TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tarkvarateaduse Instituut

Martin Nurmsaar 142390IAPB

EESTI KEELES SUHTLEV SOTSIAALSE KÄITUMISEGA ROBOT

Bakalaureusetöö

Juhendaja: Evelin Halling

MSc

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Martin Nurmsaar

22.05.2017

Annotatsioon

Lõputöö tulemusel loodi tarkvaralahendus, mis võimaldab inimesel suhelda robotiga eesti keeles. Robotina on kasutusel humanoidrobot NAO. Kõnetuvastuseks kasutatakse TTÜ Küberneetika instituudi Foneetika ja kõnetehnoloogia laboratooriumis loodud tarkvara eesti keelse kõne tuvastamiseks. Roboti poolne vastus loodakse Prologis kirjutatud dialoogisüsteemi abil, mis seejärel sünteesitakse kõneks kasutades Eesti Keele Instituudi poolt pakutavat tarkvara. Seejärel mängitakse sünteesi tulemusel saadud heli NAO roboti kaudu inimesele ette, mis jätab mulje, et robot suhtleb inimesega. Robot on võimeline käituma interaktiivselt, kui ta tuvastab oma kaameratega inimese näo või kuuleb suurt müra. Lisaks annab robot LED-lampide abil ka visuaalset tagasisidet selle kohta, kas robot hetkel räägib ise või kuulab hoopis inimest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 lehekülge, 3 peatükki, 26 joonist ja 1 tabelit.

Abstract

Estonian speaking robot with social competence

The main goal of the thesis is to implement a social robot, who can speak Estonian and interact with humans.

The results of the thesis are two applications. First application implements the physical moves of robot and interactions with human. Second one implements the loop of communication between robot and human.

Human speech is recorded through the microphone and sent to a speech recognition program, which returns a text containing the words, that human just said. This text is given as an input to a dialog system, which is supposed to find an answer based on the input. The answer given by the dialog system is now synthesized back to speech and played out through the speakers of robot. If there is too long silence in a conversation between human and robot, the dialog system takes the initiative and says something by itself.

Some interactions are also implemented for the robot, such as shaking hands, waving hand, giving a „high five“, scratching its head, looking towards the source of noise and detecting a human face.

Together the loop of communication and some interactions make an impression of social robot.

The thesis is in Estonian, contains 38 pages of text, 3 chapters, 26 figures and 1 table.

Lühendite ja mõistete sõnastik

library	teek, infoobjektide (failide, objektmoodulite, makrode) kogu
API	Application Programming Inteface, rakendusliides
Robot	reaalses maailmas asuv autonoomne masin, mis tunnetab, mõtleb ja reageerib keskkonnas toimuvatele muudatustele
dB	detsibell, mõõtühik, millega mõõdetakse müra taset
WAV	Waveform Audio File Format, audiofaili formaat
WebSocket	serveri protokoll, mis võimaldab täielikku kahepoolset suhtlust üle TCP
TCP	Transmission Control Protocol, edastusohje protokoll
JSON	JavaScript Object Notation, andmete hoidmise formaat
JPL	Java Interface to Prolog
JAR	Java Archive
LED	Light-Emitting Diode, valgusdiood
(J)VM	(Java) Virtual Machine, tarkvara, mis interpreteerib Java programmikeeles kirjutatud programme, mis on kompileeritud baitkoodideks ning salvestatud ".class" laiendiga failidena
HTTP	HyperText Transfer Protocol
POST	andmete edastamise protokoll üle HTTP
SFTP	SSH File Tranfer Protocol, võimaldab failide transportimist ja muutmist üle krüpteeritud ühenduse
JSCH	Java Secure Channel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust

Aina rohkem leidub roboteid, mis peavad igapäevaselt inimestega samas keskkonnas tegutsema. Sellised robotid peavad olema teatud sotsiaalsete oskustega ning oskama turvaliselt käituda inimese juuresolekul. Sotsiaalse kompetentsusega robotid on võimelised õppima uusi asju ning tegema inimestega koostööd, sellest tulenevalt on nende potentsiaalne kasutusvaldkond väga lai. Neid saab tulevikus piisava kompetentsi korral kasutada nii tööstuses, klienditeeninduses, tervishoius inimese assistendina kui ka haridusvaldkonnas. Sotsiaalses robootikas suudab robot mõista inimkeelt, kehakeelt ja talle jagatavaid käske ning tegevusi.

Stockholmi ülikooli KTH Royal Institute of Technology 2015. aastal koostatud sotsiaalse robootika arengukavas „Social Robotics Agenda“[1] jaotati sotsiaalsed robotid kolmeks peamiseks tüübiks.

Erinevad sotsiaalse roboti tüübid:

- esindusrobot – Inimese väliste omaduste järgi tehtud robot. Neid kasutatakse näiteks koosolekul kellegi asendamiseks või vanemate inimeste kodudes, et soodustada suhtlust. Üks selline robot on Geminoid HI-4, mis on ka joonisel 1 välja toodud. Parempoolne on pildil robot, vasakpoolne aga päris inimene.

[1] Social Robotics Agenda. (2015). [WWW] http://www.speech.kth.se/socialrobotics/Social_Robotics_Agenda.pdf (15.11.2015)

Joonis 1. Geminoid HI-4[2]

- humanoidrobot - Selliseid roboteid kasutatakse nii autistlike laste õpetamisel kui ka lihtsalt inimese sõbrana. Käesolevas lõputöös kasutasin samuti ühte sellist robotit, nimelt Aldebarani robotit NAO, mida on kujutatud joonisel 2.

Joonis 2. NAO robot[3]

- asukohapõhine robot – Selline robot leiab kasutust näiteks valvelaua administraatorina või giidina. Üheks selliseks robotiks on Aldebarani Pepper, mida on kujutatud joonisel 3. Telia veebilehe[4] andmetel kasutatakse nende peakontoris Pepperit külaliste vastuvõtmisel, registreerimistoimingute tegemisel ning töötajate külalistest teavitamisel.

---

[2] [WWW] http://www.geminoid.jp/projects/kibans/Images/010.jpg

[3] [WWW] http://d2fx8229io6twg.cloudfront.net/wp-content/uploads/2016/03/0M2A16791.jpg

[4]Telia uues peakontoris votab külalisi vastu baltimaades ainulaadne humanoidrobot Pepper. (2017) [WWW] https://www.telia.ee/uudis/-/uudis/telia-uues-peakontoris-votab-kulalisi-vastu-baltimaades-ainulaadne-humanoidrobot-pepp-1 (11.05.2017)

Joonis 3. Pepper robot[5]

Jüri Vainu ja Helena Sarapuu artikli „Towards context-sensitive dialogue with robot companion“[6] sissejuhatuses tuuakse välja, et on vaja välja arendada uue põlvkonna sotsiaalsed robotid selleks, et robotid saaksid füüsiliselt, sotsiaalselt ja turvaliselt integreeruda inimeste igapäevaellu. Tuuakse välja, et olemasolevad dialoogsüsteemid võimaldavad arendada lihtsama sisuga vestlust, võtta vastu häälkäsklusi teha muid lihtsamaid suhtlust nõudvaid tegevusi, aga sageli on selliste dialoogsüsteemide sõnavara limiteeritud ning puudub uute sõnade ja nende tähenduste meelde jätmise võimalus.

Eestis leiaksid sotsiaalsed robotid samuti kasutust, kuid siiani pole turul leidunud sobivat eestikeelset dialoogsüsteemi ning vastavat terviklikku lahendust, mis ühendaks kõnetuvastuse, dialoogsüsteemi ja kõnesünteesi.

1.2 Eesmärk

Käesoleva lõputöö eesmärgiks on luua tarkvara, mis paneb NAO roboti teatud määral sotsiaalselt käituma ja eesti keeles suhtlema. Robot peab olema võimeline suhtlema enda teadmusbaasi piires ning vastama mõningatele eeldefineeritud küsimusele (nt mis kell on, milline ilm on, kas uusi emaile on saabunud). Robot jätab teadmusbaasi meelde laused, mida seal eelnevalt polnud ning seeläbi täieneb teadmusbaas pidevalt uute faktidega.

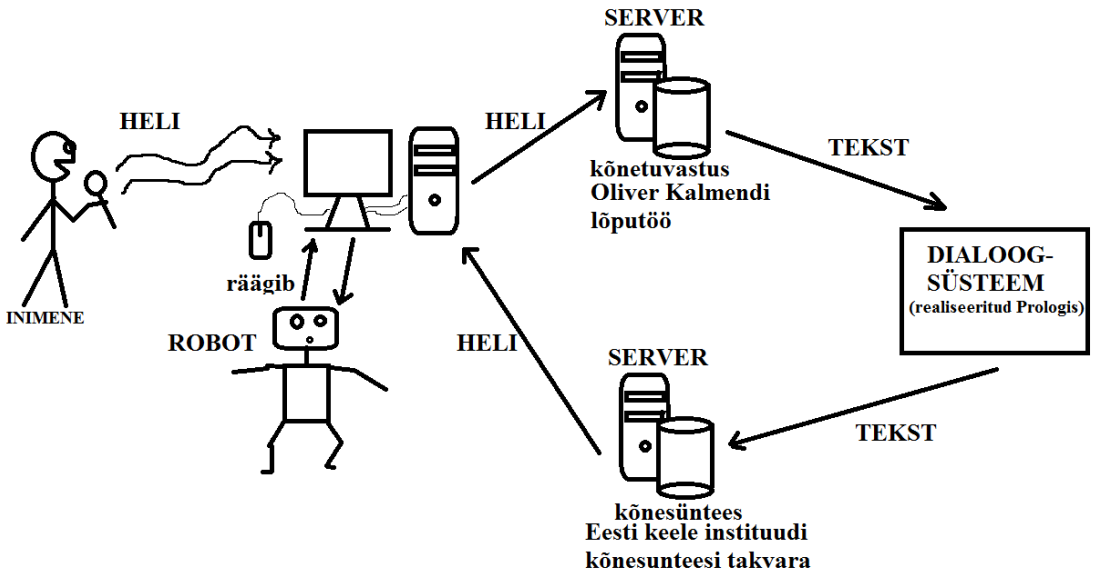
[5] [WWW] https://thenextweb.com/wp-content/blogs.dir/1/files/2014/06/pepper.jpg

[6] Vain, Jüri; Sarapuu, Helena (2014). Towards context-sensitive dialogue with robot companion. BEC 2014 : Proceedings of the 14th Biennial Baltic Electronics Conference, Tallinn University of Technology, October 6-8, 2014, Tallinn, Estonia. Piscataway, NJ: IEEE, 205–208

Robot peab suhtlema eesti keeles, kuna käesolev sotsiaalne robot on suunatud Eesti turule, kus hetkel sarnaseid robotabilisi pole.

2 Eesti keeles suhtlev sotsiaalne NAO robot

Joonisel 4 on toodud roboti ja inimese vahelise suhtlustarkvara arhitektuur. Kõigepealt räägib inimene mikrofoni, kust läheb heli arvutisse. Arvutist saadetakse saadud heli serverisse, kus toimub kõnetuvastus. Kõnetuvastuseks kasutatakse TTÜ Küberneetika instituudi Foneetika ja kõnetehnoloogia laboratooriumis loodud tarkvara eesti keele tuvastamiseks. Serverist saadetakse kõnest saadud tekst dialoogsüsteemi, mis leiab tekstile sobiva vastuse. Dialoogsüsteem on kirjutatud programmeerimiskeeles Prolog. Dialoogsüsteemi poolt väljastatav vastus on teksti kujul, seepärast saadetakse saadud tekst edasi kõnesünteesi serverisse. Kõnesünteesi teeb Eesti Keele Instituudi poolt pakutav tarkvara. Kõnesünteesist saadav kõne saadetakse tagasi arvutisse, mis on ühenduses NAO robotiga. Roboti kaudu mängitakse heli ette inimesele.



Joonis 4. Inimese suhtlus robotiga

Juhtprogramm, mis asub robotiga ühendatud arvutis ja kontrollib süsteemi tööd, on kirjutatud Javas. Dialoogsüsteem on kirjutatud Prologis ja integreeritud Java rakendusega.

2.1 NAO robot

NAO roboti näol on tegu 58 sentimeetrit kõrge Aldebarani poolt toodetud humanoidrobotiga, mida on kasutatud autistlike laste õpetamisel ning lastele robootika tutvustamisel. Sellel on oma graafiline arenduskeskkond Choreographe, mille kasutamine on lihtne ning ei nõua erilisi programmeerimisoskusi, visuaalselt saab arvutis lohistada erinevaid kastikesi ritta, mis kõik teevad erinevaid tegevusi. Veel on olemas rakendusliidesed Pythoni ja Java programmeerimiskeeltele, mille abil saab ise robotile programmi kirjutada. Käesolevas lõputöös kasutatakse nii Choreographi kui ka Java teeki NaoQi. Choreographis õppisin robotile erinevaid käsklusi jagama tehes läbi raamatus „An Introduction to Robotics with NAO“[7] olevad ülesanded ning Java NaoQi teegi jaoks kasutasin sellele teegile koostatud dokumenti[8]. Kuna paljusid teegi pakutavaid võimalusi pole Javale mõeldud dokumendis välja toodud, kasutasin ka Pythoni dokumenti[9], mille selgitused ja näited sai suhteliselt kerge vaevaga Javasse üle tuua. Kasutatava roboti tarkvara versioon on 1.14.5, mis pole küll uusim turul saadaval olev versioon (versioon 1.14.5 anti välja 2012. aastal, kuid praegu versioon väljas versioon 2.4.3, mis anti välja 2017), kuid see on piisav käesoleva lõputöö jaoks.

Lõputöö raames kasutatakse NAO kõlareid heli edastamiseks, LED-lampe erinevatest sündmustest märku andmiseks (näiteks silmade ja rindkere lamp lähevad roosaks, kui robot räägib ning kõrvade ja pea lambid teevad teevad ringe andes inimestele märku, et robot on valmis kõnet kuulama) ning realiseeritakse mõned liigutused, mida robot teatud olukordades teeb.

2.2 Projekti käivitamine

Kõigepealt tuleb avada Choreographe'i projekt „Nao_interactions“ ning panna see tööle. Programm tuleb taustal jooksma jätta.

---

[7] Beiter, M., Coltin, B., Liemhetcharat, S. (2012). An Introduction to Robotics with NAO.

[8] NaoQi Java SDK. (2012). [WWW] http://doc.aldebaran.com/1-14/dev/java/index.html

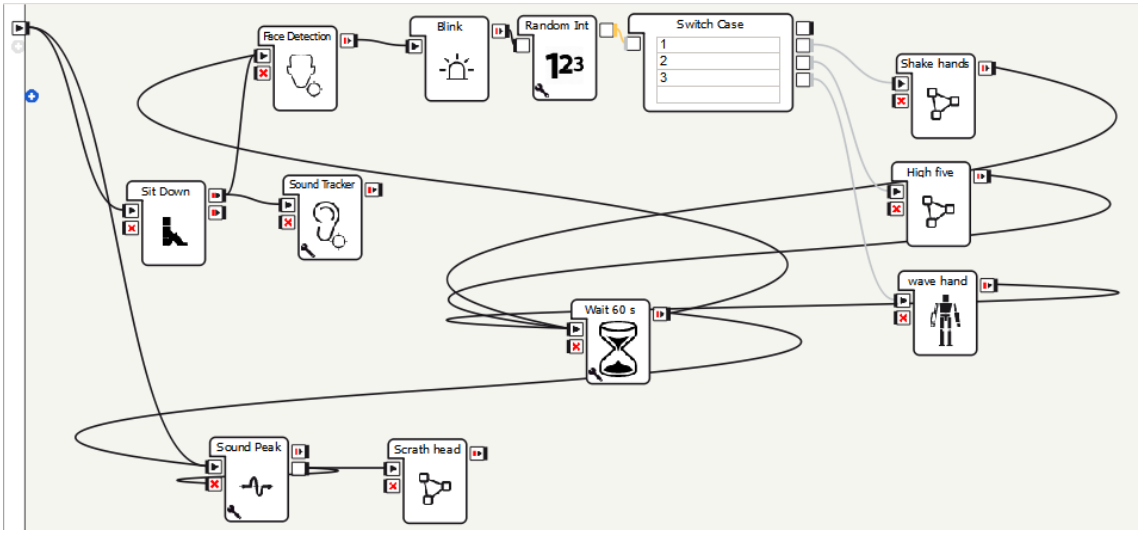
[9] NaoQi Python SDK. (2012). [WWW] http://doc.aldebaran.com/1-14/dev/python/index.html

Seejärel tuleb käivitada Java projekt „NAO_app", mille main-meetod asub „app" kaustas „RobotController" klassis. Programmi käima panemisel on vaja kaasa anda ka Java VM argument -Djava.library.path, mis sisaldab endas NAO roboti Java teegi ning Prologi ja Java integreerimiseks mõeldud teegi asukohti (nt -Djava.library.path=

"C:\Users\Martin\workspace\NAO_app\libraries\jnaoqi-1.14.5-win32\lib;C:\Program Files (x86)\swipl\bin" (Eclipse arenduskeskkonnas saab VM muutujaid lisada „Run Configurations" all olevasse „Arguments" sisendikasti).

2.3 Choreographe'i projekt

Kasutasin robotile interaktiivsete tegevuste lisamiseks arenduskeskkonda Choreographe, kuna seal on ette ära tehtud sellised tegevused nagu roboti pea liigutamne heli suunas ning puutetundlikel roboti osadel puute tundmise peale tegevuse alustamine. Interaktsioonide tegemine on vajalik, kuna muidu oleks robot lihtsalt paigal ning ei mõjuks eriti sotsiaalsena. Seletan joonise 5 abiga programmi tööpõhimõtet.



Joonis 5. Choreographe'i programm

Programmi käivitamsel võtab robot kõigepealt sisse istumise asendi, millesse ta jääb programmi sulgemiseni. Kõik interaktsioonid sooritab robot istudes. Seejärel alustatakse korraga näotuvastust ja heli allika asukoha tuvastust. Kui inimene ei jää roboti kaamera vaatevälja, ei saa näotuvastust teha. Seega on heli allika ehk inimese asukoha tuvastamine vajalik, kuna robot pöörab siis oma pea allika suunas ning see võimaldab inimnäol sattuda kaamera vaatevälja. Kui robot tuvastab enda läheduses inimese näo, vilgutatakse korraks

LED-lampe ning minnakse järgmise tegevuse juurde, milleks on juhusliku arvu genereerimine ühest kolmeni ning vastavalt genereeritud numbrile teatud interaktiivse tegevuse alustamine. Numbrile 1 vastav tegevus on käe surumine, number 2 on patsu viskamine (inglise keeles "high five") ja number 3 on käe lehvitamine. Valisin need kolm interakstisooni, kuna need on laialt levinud ning enamus inimesi peaks need liigutused ära tundma. Pärast interaktsiooni lõppu tehakse 60-sekundiline paus, pärast mida võimaldatakse jälle näotuvastust teha ning uue interaktsiooniga austada.

Käte surumine ja patsu andmine nõuavad ka inimese poolset panust. Käte surumise korral sirutab robot parema käe välja ning jääb ootama inimese kätt. Kui inimene puudutab roboti välja sirutatud käelaba puutepundliku osa, teeb robot oma käega üles-alla liigutuse ning peale seda asetab käe esialgsesse asendisse tagasi. Patsu andmise korral sirutab robot parema käe sirgelt ülesse ning paneb ta esialgsesse asendisse tagasi alles siis, kui inimene on käelaba puutetundlikku osa puudutanud.

Veel üheks roboti interaktsiooniks on pea sügamise liigutus, mis käivitub siis, kui robot tuvastab suure müra. Ka pärast seda interaktsiooni pehakse 60-sekundiline paus, peale mida võib sama interaktsioon uuesti esineda suure müra korral. Heli allika asukoha ja müra taseme tuvastamisel kasutatakse NAO küljes olevat mikrofoni.

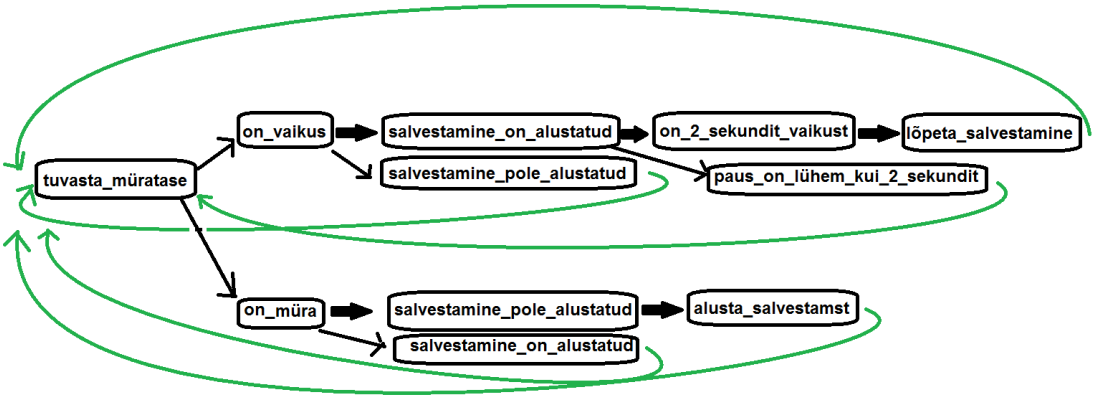
2.4 Inimkõne tuvastamine

Inimene saab robotiga suhelda mikrofoni kaudu. Kuna NAO roboti küljes olev mikrofon ei ole väga kvaliteetne ning lisaks inimkõnele satub mikrofoni ka palju taustamüra, kasutatakse suhtlemisel USB-ga arvutisse ühendatud mikrofoni, mida inimene hoiab oma suu lähedal robotiga vestlemiseks. Heli töötlemiseks kasutan teeki TarsosDSP[10], millega saab tuvastada müra taset. Helifaili salvestamiseks kasutasin Java Sound teeki ning näidet[11], milles näidati, kuidas selle teegiga heli salvestamist teha. Arvutis on käivitatud programm, mis kasutades seda teeki mõõdab kogu aeg heli tugevust ning alustab kõne

[10] A Real-Time Audio Processing Framework in Java. (2014). [WWW] https://github.com/JorenSix/TarsosDSP

[11] Capture and Record Sound into WAV File with Java Sound API. (2015) [WWW] http://www.codejava.net/coding/capture-and-record-sound-into-wav-file-with-java-sound-api

salvestamist, kui heli tugevus on piisavalt suur ning lõpetab salvestamise, kui paar sekundit on heli tugevus olnud piisavalt vaikne. Seletan programmi tööpõhimõtet joonisega 6.



Joonis 6. Kõne salvestamine

Kasutan boolean-tüüpi abimuutujat *salvestamine_on_alustatud*, mille väärtuseks määran esialgu *false*. Kõne algus tuvastatakse, kui mikrofoni jõudev müratase ületab vaikuse piiri (vaikuse piiriks on määratud -70dB). Kui muutuja *salvestamine_on_alustatud* väärtus on *false*, st et eelnevalt pole heli salvestamist alustatud, alustatakse kõne alguse tuvastamisel kõne salvestamist ning märgitakse *salvestamine_on_alustatud* väärtuseks *true*. Heli salvestamine lõpetatakse, kui programm tuvastab, et müratase mikrofonis ei ole 2 sekundit ületanud vaikuse piiri ning muutuja *salvestamine_on_alustatud* väärtuseks on *true*, st et salvestamisega on eelnevalt algust tehtud. Selliseks müratasemе tuvastamiseks Javas kasutan TarsosDSP teeki, mis võimaldab iga hetke kohta teada, kas müratase ületas või ei ületanud vaikuse piiri. Ise kirjutasin juurde heli salvestamise alustamise, 2-sekundilise pausi tuvastamise kõnes ning salvestamise lõpetamise. Heli salvestatakse .wav failina sellises formaadis, et kõnetuvastusprogramm suudaks salvestatud heli töödelda (formaat: *single channel, signed, 16-bit, little endian*).

2-sekundilise pausi tuvastamiseks tegin meetodi, mis saab argumendiks pausiks kuluvate sekundite arvu ning tagastab *boolean*-tüüpi väärtuse. Kasutasin Java Taimerit, mis kontrollis iga 0.1 sekundi tagant 20 korda järjest müra taset ning kui see tase jäi iga kord alla vaikuse piiri, tagastas meetod *true*, vastasel juhul *false*. Seda meetodit kujutab ka joonis 7.

```
boolean isSilentForNSeconds(int seconds) {
	Timer timer = new Timer();
	hundredthsOfSeconds = 0;
	TimerTask task = new TimerTask() {
		public void run() {
			if (isSpeaking == false && wasSpeaking == false) {
				hundredthsOfSeconds++;
				if (hundredthsOfSeconds >= seconds * 10) {
					timer.cancel();
					isSilentForNSeconds = true;
				}
			} else {
				timer.cancel();
				isSilentForNSeconds = false;
			}
		}
	};
	timer.scheduleAtFixedRate(task, 0, 100);
	return isSilentForNSeconds;
}
```

Joonis 7. Pausi tuvastamine

2.5 Kõnetuvastus

Kõnetuvastuse tarkvara on paigutatud serverisse. Juhtprogramm suhtleb kõnetuvastuse serveriga WebSocket-i protokolli alusel. Ühenduse loomiseks kasutan Tyrus-Standalone-Client[12] teeki, et saata serverisse binaarkujul andmeid ja võtta vastu serverist tulevat kõnest saadud teksti. Kasutasin näidet[13], kus õpetati selle teegi abil serveriga ühendust

[12] [WWW] http://repo1.maven.org/maven2/org/glassfish/tyrus/bundles/tyrus-standalone-client/1.9/tyrus-standalone-client-1.9.jar

[13] JSR 356 - Java API for Websocket (Java Client). [WWW] http://www.programmingforliving.com/2013/08/jsr-356-java-api-for-websocket-client-api.html (11.08.2013)

looma ja andmeid edastama ning vastu võtma. Kõnetuvastuse tarkvara kasutamiseks vaatasin kõnetuvastustarkvara kasutusjuhendit[14].

Kuna inimkõne salvestati WAV failina, on see vaja kõigepealt teisendada *ByteBuffer* tüüpi andmeteks, sest server võtab vastu vaid vastaval kujul binaarseid andmeid. Joonisel 8 on näha meetod *speechToBytes*, mis saab siendiks audiofaili ning tagastab audiofailist saadud binaarmassiivi.

```
byte[] speechToBytes(File speechFile) {
	ByteArrayOutputStream out = new ByteArrayOutputStream();
	BufferedInputStream in;
	try {
		in = new BufferedInputStream(new FileInputStream(speechFile));
		int read;
		byte[] buff = new byte[1024];
		try {
			while ((read = in.read(buff)) > 0) {
				out.write(buff, 0, read);
			}
			out.flush();
			return out.toByteArray();
		} catch (IOException e) {
			e.printStackTrace();
		}
	} catch (FileNotFoundException e) {
		e.printStackTrace();
	}
	return null;
}
```

Joonis 8. Helifaili binaarseks massiiviks teisiendamine

[14] Kaldi Gstreamer Server. (2014). [WWW] https://github.com/alumae/kaldi-gstreamer-server#server-usage

Seejärel teisendab joonisel 9 olev meetod *sendMessage* sisendiks saadava binaarmassiivi *ByteBuffer*-tüüpi andmeteks ning saadab selle serverisse.

```
public void sendMessage(byte[] message) {
    this.userSession.getAsyncRemote().sendBinary(ByteBuffer.wrap(message));
}
```

Joonis 9. Binaarse massivi saatmine serverisse

Serverist tuleb vastus JSON-ina (joonis 10). JSON tuleb parsida, et sealt soovitud tekst kätte saada. Selleks kasutatakse Json-Simple-1.1.1 API-t, mis võimaldab võtmesõna abil leida JSON-tüüpi tekstist soovitud väärtuse. Joonisel 9 olevas näites on näha, et tekstiks teisaldatud lause asub võtmesõna „transcript“ all, mis omakorda asub võtmesõnade „hyphotheses“ ja „result“ all. Seega tuleb teha kolm korda võtmesõna järgi otsingut (joonis 11), et saada JSON-ist soovitud väärtus.


```
{
   "status":0,
   "segment":0,
   "result":{
      "hypotheses":[
         {
            "transcript":"\u00fcks kaks kolm."
         }
      ],
      "final":false
   },
   "id":"327239d6-d614-4f82-a1de-674ab0ee3f48"
}
```


Joonis 10. Serverist saadud JSON vastus

JSON sisaldab võtmesõna „status“

```
String getParsedMessage(String message) {
	System.out.println("message: " + message);
	JSONParser jsonParser = new JSONParser();
	JSONObject jsonObject1 = null;
	String status = "";
	try {
		jsonObject1 = (JSONObject) jsonParser.parse(message);
		status = jsonObject1.get("status").toString();
		if (status.equals("0")) {
			JSONObject jsonObject2 = (JSONObject) jsonParser

	.parse(jsonObject1.get("result").toString());
			String str = jsonObject2.get("hypotheses").toString();
			str = str.replace("[", "").replace("]", "");
			JSONObject jsonObject3 = (JSONObject)
jsonParser.parse(str);
			return jsonObject3.get("transcript").toString()
					.replace(".", "");
		}
	} catch (ParseException e1) {
		e1.printStackTrace();
	}
	return "ei saanud aru";
}
```

Joonis 11. JSON-i parsimine

Olles saanud JSON-ist soovitud teksti kätte, salvestatakse see dialoogsüsteemi sisendfaili (joonis 12).

```
void addNewDSInput(String parsedMessgae) {
	file = new File(Constant.DS_INPUT_FILE);
	try {
		foStream = new FileOutputStream(file, false);
		parsedMessage += "\n";
		byte[] myBytes = parsedMessage.getBytes();
		foStream.write(myBytes);
		foStream.close();
	} catch (IOException e) {
		e.printStackTrace();
	}
}
```

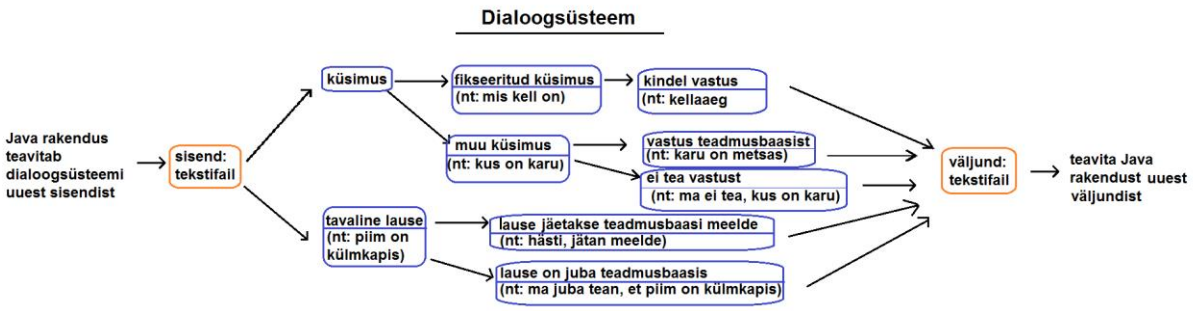
Joonis 12. Uue dialoogsüsteemi sisendi salvestamine ja dialoogsüsteemile uuest siendist teada andmine

2.6 Dialoogsüsteem

Dialoogsüsteem on kirjutatud Prolog'is. Lõputöö raames on olemasolevale dailoogsüsteemile lisatud emaili, kellaaja ja ilmaeenustuse küsimise võimalus, Java rakenduse ja dialoogsüsteemi integreerimine ning dialooogsüsteemi teadmusbaasis olevatele lausetele konteksti ja ajakohasuse lisamine. Dialoogsüsteem on seadistatud nii, et ta loeks sisendina tekstifaili ning annaks väljundi samuti tekstifaili. Kasutatakse ühte sisend- ja ühte väljundfaili, mille sisu uuendatakse pidevalt.

Kasutan JPL[15] teeki, et integreerida rakendust ning dialoogsüsteemi. See API võimaldab Java rakendusel teavitada Prologi programmi uuest sisendist ning dialoogsüsteemil teavitada rakendust uue väljundi olemasolust. Seletan dialoogsüsteemi toimimist joonise 13 abiga.

[15] [WWW] http://jpl7.org/740/ReleaseNotes740.jsp



Joonis 13. Dialoogsüsteem

2.6.1 Sisendi saamine

Kõnetuvastuse abil saadud tekst salvestatakse faili. Rakendus annab dialoogsüsteemile märku, et uus sisend on saadaval. Märku andmine käib nii, et rakenduses määratakse pärast kõnetuvastusest saadud teksti salvestamist faili *boolean* muutuja *isNewInputAvailable* väärtuseks *true* ning dialoogsüsteemi programmis on term, mis kontrollib selle muutuja väärtust (joonis 13). Joonisel 14 on näha, kuidas Java objekt Prologis välja kutsutakse ning selle muutuja väärtust Prologis kontrollitakse. Kui leitakse, et uus sisend on olemas, loeb dialoogsüsteem failist sisendi ja hakkab vastust otsima.

```
isInputAvailable(Condition, X) :-
        jpl_call(X, isNewInputAvailable, [], Condition).
```

Joonis 14. Uue sisendikontrolli term

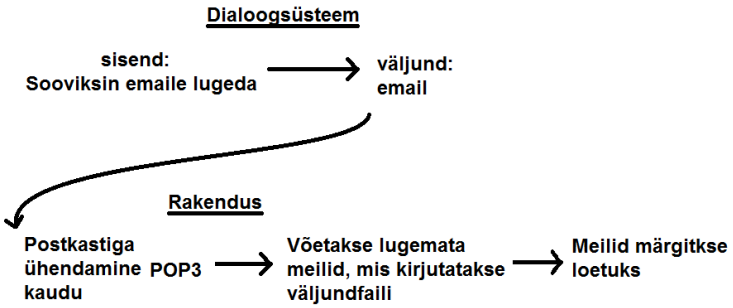
```
isInputAvailable(Condition, Obj), Condition == @(true)
```

Joonis 15. Sisendi kontrolli termi välja kutsumine

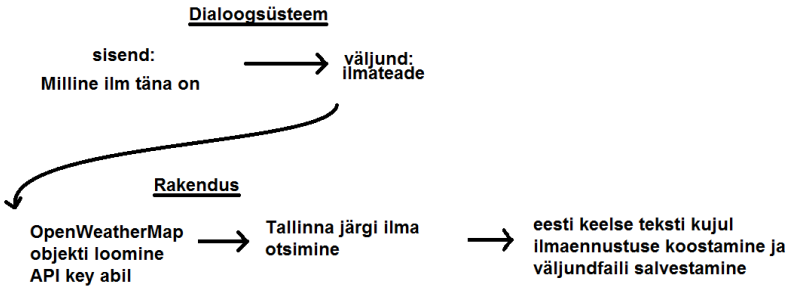
2.6.2 Ilma, emaili ja kellaja küsimine

Sisendiks saab olla küsimus või jutustav lause. Küsimusi on kahte liiki: kindla sõnastusega küsimused ja vabas vormis küsimused. Vabas vormis küsimusele otsitakse vastust teadmusbaasist ning seejärel tagastatakse leitud vastus või antakse teada, et ei leitud vastust. Teadmusbaasina kasutatakse tekstifaili, kuhu salvestatakse kõik uued laused koos lause konteksti ja ajakohasuse indeksiga.

Kindla sõnastusega küsimustele ei tea dialoogsüsteem ise vastuseid, seetõttu annab ta väljundiks küsimusele vastava märksõna (joonis 18), mille peale Java rakendus otsib välja sobiva vastuse. Sellised kindla sõnastusega küsimused on loodud, et anda küsijale võimalus saada teada informatsiooni, mida pole teadmusbaasis. Sel viisil on võimalik teada saada ilmaennustust (joonis 17), emaile (joonis 16) ja kellaaega. HTML-i kujul emailide kätte saamiseks kasutan JavaMail[16] teeki ja HTML-i parsimiseks JSOUP[17] teeki, ilmaennustuse jaoks OpenWeatherMap[18] teeki. Ilmaennustuse linnaks olen määranud Tallinna.



Joonis 16. Emailide lugemine



Joonis 17. Ilmaennustuse küsimine

[16] [WWW] http://www.oracle.com/technetwork/java/javamail/index.html

[17] [WWW] https://jsoup.org/

[18] [WWW] https://openweathermap.org/api

```
interpret([Term1,Term2, Term3, Term4], _, _, Answer3F):-
       Term1=milline, Term2=ilm, Term3=täna, Term4=on, Answer3F=[ilmateade],!.


interpret([Term1,Term2, Term3], _, _, Answer3F):-
       Term1=sooviksin,Term2=emaile , Term3=lugeda, Answer3F=[postkast], !.


interpret([Term1,Term2, Term3], _, _, Answer3F):-
       Term1=mis,Term2=kell , Term3=on, Answer3F=[kellaaeg], !.
```

Joonis 18. Dialoogsüsteemi kindla sõnastusega küsimused

2.6.3 Kontekst ja ajakohasuse indeks

Loodud on neli konteksti, milleks on sport, ilm, reisimine ja üldine kontekst, mille alla kuuluvad kõik laused. Igale sisendina tulevale lausele määratakse kontekst(id). Konteksti määramine käib nii, et iga konteksti kohta on teatud hulk märksõnu ning sisendiks oleva lause sõnu võrreldakse märksõnadega. Konteksti lisamine lausele võimaldab dialoogsüsteemil leida teemakohasemaid vastuseid esitatud küsimustele. Esialgu on neli konteksti piisav, aga pikemas perspektiivis, kui käesolevat programmi ja robotit hakatakse igapäevaselt inimeste keskel kasutama, on vaja kindlasti rohkem erinevaid kontekste. Sel juhul oleks mõistlik mõelda välja viis, kuidas dialoogsüsteem oskaks end ise uute kontekstide ja nende alla kuuluvate märksõnadega täiendada.

Lisaks määratakse igale lausele ka ajakohasuse indeks. Indeksina kasutatakse kellaaega ja kuupäeva, mil lause tuli sisendiks ning see salvestatakse koos lause ja lause konteksti(de)ga teadmusbaasi. Ajakohasuse indeks tuleb kasuks lausele vastuse otsimisel, võimaldades vastust otsida uuemate lausete hulgast. Seda saab arvestada ka siis, kui dialoogsüsteem ise ütleb vestluses pärast pikemat pausi suvaliselt valitud lause. Kui välja valitud lause juhtus olema teatud konteksti põhine lause, siis saab arvestada ajaindeksit ning valida esitamiseks uuem lause. Praegu on dialoogsüsteem seadistatud nii, et konteksti järgi otsitavate lausete korral otsitakse kõigepealt lauseid, mis on teadmusbaasi salvestatud ühe päeva jooksul. Kui sellele tingimusele ei vasta ükski lause, tehakse lause valimine kõikide vastava kontekstiga lausete hulgast.

2.6.4 Vastuse leidmine jutustavale lausele

Dialoogsüsteemile on võimalik sisendiks anda tavalisi jutustavaid lauseid. Sel juhul kontrollitakse, kas sisendiks tulnud lause on juba teadmusbaasis olemas. Kui lause oli teadmusbaasis olemas, öeldakse väljundis, et vastav fakt on juba teada. Kui lause polnud teada, jäetakse see teadmusbaasi meelde. Meeldejätmine võimaldab teadmusbaasil täieneda ning uusi lauseid õppida. Dialoogsüsteemist saadav vastus salvestatakse väljundfaili ning seejärel antakse Java rakendusele märku, et uus väljund on saadaval (joonis 19). Seda märguannet on vaja, et rakendus teaks, millal hakata väljundfailis oleva tekstiga kõnesünteesi tegema. Enne kõnesünteesi tegemist kontrollitakse rakenduses, kas dialoogsüsteemi väljundiks oli märksõna (nt email, ilmateade või kellaaeg) või mitte. Kui väljundiks oli märksõna, siis kutsutakse rakenduses välja märksõnale vastav meetod, mis koostab vastavalt ilmaennustuse, emailide või kellaaja teksti ning salvestab selle väljundfaili. Kui dialoogsüsteemi väljundiks ei olnud märksõna, ei pea väljundfailis midagi muutma. Saadud väljundfail edastatakse kõnesünteesi.

```
outputAvailable(Obj) :-
        jpl_call(Obj, dsOutputAvailable, [], _),
        jpl_call(Obj, getAnswer, [], _).
```

Joonis 19. Term, mis kutsub välja rakenduse meetodi

2.6.5 Vestluse alustamine dialoogsüsteemi poolt pika vaikuse korral

Selleks, et vestlus ei takerduks, kui inimene midagi ei räägi, on realiseeritud funktsionaalsus, kus piisavalt pika pausi korral ütleb dialoogsüsteem ise midagi omal initsiatiivil. Öeldava lause valib dialoogsüsteem juhuslikult selleks puhuks loodud lausete hulgast. Ühe sellise lause näiteks on lugemata emailide arvu ütlemine. Selle eesmärk on panna inimene rääkima, et siis vestlust jätkata. Kui inimkõnet jätkuvalt ei tuvastata, ütleb dialoogsüsteem jälle piisavalt pika pausi pärast juhuslikult valitud lause.

2.7 Kõnesüntees

Kõnesünteesi kasutatakse tekstist kõne saamiseks ning vastav tarkvara[19] on Eesti Keele Instituudi loodud ning vabavarana kasutatav. Kõnesüntees on paigutatud serverisse, seega peab rakendus looma ühenduse serveriga (joonis 20), saatma sinna sisendina dialoogsüsteemi väljundfailist saadud teksti (joonis 21) ning võtma vastu kõnesünteesi tulemusena saadavad audioformaadis andmed (joonis 22), mis salvestatakse helifaili. Tekst saadetakse üle HTTP POST-meetodiga, selle jaoks kasutatakse Java teeki HttpClient-4.3[20]. Kasutasin ühenduse loomiseks näidet lehelt www.mykong.com[21], kus on HttpClient teegiga realiseeritud serveriga ühenduse loomine ja POST meetodiga andmete saatmise ja vastu võtmise kohta üle võrgu. Nende näidete järgi tegin ka enda projektis serveriga ühenduse loomise ja andmete saatmise/vastuvõtmise POST meeetodi abil.

```
void createConnection(String targetURL) {
	try {
		url = new URL(targetURL);
		connection = (HttpURLConnection) url.openConnection();
		connection.setRequestMethod("POST");
		connection.setRequestProperty("User-Agent", “MOZILLA/5.0”);
		connection.setRequestProperty("Accept-Language", "UTF-8");
		connection.setUseCaches(false);
		connection.setDoOutput(true);
	} catch (IOException e) {
		e.printStackTrace();
	}
}
```

Joonis 20. Serveriga ühendamine

[19] Eesti keele HTS-kõnesüntesaator. (2016). [WWW] https://github.com/ikiissel/synthts_et

[20] [WWW] http://www.java2s.com/Code/Jar/h/Downloadhttpclient43beta1jar.htm

[21] How to Send HTTP Request. (2013). [WWW] https://www.mkyong.com/java/how-to-send-http-request-getpost-in-java/ (25.05.2017)

```
void sendRequest(String urlParameters) {
	try {
		DataOutputStream wr = new DataOutputStream(
				connection.getOutputStream());
		wr.writeBytes(urlParameters);
		wr.close();
	} catch (IOException e) {
		e.printStackTrace();
	}
}
```

Joonis 21. andmete edastamine serverisse (nt urParameters="data=mis kell on")

```
public String getResponse() {
	try {
		is = new InputStreamReader(connection.getInputStream());
		BufferedReader rd = new BufferedReader(is);
		StringBuffer response = new StringBuffer();
		String line;
		while ((line = rd.readLine()) != null) {
			response.append(line);
			response.append('\r');
		}
		rd.close();
		return response.toString();
	} catch (IOException e) {
		e.printStackTrace();
	}
	return "bad response";
}
```

Joonis 22. Serverist vastuse saamine

2.8 Heli robotisse

Olles kõnesünteesi tagajärjel saanud helifaili, tuleb see nüüd robotisse talletada ja selle kaudu ette mängida. NAO Java rakendusliides võimaldab robotiga mängida audiofaile, mis asuvad roboti enda mälus. Faile saab robotisse saata üle SFTP protokolli, mille kasutamist Javas võimaldab teek JSCH[22]. JSCH on teek, mis võimaldab luua failide edastamiseks turvalise ühenduse üle võrgu. Selle teegi kasutamist serveriga ühenduse loomiseks ja faili edastamiseks on näidatud kodehelp.com[23] lehel, kus on juhend ühenduse loomiseks ja faili saatmiseks, mida kasutati ka käesolevas töös ühenduse loomiseks ja faili saatmiseks. Joonisel 23 on näidatud, kuidas luuakse serveriga ühendus ning joonisel 24 on faili saatmine serverisse.

```
void createConnection() {
	JSch jsch = new JSch();
	try {
		session = jsch.getSession(SFTPUSER, SFTPHOST, SFTPPORT);
		session.setPassword(SFTPPASS);
		java.util.Properties config = new java.util.Properties();
		config.put("StrictHostKeyChecking", "no");
		session.setConfig(config);
		session.connect();
	} catch (JSchException e) {
		e.printStackTrace();
	}
}
```

Joonis 23. Turvalise ühenduse loomine FTP serveriga

[22] [WWW] https://mvnrepository.com/artifact/com.jcraft/jsch/0.1.54

[23] [WWW] https://kodehelp.com/java-program-for-uploading-file-to-sftp-server/

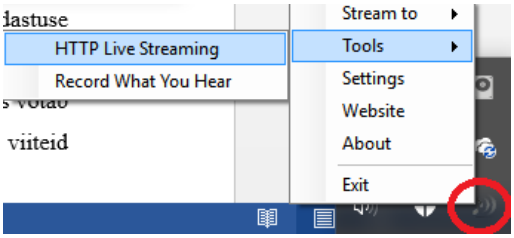
```
void uploadFile() {
	channelSftp = (ChannelSftp) channel;
	try {
		channelSftp.cd(SFTPWORKINGDIR);
		File f = new File(Constant.SPEECH_SYNTHESIS_OUTPUT_FILE);
		channelSftp.put(new FileInputStream(f), f.getName());
	} catch (SftpException | FileNotFoundException e) {
		e.printStackTrace();
	}
}
```

Joonis 24. Faili saatmine robotisse

Heli on võimalik robotisse ka voogedastuse kaudu edastada. Seda oleks vaja kasutada näiteks juhul, kui kõnesünteesi serveriga ei saada ühendust. Sel juhul tuleks arvutisse paigaldada EKI poolt pakutav programm festval.exe[24], mis pakub kahte erinevat mees- ja naishäält. Kuna see programm mängib kõnesünteesi tulemuse kohe arvuti kõlarite kaudu ette ega võimalda tulemust faili salvestada, tuleks arvuti kõlaritesse suunatud helist voogedastus teha. Sellist võimalust pakub Windowsi rakendus „Stream What You Hear“[25], mis koostab veebiaadressi, mille kaudu edastatakse arvuti heli. Selleks tuleb teha valik „Tools“ (joonis 25), seejärel valida „HTTP Live Streaming” ning avaneb aken, kust saab veebiaadressi kopeerida (joonis 26). NAO Java rakendusliides võimaldab voogedastuse heli roboti kaudu mängida. Selline voogedastuse lahendus pole siiski eriti efektiivne, kuna tekkis ligikaudu nelja sekundi pikkune viide kõnesünteesi ja heli robotisse jõudmise vahel, mis võtab robotilt sotsiaalset väärtust ära, kuna päris inimestevahelistes vestlustes nii pikki viiteid tavaliselt ei esine.

[24] [WWW] https://www.eki.ee/heli/kiisu/hts_v10_64bit.zip

[25] [WWW] http://www.streamwhatyouhear.com/download/



Joonis 25. „Sream What You Hear“ programmi avamine

Joonis 26. „Sream What You Hear“ voogedastuse aadress

2.9 Tulemused

Lõputöö tulemusena valmis Choreographe'i projekt, mis võimaldab NAO robotil teatud interaktiivseid liigutusi teha ning programm, mis ühendab ära inimkõne salvestamise, vaikuse tuvastamise, kõnetuvastuse, Prologi dialoogsüsteemi kasutamise, kõnesünteesi ning sünteesi tulemuse ettemängimise NAO roboti kaudu. Kõik see kokku tekitab suhtlusringi inimese ja arvuti vahel ning võimaldab robotil jätta mulje sotsiaalsest käitumisest.

Oluline näitaja sellise programmi puhul on ajaline viide inimkõne lõpu ja roboti sünteesitud vastuse ette mängimise vahel. Kui see on liiga pikk, ei meenuta selline suhtlus enam päris inimestevahelist vestlust. Kasutasin viite mõõtmiseks Java funktsiooni *System.currentTimeMillis()*, Valisin mõõtmiseks kõnetuvastuse, kõnesünteesi, faili saatmise robotisse, helifaili esitamise robotis ning dialoogsüsteemist vastuse saamise, kuna pidasin neid kõige tõenäolisemateks viite tekkimise kohtadeks. Tegin 10 mõõtmist ja panin nende tulemused kirja tabelisse 1.

Nr	Kõnetuv astuse viide	Dialoogsüsteemist vastuse saamise viide	Kõnes ünteesi viide	Viide faili saatmisel robotisse	Viide faili ette mängimisel roboti kaudu	Viited kokku (millisekundites)
1	1035	27	601	326	785	3475
2	1027	28	476	323	793	3551
3	1043	27	567	392	810	3580
4	1179	30	468	183	752	3246
5	1039	26	582	202	768	3424
6	1048	27	503	186	757	3310
7	1011	23	472	182	722	3170
8	1050	24	607	318	756	3228
9	1040	27	503	189	778	3293
10	1126	26	483	213	794	3414
Keskmised tulemused	1060	27	526	251	772	3369

Tabel 1. Viited programmis

Ilmeb, et programmis esineb keskmiselt 3369 millisekundiline viide inimkõne lõpu ja roboti dialoogsüsteemist saadud sünteesitud vastuse roboti kaudu ette mängimise vahel. See viide pole väga pikk, kuid selleks, et selline suhtlusring meenutaks rohkem päris vestlust, võiks viiteaeg lühem olla. Artiklis „Pauses Can Make or Brake a Conversation“[26]

[26] Pauses Can Make or Brake a Conversation. (2015). [WWW] http://hum.gu.se/english/current/news/Nyhet_detalj/pauses-can-make-or-break-a-conversation.cid1320238 (7.09.2015)

tuuakse välja, et kahe inimese vahelises vestluses on keskmine paus ühe inimese lause lõpu ja teise inimese lause alustamise vahel ligikaudu pool sekundit. Arvutiga vestluses oleks väga keeruline nii kiiret suhtlusringi realiseerida. Kõige pikema viite tekitas kõnetuvastus, märgatavad viited on ka kõnesünteesil, helifaili saatmisel robotisse ja heli ette mängimisel. Dialoogsüsteemil tekkivat viidet inimene ei taju. Kuna selliseid viiteid, mida inimene tajub, tekib mitmes kohas, on mitmeid võimalusi viiteaja vähendamiseks.

Üheks võimaluseks on otsida selline voogedastamise rakendus, kus on väike viide. Lõputöö raames testisin tasuta saadaval olevat voogedastuse programmi „Stream What You Hear“, mis tekitas mitmeid kordi suurema viite kui faili edastamine robotisse ja selle robotis ette mängimine. Heli ette mängimise viiteaega ei saa vähendada, seda võimalust pakub NaoQi teek ning seal midagi muuta ei saa. Kiiremat lahendust faili edastamiseks robotisse kui praegu realiseeritud lahendus, ei leitud. Sama kehtib ka kõnetuvastuse kohta, kus leiti vaid javax.WebSocket teek ühenduse loomiseks ja andmete edastuseks. Kõnesünteesi jaoks andmete edastamiseks HTTP protokolli ning on mitmeid erinevaid teeke, mis pakuvad võimalust selle protokolli järgi andmeid edastada. Seega saaks proovida, kas mõnda teist teeki kasutades saadakse väiksem viide.

Käesoleva lõputöö raames kasutati TarsosDSP teeki heli tugevuse tuvastamiseks ning Java Sound teeki heli salvestamiseks. Hetkeseisuga ei toimi salvestamine korrektselt, nimelt salvestusprotsessi tulemusena saadakse soovitavast tulemusest kiirem ja mahult väiksem heli. See teeb raskemaks kõne tuvastamise, kuna kõnetuvastusprogramm ei pruugi kõikidest sõnadest enam aru saada.

3 Kokkuvõte

Lõputöö tulemusena valmis terviklik süsteem, mis edastab mikrofoni räägitud jutu kõnetuvastusse, saadab kõnetuvastusest saadud teksti dialoogsüsteemi, saadab programm dialoogsüsteemist saadud vastuse kõnesünteesiga tegelevasse serverisse, edastab sealt saadava heli arvutisse, mis on ühenduses NAO robotiga ning lõpuks mängib heli roboti kaudu ka ette. Lisaks loodi robotile teatud liigutused, mis jätavad robotist sotsiaalsema mulje. Dialoogsüsteemile lisati võimalus küsida mõned küsimused, mille vastuseid ei ole võimalik leida dialoogsüsteemi teadmusbaasist. Igale sisendiks tulevale lausele määratakse kontekst ning antakse kaasa ajakohasuse indeks, mis teeb lihtsamaks sobivamate vastuste otsimise.

Realiseeritud suhtlusprogrammis esineb inimkõne lõpu ja roboti vastuse vahel keskmiselt 3369 millisekundiline viide, mis on veidi liiga pikk. Probleeme esineb ka kõne salvestamisel ja kõnesünteesi tulemuse faili salvestamisel, nimelt mõlema tulemusel saadavatel helifailidel on probleeme heli kvaliteediga.

Kasutatud kirjandus

1. 15.11.2015. Social Robotics. A strategic innovation agenda. [WWW] http://www.speech.kth.se/socialrobotics/Social_Robotics_Agenda.pdf
2. Telia uues peakontoris votab külalisi vastu baltimaades ainulaadne humanoidrobot Pepper. (2017) [WWW] https://www.telia.ee/uudis/-/uudis/telia-uues-peakontoris-votab-kulalisi-vastu-baltimaades-ainulaadne-humanoidrobot-pepp-1 (11.05.2017)
3. Vain, Jüri; Sarapuu, Helena (2014). Towards context-sensitive dialogue with robot companion. BEC 2014 : Proceedings of the 14th Biennial Baltic Electronics Conference, Tallinn University of Technology, October 6-8, 2014, Tallinn, Estonia. Piscataway, NJ: IEEE, 205−208
4. Kalmend, O. (2017). Robotplatvorm loomuliku keele dialoogsüsteemidele : magistritöö. Tallinn, Tallinna Tehnikaülikool.
5. Beiter, M., Coltin, B., Liemhetcharat, S. (2012). An Introduction to Robotics with NAO.
6. NaoQi Java SDK. (2012). [WWW] http://doc.aldebaran.com/1-14/dev/java/index.html
7. NaoQi Python SDK. (2012). [WWW] http://doc.aldebaran.com/1-14/dev/python/index.html
8. A Real-Time Audio Processing Framework in Java. (2014). [WWW] https://github.com/JorenSix/TarsosDSP
9. Capture and Record Sound into WAV File with Java Sound API. (2015) [WWW] http://www.codejava.net/coding/capture-and-record-sound-into-wav-file-with-java-sound-api
10. JSR 356 - Java API for Websocket (Java Client). [WWW] http://www.programmingforliving.com/2013/08/jsr-356-java-api-for-websocket-client-api.html (11.08.2013)
11. Kaldi Gstreamer Server. (2014). [WWW] https://github.com/alumae/kaldi-gstreamer-server#server-usage
12. [WWW] http://jpl7.org/740/ReleaseNotes740.jsp

13. [WWW] http://www.oracle.com/technetwork/java/javamail/index.html
14. [WWW] https://jsoup.org/
15. [WWW] https://openweathermap.org/api
16. Eesti keele HTS-kõnesüntesaator. (2016). [WWW] https://github.com/ikiissel/synthts_et
17. [WWW] http://www.java2s.com/Code/Jar/h/Downloadhttpclient43beta1jar.htm
18. How to Send HTTP Request. (2013). [WWW] https://www.mkyong.com/java/how-to-send-http-request-getpost-in-java/ (25.05.2017)
19. [WWW] https://kodehelp.com/java-program-for-uploading-file-to-sftp-server/
20. [WWW] http://www.streamwhatyouhear.com/
21. Pauses Can Make or Brake a Conversation. (2015). [WWW] http://hum.gu.se/english/current/news/Nyhet_detalj/pauses-can-make-or-break-a-conversation.cid1320238 (7.09.2015)
22. Inglise-Eesti sõnastik. (2017). [WWW] http://www.vallaste.ee/