o kojem se markeru radi te je njegov izlaz identifikacijski broj markera. Za ovaj primjer definirana su dva markera te, ovisno o tome koji je očitan, izvršava se jedno od dva različita načina kretanja u prostoru. Korištenje markera nije svedeno samo na kretnje, mogli bi ih uz bilo koji drugi blok koristiti na sličan način, ali je ovo njihova najčešća uporaba. Ukoliko bi se iz nekog razloga htjeli koristiti neki drugi markeri, primjerice popularni *ARUCO* markeri, to ne bi bio problem. Prvo bi ih se trebalo uvesti u bazu podataka za vizualno prepoznavanje (engl. *Vision recognition database*), nakon čega bi se preko bloka vizualno prepoznavanje (engl. *Vision recognition*) dobila informacija o kojem se markeru radi, a svi koraci iza toga bili bi analogni ovima kod *NAOMarker*-a.



Slika 25. Primjeri *NAO* markera [8]

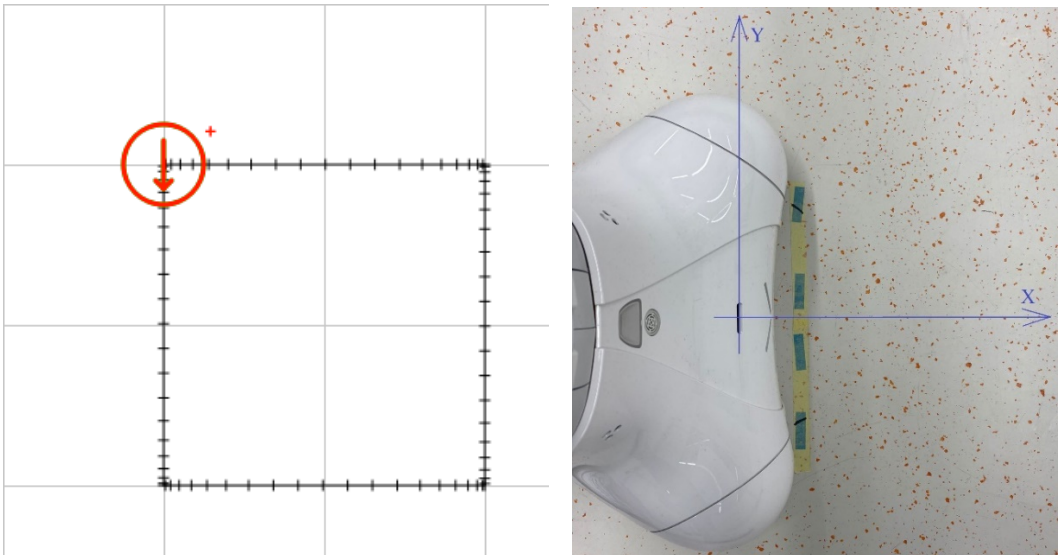


Slika 26. Tijek programa kretanja pomoću *NAO* markera

Napomena vezana za sve načine realizacije kretanja u prostoru je da će robot stati ako se preblizu približi nekoj prepreci ili predmetu kako bi spriječio koliziju i mogući pad te oštećenja. To se dešava uvijek kada njegovi ultrazvučni senzori u bazi detektiraju prepreku ili objekt udaljen 15 cm od baze robota.

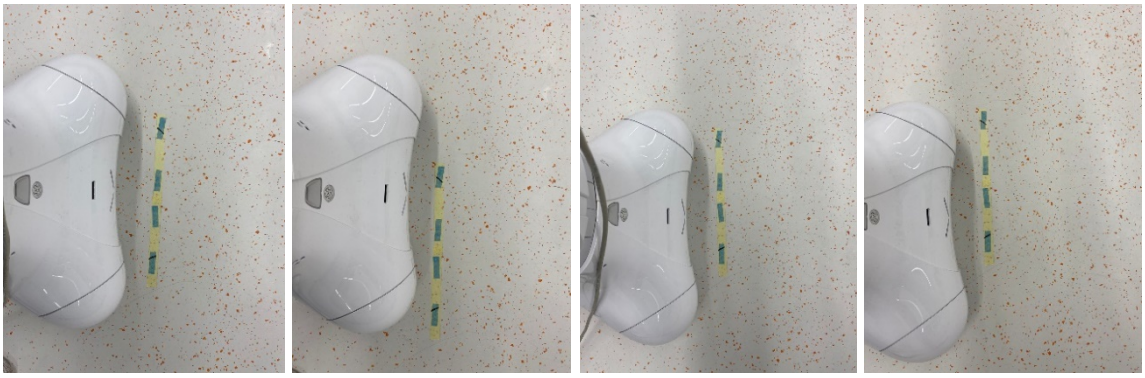
4.3. Validacija kretanja u prostoru

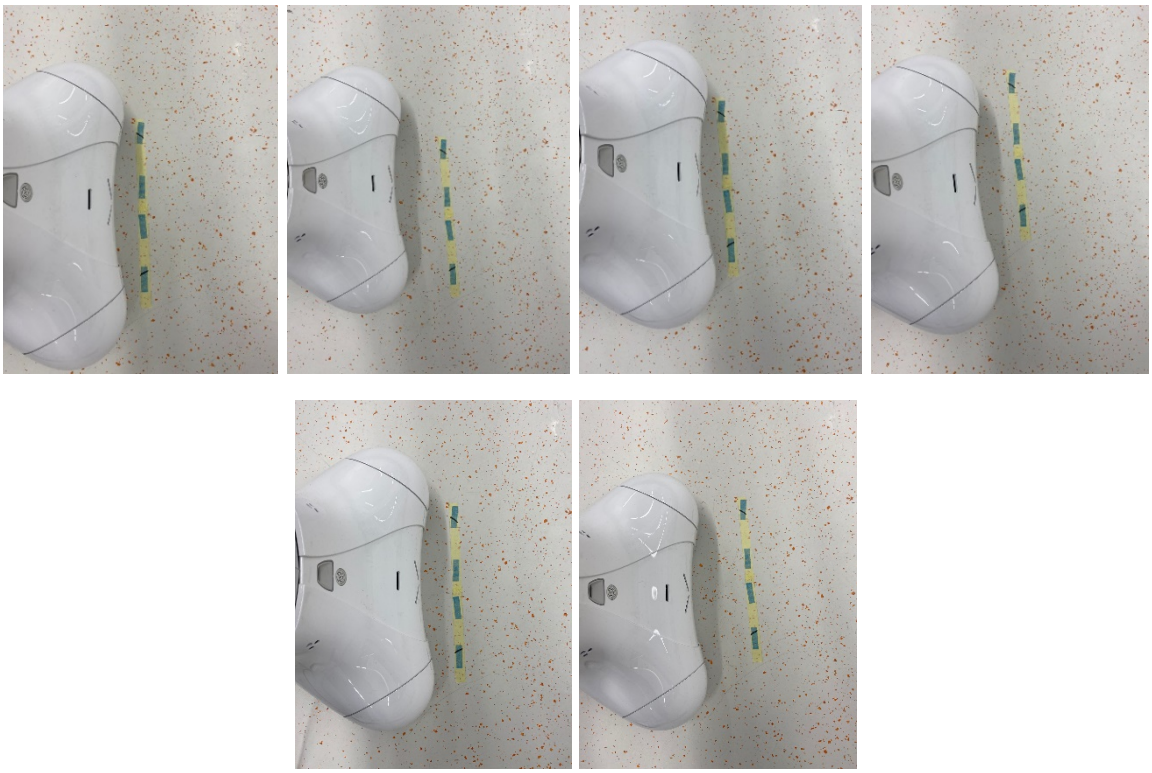
Nakon objašnjenja kretanja u prostoru pomoću definiranja trajektorija ili korištenjem markera, slijedi validacija svakog. Kod ravninskog definiranja trajektorija validacija je provedena na 10 uzastopnih ispitivanja gdje su se promatrala dva parametra. Prvi je pomak po x-osi, a drugi je pomak po y-osi. Zadatak je osmišljen tako da se robot kreće na udaljenostima od jedan metar, te se nakon svakog prijeđenog metra rotira za pravi kut. Nakon što to učini četiri puta, mora se naći u istom položaju te istoj poziciji iz koje je krenuo. Slika 27 prikazuje dijagram ravninskih trajektorija za takav slučaj te početni položaj robota s definiranim koordinatnim sustavom. Nakon toga slijede fotografije i grafički prikaz rezultata.



(a) (b)

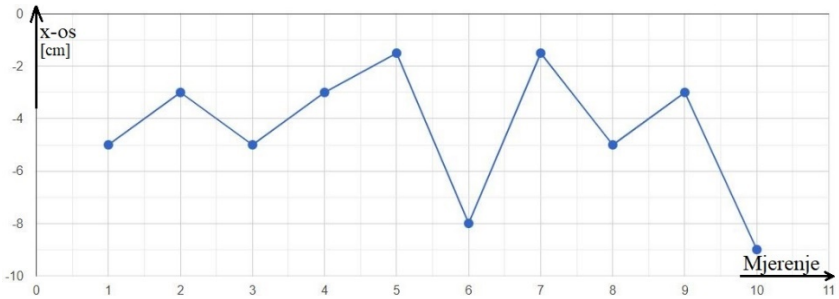
Slika 27. (a) Definirane trajektorije (b) Početni položaj i koordinatni sustav



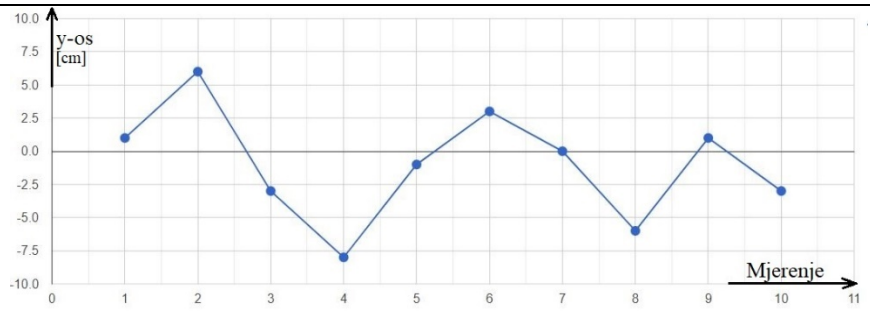


Slika 28. Fotografije mjerenja (po redu od prvog do desetog)

Rezultati mjerenja ukazuju da je korištenje unaprijed definiranih trajektorija loš odabir ako je bitno da se robot točno pozicionira. Iz grafova je vidljivo da opseg greške kod obje osi iznosi $\pm$ 10 cm. Bitno je za napomenuti da su sve greške u smjeru x-osi bile negativne (prema koordinatnom sustavu sa slike 15.)



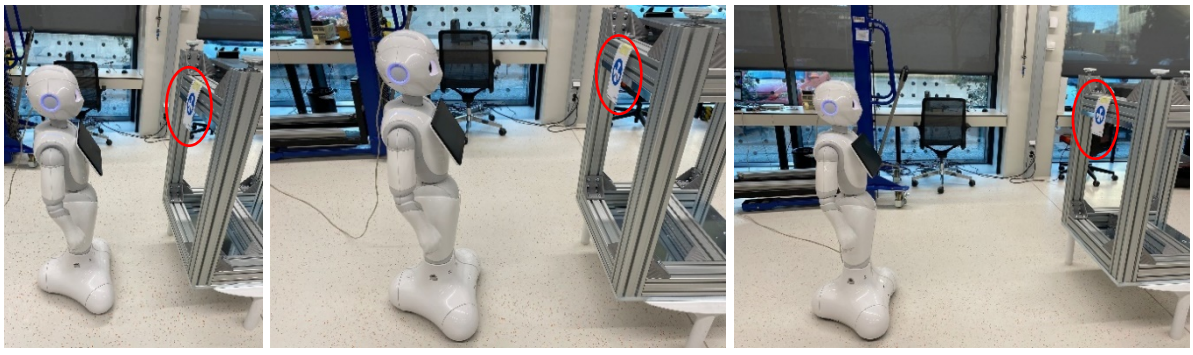
Slika 29. Rezultati mjerenja za pomake po x-osi



Slika 30. Rezultati mjerenja za pomake po y-osi

Proveden je niz ispitivanja prepoznavanja *NAO* markera. Zadatak je bio da robot prepozna marker i priđe mu na 0,3 metra udaljenosti. U opisu će ispitivanja biti predstavljena po serijama. Za svako ispitivanje će biti navedena najveća udaljenost s koje je robot prepoznao marker, a zatim će biti prikazano rješenje pomoću kojeg bi robot ispravljao svoj položaj i orijentaciju sve dok ne prepozna marker i priđe mu. Blok kreni prema (engl. *Move toward*) se koristio za prilazak robota markeru. U njegovim postavkama se odredi kolika je udaljenost na kojoj se robot mora zaustaviti ispred robota. U ovom radu ona je definirana na 20 cm. Na fotografijama će marker, kako bi se lakše uočio, biti zaokružen crvenom bojom. Za sva ispitivanja korišteni su markeri plave boje te promjera 9 cm.

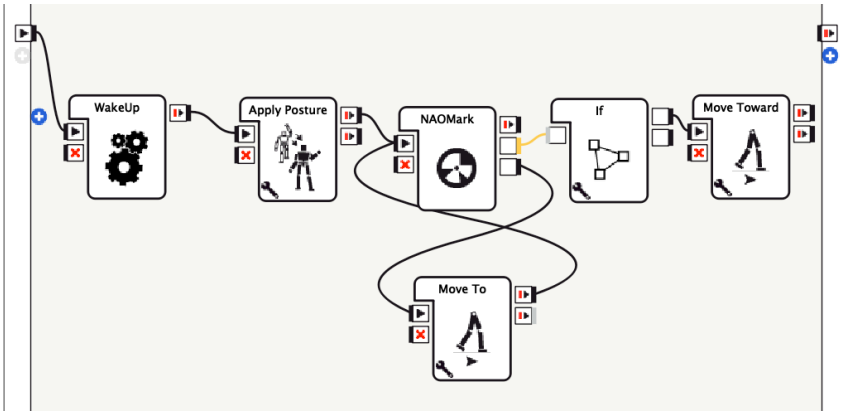
U prvoj seriji ispitivanja, marker je postavljen ravno ispred robota, na razini njegovih očiju. Ispitivanje je izvršeno na udaljenostima: 0,5 m, 1 m, 1,5 m, 2 m i 2,5 m. Nakon 2,5 m ispitivanje je prekinuto jer na toj udaljenosti robot više nije bio sposoban očitati marker. Ovaj problem je riješen tako da je u kod dodan blok kreni za (engl. *Move to*) koji, u slučaju da je robot predaleko i ne uspije očitati marker, pošalje robota za jedan metar naprijed ravno. Ako tada uspije očitati, dalje se koristi blok kreni prema, a ako ne, robot će se opet automatski pomaknuti metar naprijed i tako sve dok ne očita marker.



(a) (b) (c)

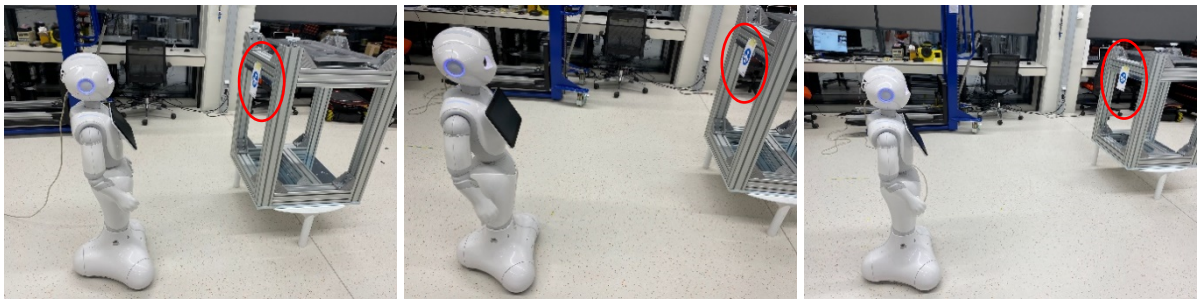
(d) (e)

Slika 31. *Pepper* udaljen od *NAO* markera (a) 0,3 m (b) 0,5 m (c) 1 m (d) 1,5 m (e) 2 m



Slika 32. Rješenje problema prve serije ispitivanja

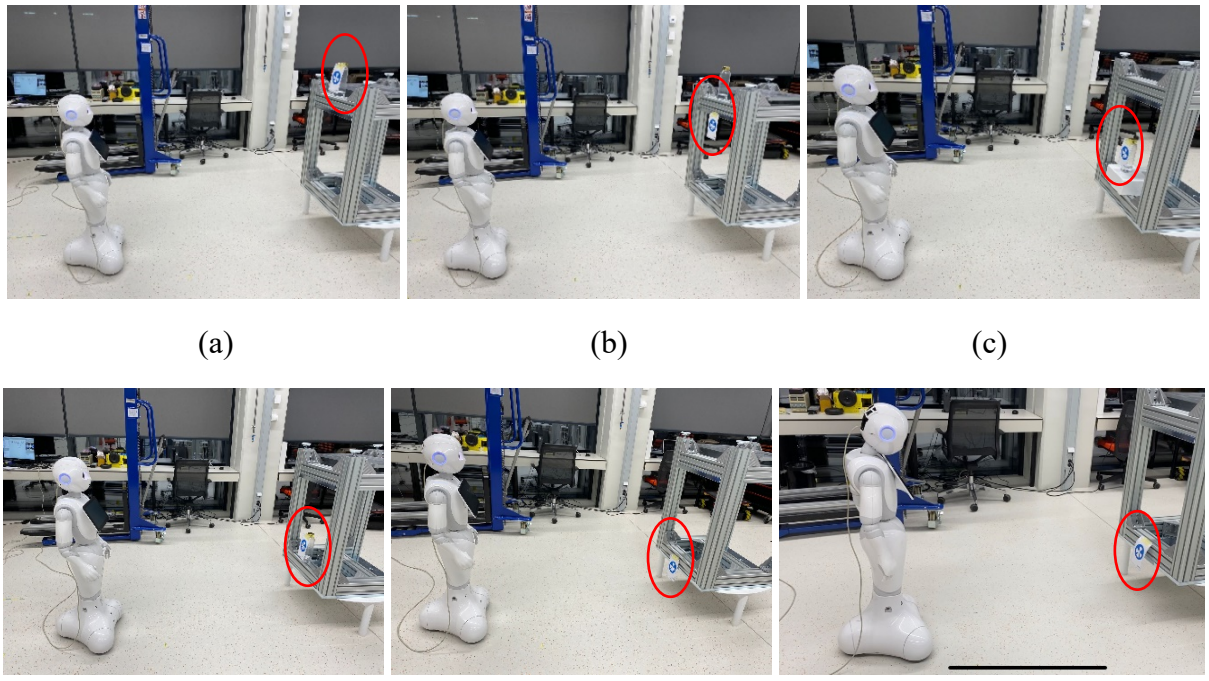
U drugoj seriji ispitivanja robot je stavljen lijevo ili desno od robota. Kako se udaljavao od markera, tako je bio postavljen i na veći kut u odnosu na marker, tako da mu on i dalje ostane u vidnom polju. Već na trećem testiranju, točnije, na 1,5 metra od markera, robot nije mogao prepoznati marker. Ovaj problem također je riješen pomoću bloka kreni za, ali u ovom slučaju robot se nije kretao metar unaprijed, nego se okrenuo za 30 stupnjeva. Dakle, tijek programa isti je kao i u prošlom primjeru (Slika 32), samo je promijenjen parametar koji definira što robot mora promijeniti kako bi uspješno očitao marker. U prvom primjeru je promjena udaljenosti u odnosu na marker (približi mu se), a u drugom je promjena kuta gledanja robota.



(a) (b) (c)

Slika 33. *Pepper* udaljen od *NAO* markera pod nekim kutem (a) 0,5 m (b) 1 m (c) 1,5 m

U trećoj seriji ispitivanja *Pepper* je postavljen ravno u odnosu na marker, na udaljenosti od 1,5 m. Marker je stavljen na pet različitih visina te se pratilo hoće li robot reagirati na sve njih. Udaljenost od 1,5 m od markera se u prvom ispitivanju pokazala dobrom za očitavanje, a dovoljno je daleko da na svim visinama marker ostane u vidnom polju robota. U ovoj seriji ispitivanja problem je riješen uvođenjem bloka pogledaj (eng. *Look at*). Postavke su namještene tako da, kada robot ne prepozna marker, spušta pogled i traži marker.



(a) (b) (c)

(d) (e) (f)

Slika 34. (a, b, c, d, e) *NAO* marker na različitim visinama (f) Robot spušta glavu kako bi bolje identificirao marker

Kombinacijom ova tri rješenja postiglo bi se da je robot sposoban tražiti marker u prostoru i prići mu. Bitno je za napomenuti kako su prilikom ispitivanja često bili problemi s blokom kreni prema. Prvi problem je bio da se nije htio aktivirati, a drugi da nije uvijek prišao markeru, nego je krenuo u nekom nasumičnom pravcu.

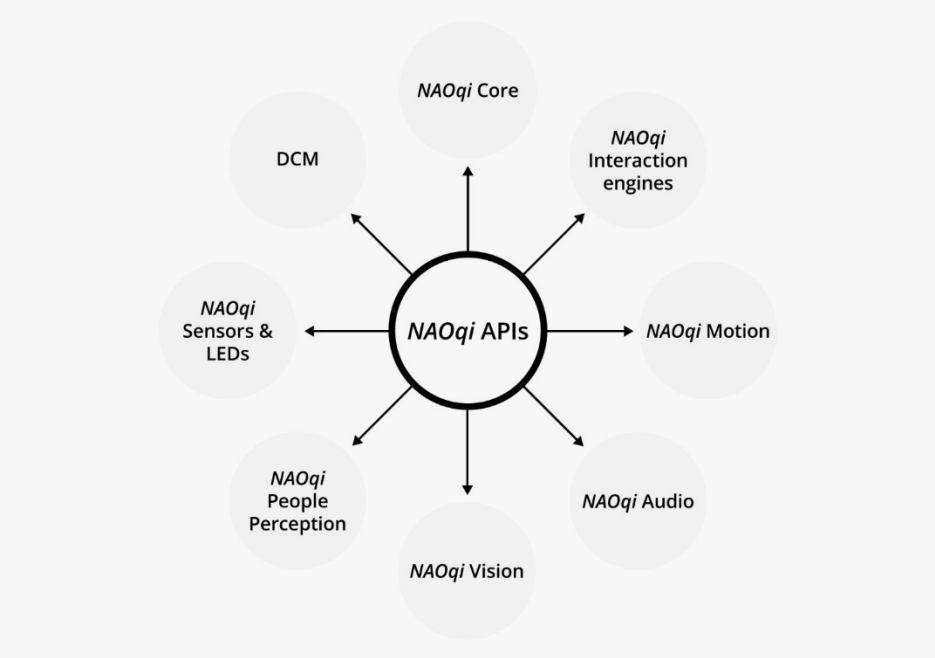
5. PROGRAMIRANJE

Unutar programskog dijela gotovih opcija postoji podjela na vremenske i logičke te na kontrolu ponašanja. Vremenske funkcije su vezane za izvedbu programa u vremenu, primjerice naredba čekaj (engl. *Wait*), naredba odgodi (engl. *Delay*). Logičke funkcije su poznate logičke petlje, pričekaj na signal (engl. *Wait for signals*) te ponovi samo jedom (engl. *Only once*).

Pepper u sebi ima unaprijed programiran autonomni život koji je automatski uključen čim se on pokrene. Autonomni život upravlja signalnim svjetlećim diodama, pozadinskim pokretima, pokretima pri slušanju zvukova te pokretima pri govoru, ali i osnovnom sviješću. Pozadinski pokreti su kada se robot lagano njiše, a pokreti pri slušanju zvukova i govoru su izvođenje nekih gesti (klimanje glavom, odmahivanje glavom, sugestije rukama i slično), a njima oponaša ljudsko ponašanje. Osnovna svijest označava praćenje zvukova i pokreta u okolini robota (ako *Pepper* nešto čuje, orijentirat će se prema izvoru zvuka te krenuti prema njemu). Autonomni život doprinosi tome da se *Pepper* doživi kao živo biće i da je sposoban uspostaviti interakciju s istim. Robot ima dva načina rada, prvi u kojem je autonomni život uključen i drugi, u kojem je on isključen. Uključen autonomni život zvuči najčešće ometa izvođenje programa jer ga nadjačava (engl. *Override*). Može se desiti da je cilj programa razgovor između korisnika i robota, te da usred razgovora robot čuje nešto u pozadini i svoju pažnju u potpunosti skrene na tu buku što je nepoželjno. Dakle, prilikom izvedbe programa cilj je ugasiti autonomni život što se postiže aktivacijom opcije autonomne sposobnosti. Opcija za isključenje autonomnih sposobnosti je definirana unutar bloka autonomne sposobnosti (engl. *Autonomous abilities*). Autonomne sposobnosti se unutar jednog programa mogu aktivirati i deaktivirati proizvoljan broj puta. Međutim, to je dovoljno učiniti ako je program spremljen na robotu, ali u slučaju da je robot spojen na kompjuter, tada se autonomni život mora isključiti direktno unutar *Choregraphe-a*. Tu se javlja novi problem jer u tom slučaju robot motore stavlja u stanje mirovanja te zauzima pognut položaj. Kako bi mogao izvesti kretnje, robotu motori moraju biti aktivni, pa ih se pokreće blokom probudi se (engl. *Wake up*). Nakon toga, preporuka je aktivirati blok postavljanja neke od ravnotežnih, gotovih poza kako bi se osiguralo da je sve na željenoj poziciji i da se može uspješno nastaviti s radom.

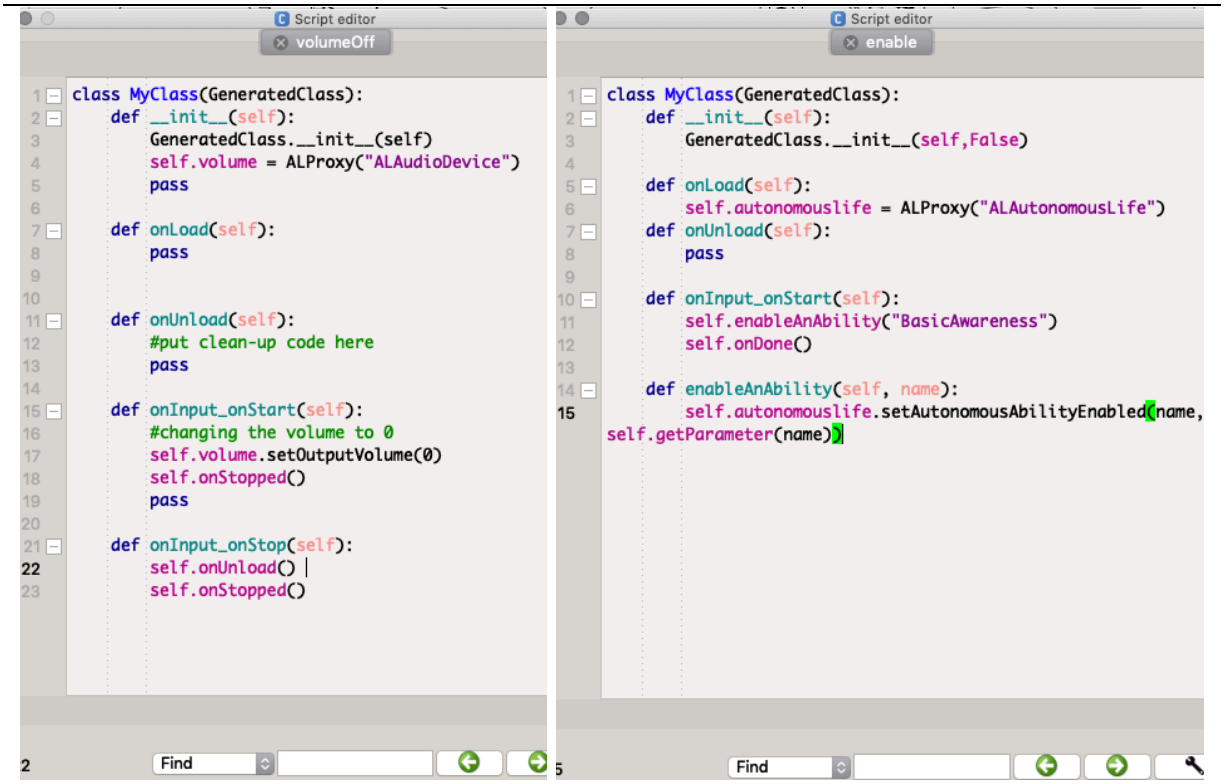
Postoji i mogućnost proizvoljnog programiranja unutar *Choregraphe-a*. Softver koji radi na robotu te njime upravlja zove se *NAOqi*. *NAOqi* omogućava paralelni rad različitih sustava, sinkronizaciju te podizanje događaja. On se može programirati koristeći programske jezike *C++* i *Python*. Sučelje za programiranje aplikacija (engl. *Application programming interface, API*) sastoji se od niza metoda svrstanih u module koji su sastavni dio softvera te su uvijek

dostupni. Slijedi grafički prikaz podjele modula sučelja za programiranje aplikacija (Slika 35). Bitno je napomenuti da se sve metode mogu pronaći na službenim *SoftBanks* stranicama te da su tamo upisani načini njihovog korištenja, ulazni argumenti i slično [10].



Slika 35. Podjela modula

Slijede dva primjera gdje je samostalno pisan kod. Prvi je za program pod oznakom „a“ (Slika 36), blok pod nazivom isključi zvuk (engl. *Volume off*). A drugi je za program u kojem je cilj bio isključiti samo jedan aspekt autonomnog života, točnije osnovnu svijest. Kada se programira blok, on je nezavisna jedinica i na njega se odnosi samo ono što je u njemu napisano. Iz tog razloga je moguće obje klase nazvati istim imenom (`'My class'`) i koristiti ih u istom programu bez da dolazi do problema. Ne pozivaju se klase izravno nego blokovi unutar kojih su one definirane. Svaki blok je aktivan sve dok se klasa unutar njega ne izvrši, a tada se aktivira drugi blok dok prvi postaje neaktivan. U prvom primjeru koristi se metoda podesi glasnoću (engl. *Set output volume*). Ona je dio modula *NAOqi Audio*. Pri definiranju odabire se jedan ulazni argument, a definira se kao broj između nula i sto.

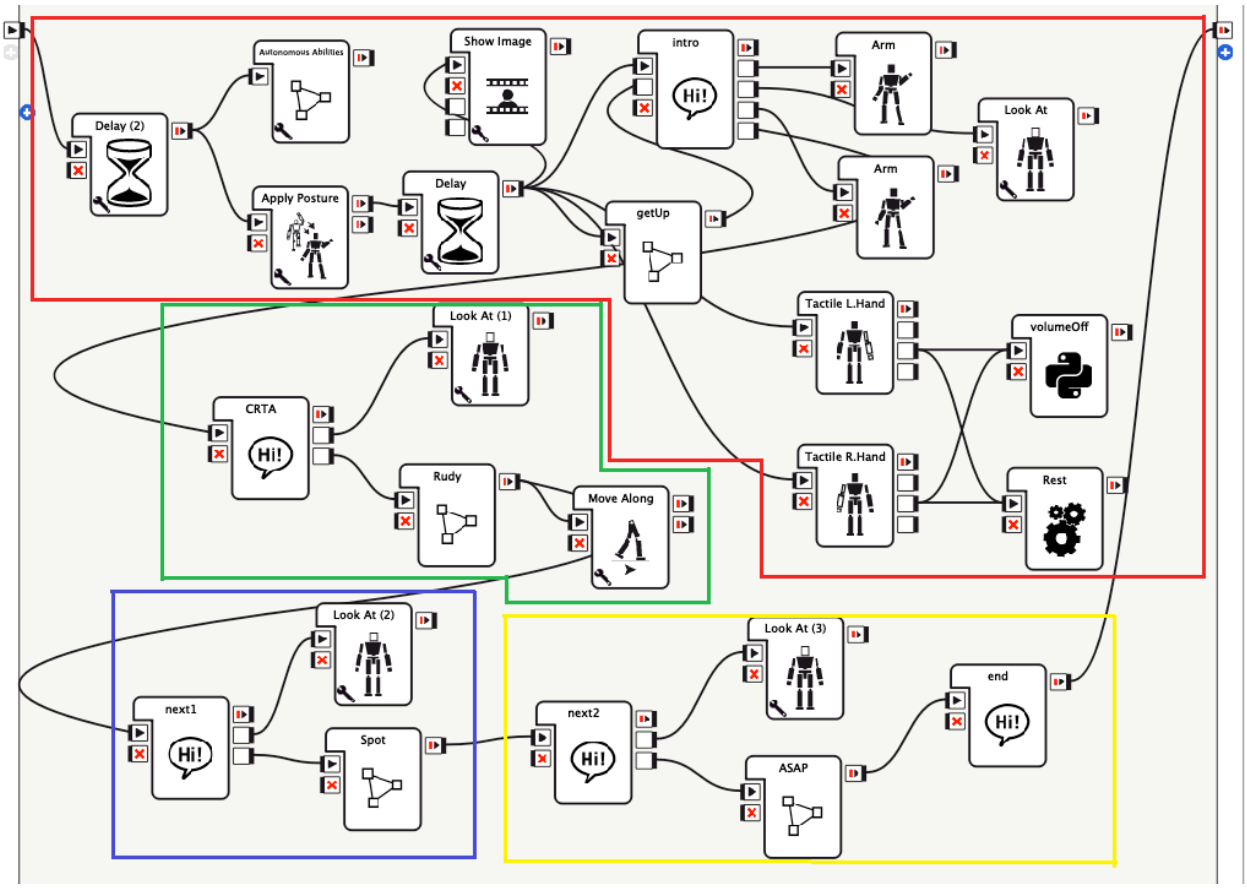


(a) (b)

Slika 36. (a) Kod za isključenje zvuka robota (b) Kod za isključenje osnovne svijesti robota

6. VALIDACIJA CJELOVITOG SCENARIJA INTERAKCIJE ČOVJEKA I ROBOTA

Kao što je spomenuto u uvodu, u primjeru koji će biti sinteza svih sustava i mogućnosti robota istraženih u ovom radu, *Pepper* će imati funkciju vodiča kroz Laboratorij za računalnu inteligenciju. Program je podijeljen na četiri dijela (Slika 37).

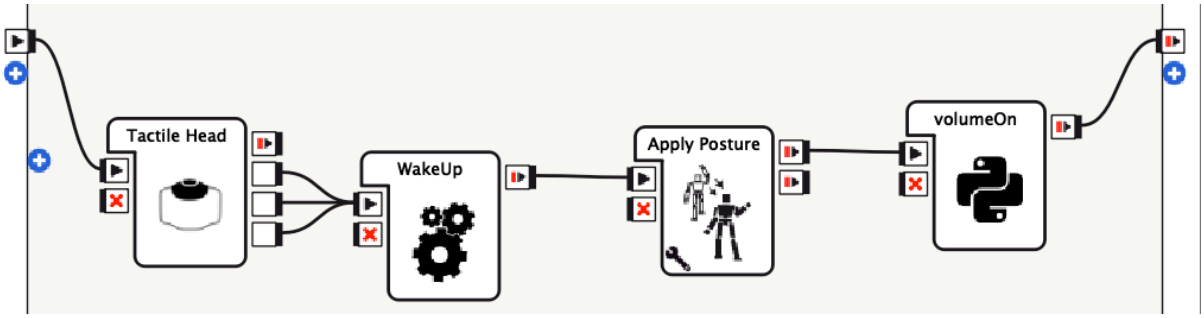


Slika 37. Program cjelovitog scenarija interakcije čovjeka i robota

6.1. Aktivacija programa i uvodni razgovor

Prvi, crveni dio programa (Slika 37) na samom početku isključuje blok za autonomne sposobnosti robota (`'Autonomous Abilities'`). Nakon toga, postavlja se uspravni stav robota (`'Apply Posture'`) koristeći opciju stani (engl. *Stand*). Preporučeno je stavljati tu opciju na svaki početak programa jer može ukazati na grešku u sustavu. Naime, ako robot ne zauzme taj položaj, postoji problem u nekom dijelu njegovog sustava te je tada najbolje ponovno podići cijeli sustav. Na dva mjesta vremenski se odgađa izvršenje programa na par sekundi (`'Delay'`) kako bi se robotu dalo vremena da na samom početku podigne sve svoje funkcije. Nakon drugog odgađanja istovremeno se aktivira pet blokova. Prvi se koristi kako bi se na tabletu prikazala željena slika (`'Show Image'`). U ovom slučaju, na tabletu robota prikaže se logotip CRTA-e. Zatim, aktiviraju se blokovi za očitavanje informacija s taktilnog senzora

obje ruke ('Tactile Left Hand', `'Tactile Right Hand'`). Ti blokovi su dalje spojeni na blok za gašenje motora (`'Rest'`) te na blok unutar kojeg je definirano stišavanje robota (Slika 36, (a)). Ovaj dio programa napravljen kao sigurnosna opcija. U slučaju bilo kakve greške u bilo kojem trenutku izvođenja programa, ako se odmah treba zaustaviti robota, samo se dotakne taktilni senzor na njegovoj ruci (nije bitno kojoj, promjene stanja senzora se prate na obje). Tada se motori gase i robot zauzima ravnotežni položaj. U prvim verzijama programa je, unatoč gašenju motora, robot ponekada znao nastaviti govoriti. Kako bi se to spriječilo, uvedena je opcija potpunog stišavanja robota. Četvrti blok paralelno aktiviran je blok koji omogućuje ponovnu aktivaciju robota nakon gašenja motora i stišavanja. Kako bi program bio pregledniji, neke su stavke grupirane i definirane unutar dijagram bloka (engl. *Diagram box*). Jedna od njih je upravo taj blok (`'getUp'`), a on se zapravo sastoji od bloka koji prima informacije s taktilnih senzora na glavi, bloka koji ponovno podiže motore, bloka za zauzimanje položaja robota te bloka koji poveća glasnoću robota (Slika 38). Blok za povećavanje glasnoće definiran je isto kao i onaj za stišavanje (Slika 36, (a)), ali je ulazni argument obavezno veći od nule. U ovom primjeru, glasnoća je postavljena na 70%.



Slika 38. Dijagram blok za ponovno pokretanje robota

Zadnji dio koji je paralelno aktiviran je dijalog blok (Slika 39). Pri definiciji bloka postavljen je proizvoljan ulaz (`'inputHead'`), kako bi se nakon ponovne aktivacije robota, ako je došlo do gašenja motora i stišavanja, konverzacija pokrenula ispočetka. Također, dodana su četiri proizvoljna izlaza, (`'outputWelcome'`, `'outputQuestion'`) koji aktiviraju blok za pomicanje ruku, (`'outputLookAt'`) kako bi robot pogledao na visinu očiju korisnika te (`'outputRobots'`) kako bi se razgovor prebacio na drugi dijalog blok nakon izvršenja ovog. Unutar ovog dijalog bloka robot se aktivira očitanjem promjene pozicije korisnika ili taktilnim senzorima na glavi. Nakon aktivacije robota on kratko pozdravlja korisnika, pita ga kako je i kako se zove.

```
concept: (good) {"I am"} {"I'm"} {feeling} {doing} {pretty} [good excellent outstanding "all right" great okay ok fine
"not bad" well awesome] {"Thank you"} {Thanks} {"How are you"} {"How about you"} {feeling} {doing} {you}
concept: (bad) {"I am"} {"I'm"} {feeling} {doing} {pretty} [bad terrible horrible awful "not [great good well]"] {"Thank
you"} {Thanks} {"How are you"} {"How about you"} {feeling} {doing} {you}
concept: (hello1) [hi hello hey "hi there" greetings "good day"] {Pero} {Pepper}
concept: (hello2) [hi hello "hi there" greetings]
concept: (yes) [yes certainly definitely "of course" gladly indeed yeah yup sure "you bet" totally alright] {"I would"}
{like} {that} {to} {meet}
concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" ] {sorry} {I} {do not} {don't}
{want} {to} {wanna}

#u:(~hello1) ~hello2 \pau=500\ ^gotoReactivate(WELCOME)
u:(e:EngagementZones/PersonApproached) ~hello2 \pau=500\  ^gotoReactivate(WELCOME)
u:(e:inputHead) ~hello2 \pau=500\ ^gotoReactivate(WELCOME)


proposal: %WELCOME $outputWelcome=1  Welcome to the regional center of excellence for robotic techology. \pau=250\
$outputLookAt=1 \rst=100\ \rspd=75\ How are you doing? $outputLookAt=1
    u1:(~good) \pau=500\ Great!^gotoReactivate(NAME)
    u1:(~bad) \pau=250\ I am so sorry to hear that!^gotoReactivate(NAME)

proposal: %NAME \pau=500\ Can you tell me your name? $outputQuestion=1 $outputLookAt=1
    u1:(e:Dialog/NotUnderstood) \pau=1000\ I like that name! My name is \rspd=75\ Pero\pau=250\ $outputRobots=1
```

Slika 39. Program unutar dijalog bloka intro

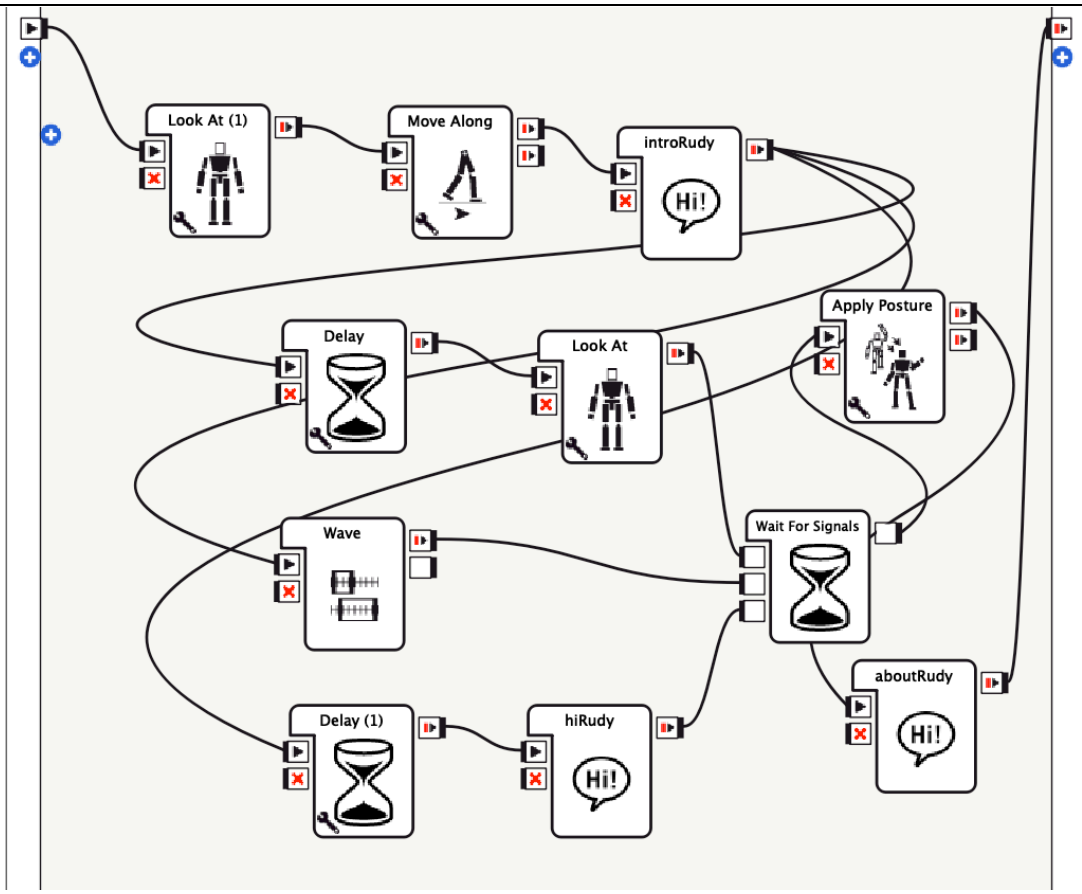
6.2. Predstavljanje laboratorija i prvi robot

Po redoslijedu drugi dio programa (Slika 37) kreće aktivacijom dijalog bloka (Slika 40). Aktivira se dolaskom signala na ulaz bloka (nakon što se u dijalog bloku iz prvog dijela aktivirao izlaz „outputRobots“). Ima dva izlaza, prvi na samom početku aktivira pomak pogleda robota na visinu očiju korisnike, a drugi na kraju aktivira dijagram blok „Rudy“.

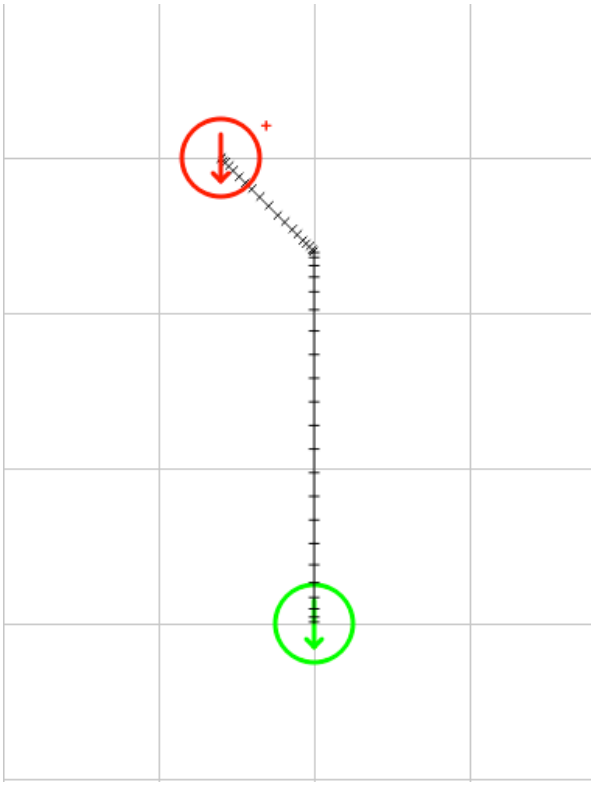
```
concept: (yes) [yes certainly definitely "of course" gladly indeed yeah yup sure "you bet" totally
alright] {"I would"} {like} {love} {that} {to} {meet} {please}
concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" ] {sorry}

u:(e:onStart) $outputLookAt=1 This is labaratory of computer intelligence.\pau=250\ It is home for my
friends and I.\pau=450\ I would like to show you one of them. $outputRudy=1 \pau=350\
```

Slika 40. Program unutar dijalog bloka CRTA

Aktiviran dijagram blok (Slika 41) na početku postavlja razinu pogleda robota (`'Look at'`). Zatim slijedi blok u kojem je definirano planarno prostorno gibanje kako bi *Pepper* zauzeo poziciju kraj prvog robota kojeg predstavlja (Slika 42). U ovom primjeru odabrano je korištenje unaprijed definiranih trajektorija, a ne snalaženje pomoću *NAO* markera. Razlog je to što bi markeri za svoje funkcioniranje zahtijevali točno određenu startnu poziciju i orijentaciju. Također, ova kretanja kroz laboratorij sadrže puno promjena orijentacije robota kako bi njegovo kretanje izgledalo prirodno kada hoda kroz njega, ali i da ne priča korisniku okrenut mu leđima. Česte i velike promjene orijentacije robota zahtijevale bi korištenje velikog broja markera te je definicija trajektorija bila bolji izbor kako bi program bio manje kompliciran. Validacijom kretanja pomoću definiranja trajektorija ocijenjene su određene greške, ali one su za ovaj primjer prihvatljive.



Slika 41. Dijagram blok *Rudy*



Slika 42. Plan trajektorija za planarno gibanje do *Rudy*-ja

Prvi robot kojeg *Pepper* predstavlja je *Rudy*. Prvo predstavi njegovo ime, zatim se aktivira blok vremenske linije te mahne *Rudy*-ju. Na kraju, aktivira se dijalog blok (Slika 43) unutar kojeg se nalaze osnovne informacije u *Rudy*-ju, koji je zapravo robot *iCub*.

```
4  u:(e:onStart) \rspd=85\ Rudy is a humanoid robot, just like me. \pau=500\ Rudy is from
   Italy. \pau=500\ It's main goal is research into human cognition and artificial
   intelligence. \pau=750\ And what a cool blue suit, don't you think so? \pau=500\
   $onStopped=1
5
```

Slika 43. Dijalog blok *Rudy*

Na kraju, izlaz iz dijagram bloka Rudy aktivira treći dio programa.

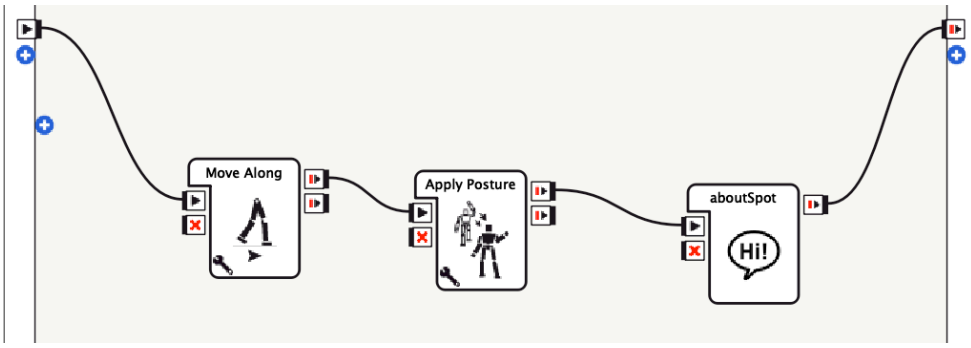
6.3. Predstavljanje drugog robota

Nakon aktiviranja trećeg, plavog dijela programa, prvo se aktivira dijalog blok (Slika 44). Unutar njega korisniku se nudi mogućnost da nastavi obilazak laboratorija ili da ga prekine. Također, postoji i opcija da se obilazak, ako se korisnik u bilo kojem trenu predomisli, nastavi.

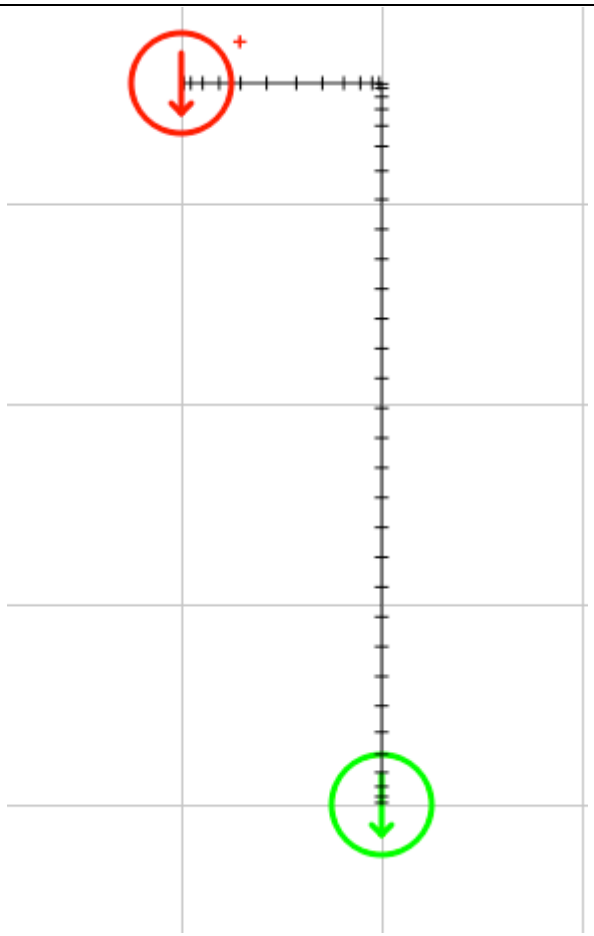
```
4  concept: (yes) {I} {changed} {my} [ mind yes ok certainly definitely "of course" gladly indeed yeah yup
   sure "you bet" totally alright] {"I would"} {like} {that} {to} {meet} {see} {something} {cool} {show}
   {me}
5  concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" "not really"]
   {sorry} {I} {do not} {don't} {want} {to} {wanna}
6
7  u:(e:onStart) \pau=500\ $outputLookAt=1 \rspd=85\ \pau=250\ Did you like the tour so far?
8      u1: (~yes) \pau=500\ Great! Would you like me to show you more?
9          u2: (~yes) \pau=500\ ^gotoReactivate(SPOT)
10         u2:(~no) \pau=500\ ok! \pau=250\ I'll be right here, tell me if you change your mind.
   ^stayInScope
11     u1:(~no) \pau=500\ I am really sorry. Would you like me to move on?
12         u2: (~yes) \pau=500\ ^gotoReactivate(SPOT)
13         u2:(~no) \pau=500\ ok! \pau=250\ I'll be right here, tell me if you change your mind.
   ^stayInScope
14
15
16 proposal: %SPOT Awesome! \pau=250\ Four legged robot \pau=150\ Spot \pau=100\ should be right around
   corner. $outputSpot=1
```

Slika 44. Dijalog blok odluke o nastavku obilaska, *Spot*

Čim se robotu da do znanja da se želi nastaviti s obilaskom, aktivira se dijagram blok (Slika 45). On se sastoji od planarnog prostornog kretanja (Slika 47) prema drugom robotu, Spotu te kratkom opisu njega (Slika 47).



Slika 45. Dijagram blok *Spot*



Slika 46. Plan trajektorija za planarno gibanje do *Spot*-a

```
4   u:(e:onStart) \rspd=85\ \pau=150\ Spot \pau=100\ was made by Boston Dynamics in America. \pau=500\ It
    can map its environment \pau=200\ sense and avoid obstacles \pau=200\ climb stairs \pau=200\ open doors.
    \pau=150\ Pretty impressive \pau=250\ right?$onStopped=1
5
```

Slika 47. Dijalog blok *Spot*

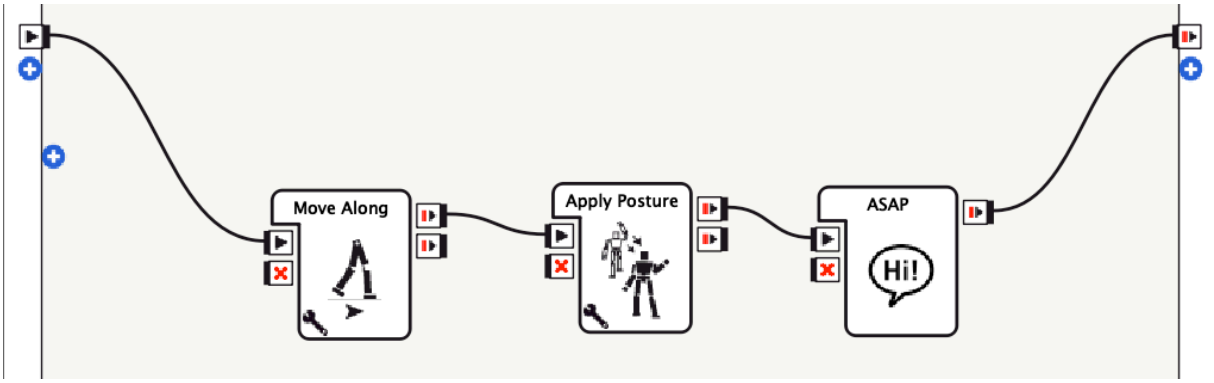
6.4. Predstavljanje trećeg robota

Zadnji dio programa sastoji se od ponovnog pitanja za korisnika vezanog za odabir nastavka ili zaustavljanja obilaska (Slika 48).

```
concept: (yes) {I} {changed} {my} [ mind yes ok certainly definitely "of course" gladly indeed yeah yup
sure "you bet" totally alright] {"I would"} {like} {that} {to} {meet} {see} {something} {cool} {show}
{me}
concept: (no) {"I am"} {I'm} {sorry} {but} [no nah nope "no way" "not now" "no thanks" "not really"]
{sorry} {I} {do not} {don't} {want} {to} {wanna}

u:(e:onStart) $outputLookAt=1 \rspd=85\ \pau=250\ Did you like the tour so far?
    u1: (~yes) \pau=500\ Great! Would you like me to show you more?
        u2: (~yes) \pau=500\ ^gotoReactivate(SPOT)
        u2:(~no) \pau=500\ ok! \pau=250\ I'll be right here,\pau=350\ tell me if you change your mind.
^stayInScope
    u1:(~no) \pau=500\ I am really sorry.\pau=350\ Would you like me to move on?
        u2: (~yes) \pau=500\ ^gotoReactivate(SPOT)
        u2:(~no) \pau=500\ ok! \pau=250\ I'll be right here, tell me if you change your mind.
^stayInScope


proposal: %SPOT Great! \pau=250\ Follow me and I will show you \pau=100\ a cool robot in ASAP project.
\pau=250\ It is a custom \pau=100\ wall-climbing robot $outputASAP=1
```

Slika 48. Dijalog blok odluke o nastavku obilaska, *ASAP*

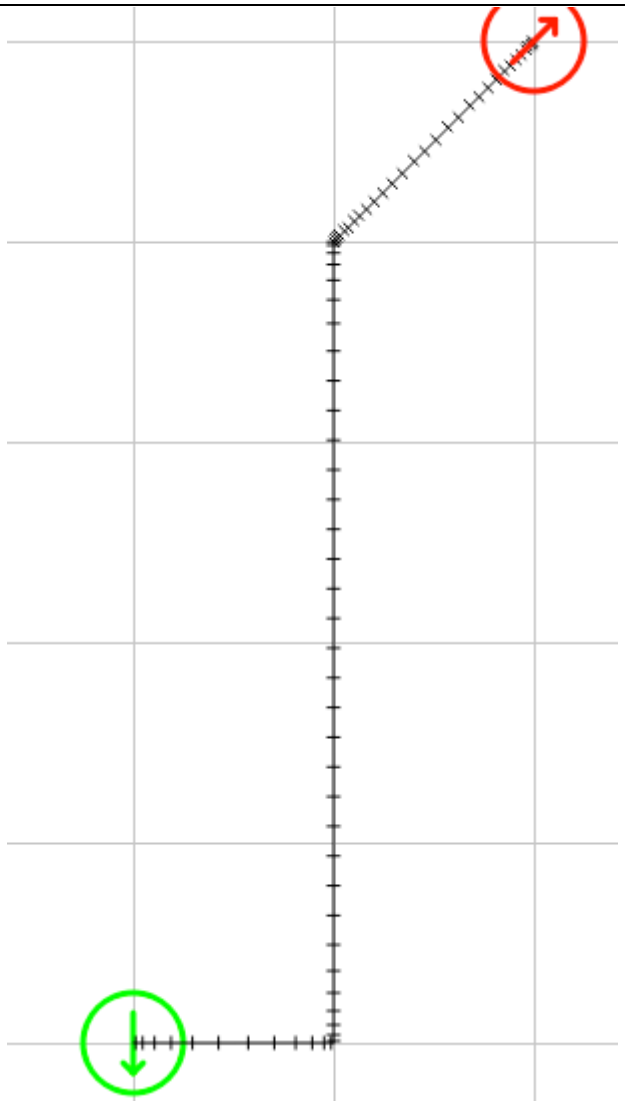
Nakon toga, aktivira se dijagram blok (Slika 49). Unutar njega definirane su informacije o trajektorijama koje *Pepper* mora proći kako bi se približio tom robotu (Slika 51) te informacije o robotu unutar dijalog bloka (Slika 50).



Slika 49. Dijagram blok *ASAP*

```
concept: (repeat) {can} {you} {please} [repeat say] {info} {information} {again} {about}

u:(e:onStart) \rspd=85\ \pau=250\ Project ASAP is an interdisciplinary project between \pau=150\ the
Faculty of Mechanical Engineering and Naval Architecture \pau=200\ Civil engineering \pau=150\ and
Electrical engineering. \pau=500\ The scope of the project ASAP is the development of a robotic system
for autonomous inspection of vertical concrete infrastructure \pau=500\ The system consists of \pau=150\
a wall climbing robot \pau=150\ unmanned aerial vehicle \pau=150\ and smart data collection system.
\pau=250\ $onStopped=1

u:(~repeat) \pau=500\ Gladly! \pau=500\ The scope of the project ASAP is the development of a robotic
system for autonomous inspection of vertical concrete infrastructure \pau=500\
u:(~repeat what system consist of) \pau=500\ Gladly! The system consists of \pau=150\ a wall climbing
robot \pau=150\ unmanned aerial vehicle \pau=150\ and smart data collection system. \pau=250\
u:(~repeat which faculties are included in the project) \pau=500\ Gladly! \pau=250\ Project ASAP is an
interdisciplinary project between \pau=150\ the Faculty of Mechanical Engineering and Naval Architecture
\pau=200\ Civil engineering \pau=150\ and Electrical engineering. \pau=500\
```

Slika 50. Dijalog blok *ASAP*



Slika 51. Plan trajektorija za planarno gibanje do *ASAP*-a

Na samom kraju, aktivira se dijalog blok u kojem *Pepper* pozdravi korisnika (Slika 52).

```
4   u:(e:onStart) \pau=500\ This is the end of the tour! \pau=500\ Thank you for coming! \pau=250\ I hope I
    will se you again $onStopped=1
```

Slika 52. Dijalog blok za kraj obilaska

7. ZAKLJUČAK

U završnom radu prikazano je programiranje humanoidnog robota *Pepper*-a koristeći programiranje blokovima. Ovakav način programiranja daje dobru preglednost rada i lakše snalaženje unutar njega. Pri izradi tih blokova, koristilo se programiranje u *QiChat*-u, *Python*-u te definiranje trajektorija. Uz programiranje, koristile su se mehaničke, senzorske i upravljačke komponente robota.

Prvo je po sustavima prikazano istraženo i naučeno, ističući kod svakog ono najbitnije i najčešće korišteno. Neke funkcije su evaluirane kako bi se dobio osjećaj o preciznosti robota. Na kraju je postignuta sinteza naučenog koja je uključila sve navedene vrste programiranja i fizičke komponente robota kako bi se postigla interakcija čovjeka i humanoidnog robota. Sinteza je napravljena uspješno te je *Pepper* sada u mogućnosti kroz Laboratorij za računalnu inteligenciju provesti čovjeka, ispričati mu nešto o tri različita robota te s čovjekom voditi razgovor.

U nastavku rada na ovom području moglo bi se poboljšati kretanje koristeći *NAO* markere kako bi se dobio robusniji sustav kretanja. Također, moglo bi se raditi na autonomnosti robota gdje bi on bio sposoban obavljati zadatke bez da mu se izričito definira na koji način da to učini. Primjerice, da mora razvrstati različite predmete u prostoru po nekim parametrima (koristila bi se manipulacija predmetima, vizijski sustavi za raspoznavanje predmeta te *ROS* za kretanje u prostoru).

LITERATURA

[1] Pepper Robot Discover its technical specifications, Generation Robots, https://www.generationrobots.com/pepper/technical-specifications.html?lang=en, pristupljeno 17.2.2022.

[2] Gardecki A, Podpora M, Beniak R, Klin B. The Pepper humanoid robot in front desk application. 2018. https://www.researchgate.net/publication/327517390_The_Pepper_Humanoid_Robot_in_Front_Desk_Application, pristupljeno 22.2.2022.

[3] Ferretti M, Morgavi G, Veruggio G. The ACCEPTABILITY of Caregiver Robots in Elderly People. 2019. https://www.scitepress.org/papers/2018/66743/66743.pdf, pristupljeno 22.2.2022.

[4] For better business just add Pepper, SoftBank Robotics, https://us.softbankrobotics.com/pepper, pristupljeno 17.2.2022.

[5] ALEngagementZones, SoftBank Robotics ,http://doc.aldebaran.com/2-5/naoqi/peopleperception/alengagementzones.html, pristupljeno 17.2.2022.

[6] Motors, SoftBank Robotics, https://developer.softbankrobotics.com/pepper-naoqi-25/pepper-documentation/pepper-developer-guide/technical-overview/motors, pristupljeno 17.2.2022.

[7] Lynch, K.M., Park, F.C.: Modern Robotics Mechanics, Planning and Control, Cambridge University Press, 2017.

[8] ALLandMarkDetection, SoftBank Robotics, http://doc.aldebaran.com/2-5/naoqi/vision/allandmarkdetection.html, pristupljeno 17.2.2021.

[9] Contact and tactile sensors, SoftBank Robotics, http://doc.aldebaran.com/2-4/family/pepper_technical/contact-sensors_pep.html, pristupljeno 18.2.2021.

[10] NAOqi APIs, SoftBank Robotics, http://doc.aldebaran.com/2-4/naoqi/index.html, pristupljeno 22.2.2021.