Sapienza Università di Roma
Department of Computer, Control, and Management
Engineering Antonio Ruberti
Artificial Intelligence and Robotics

Project Pepper

Course: Elective in Artificial Intelligence

Student: **Amila Sikalo**
Matricola: **1938032**
Student: **Andreea Nica**
Matricola: **1593889**

July 15, 2022

Contents

1 Introduction 3
2 Related work 4
3 Solution 6
3.1 Setup Pepper Environment 6
4 Implementation 9
4.1 Connecting 9
4.2 Planning 9
4.3 Main code 12
4.4 Pepper's Tablet 14
4.5 Robot movements 16
5 Final result 17
6 Conclusion 19

List of Figures

1 Pepper robot [2] . . . 4
2 Using Pepper in pediatric cases[8] . . . 4
3 Using Pepper in medical procedures [11] . . . 5
4 Pepper Emulator . . . 6
5 Architecture of the project . . . 7
6 Planning in the case of answer 'no' . . . 10
7 Planning in the case of choosing dancing moves . . . 11
8 Interaction using two .top files . . . 11
9 Starting the tablet view . . . 12
10 Original html site for introduction . . . 14
11 Playing the game . . . 14
12 Page for choosing the story . . . 15
13 Pepper's story . . . 15
14 Page for choosing a video . . . 15
15 Pepper's Video Ice Age . . . 16
16 Pepper move - Hi . . . 16
17 Dance move example 1 . . . 17
18 Dance move example 2 . . . 17
19 Buttons are hidden before human-robot interaction . . . 18
20 Buttons are shown after human-robot interaction . . . 18
21 Interaction with cat motion - part 1 . . . 18
22 Interaction with cat motion . . . 19

Both members worked equally on this project.

1 Introduction

Social robots are a current trend that is being studied in different healthcare services, all designed to help humans go beyond what we can naturally and safely do ourselves.These robots' uses in surgery and other branches of medicine are still expanding quickly. One example of how healthcare is continuing to push the limits of technology is the use of robots in operating rooms and clinics, which is now becoming the norm.

Although there are now only few care robots employed to help nurses with the numerous activities they complete each hour, this number is anticipated to rise dramatically over the next ten years. Many of these activities, like collecting blood, keeping track of temperature, or enhancing patient hygiene, are straightforward but essential. Nurses would have more time to devote to developing treatment regimens and providing tailored patient care if robots could assist with these straightforward repetitive activities[1].

Through its ability to socially interact, encourage patients, and show patients how to accomplish certain motor tasks, social robots provide patients—particularly the old and young patients—in the hospital setting with cognitive support. Social robotic technology, which is still in its infancy, has the ability to support and bolster care while also helping the healthcare system meet rising demand. Social robots may offer a valuable platform for meeting the special care needs that children present. Robots might help kids manage chronic illnesses by promoting healthy behaviors and providing information. These more human-like robots are capable of carrying out their duties with a high degree of autonomy and doing so while relating to patients and clinical staff in a natural way. These social robots may be able to give patients the social interaction they could be missing at hospitals across the globe that are suffering from a nursing shortage. Child life specialists will offer therapeutic interventions to hospitalized children for developmental and coping assistance in several hospitals' pediatric units. Play, preparation, education, and behavioral diversion are all part of this for both ordinary medical treatment and before, during, and after challenging operations. Therapeutic medical play and re-establishing a normal atmosphere through pastimes like making crafts, playing games, and celebrating are examples of traditional therapies. For the first time, a new study shows that "social robots" employed in support sessions held in pediatric units of hospitals might help sick children feel better[1].

This work presents a comprehensive overview of social robots in therapy and the healthcare of children, in order to clarify and summaries the current state of the art and to contribute to the facilitation of ongoing research and potential clinical applications. Specifically, the review aims to determine the types of studies that have been conducted, the health conditions that social robots are used with, the effectiveness of the robots, and the gaps that remain in the research [1].

In this work we will see Pepper robot (see Figure 1) immersed in an environment with the aim to contribute to the improvement of the hospital services, bringing huge help to the whole medical staff and becoming part of it. In our case, Pepper must be able to help distract children coping with acute medical procedures or provide companionship and comfort.

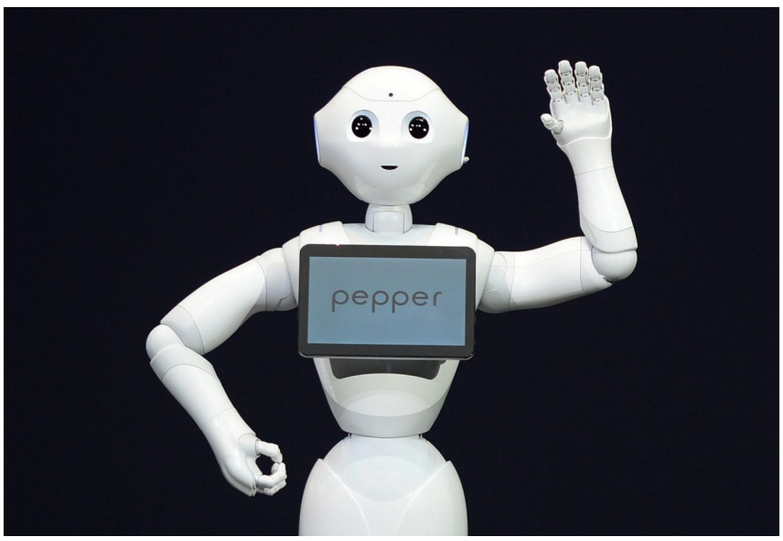

Figure 1: Pepper robot [2]

2 Related work

Research in pediatric procedural pain has increased exponentially since the late 1980s, and yet quality assurance of pain management has not yet been established [3]. Given the often inadequate pain reduction achieved with the isolated use of pharmacotherapy, and the relative noninvasive and general ease of most non-pharmacological approaches, more and more attention has turned to non pharmacologic interventions. Distraction, as such, is a method of pain management that redirects attention away from a painful stimulus. Strategies such as health-care professionals wearing distracting designs on their clothing and the use of clown doctors are emerging as effective approaches by encouraging children to attend to playful stimuli although music and cartoons have shown effectiveness in reducing pain and anxiety among children undergoing a variety of medical procedures [4], [5], other studies reveal that their results are not consistent [6] [7].

Figure 2: Using Pepper in pediatric cases[8]

It would appear that these distractions are not always strong enough to turn children's attention away from the pain. It is now believed that multi sensory strategies, which combine visual, auditory, and tactile senses, may have a greater impact on pain than single-sensory strategies. Given the mixed results mentioned above, it stands to reason that stronger and more engaging forms of distraction, which invite the child to engage in an activity, may be necessary for medical procedures [9].

The goal is to turn children's attention away from the pain of the needle toward an enjoyable activity. Indeed, one of the principles of attention capacity theory is that the distraction stimulus must be stronger than the pain stimulus to gain the child's attention.

As such, novel, interactive, and multi sensory distractions are needed. In view of children's growing enthusiasm for technological devices, we propose that the use of technologically enhanced devices may create a strong distraction to gain children's attention, and have a positive impact on their emotions during medical procedures as exhibited by increased smiling and decreased crying [9].

Our idea is to place Pepper in a pediatric hospital. Children can interact with the robot through the tablet (placed on the robot's chest) or it is possible to directly conversation with it. Pepper provides also cartoons and fairy tales and allows you to listen songs [10].

Body language is a large component of human communication and can be useful if the robot and human are expected to interact with the environment. Gestures, when augmenting a vocal grammar, are an important part of interactions with the real world. Then, to let Pepper be more social and sustain a more natural interaction, we implemented several gestures.

Previous evidence suggests that eye-contact serves a number of different functions in two-person encounters, of which one of the most important is gathering feed-back on the other person's reactions. It is further postulated that eye-contact is linked to affiliation motivation, and that approach and avoidance forces produce an equilibrium level of physical proximity, eye- contact and other aspects of intimacy. If one of these is disturbed, compensatory changes may occur along the other dimensions. Experiments are reported which suggest that people move towards an equilibrium distance, and adopt a particular level of eye-contact. As predicted, there was less eye-contact and glances were shorter, the closer two subjects were placed together (where one member of each pair was a confederate who gazed continuously at the other). The effect was greatest for opposite-sex pairs. In another experiment it was found that subjects would stand closer to a second person when his eyes were shut, as predicted by the theory [10].

Figure 3: Using Pepper in medical procedures [11]

For this reason, we have introduced a computer vision system that enables Pepper to understand the patient's emotions, and we employed this system to monitor them when they are listening to a song, in order to understand if that particular song is liked or not. So, the robot is able to understand the situations and give advice according to the patient's behavior. Moreover, Pepper, through its sonar, can perceive the humans in its vicinity, and this capability lets Pepper approach patients. Finally, the robot can also perceive if a person touched him.[8]

In this regard, for the university course delivered by professor Iocchi, Human-Robot Interaction, we developed a project focusing on social interaction with people. We used a virtual environment provided by SoftBank Robotics, which makes it possible to interact with the Pepper Robot. Pepper is a humanoid robot, purposely designed to allow social interaction. The humanoid morphology is fundamental because during an interaction people expect to find a similar one, who is able to behave according to some social rules and is capable of different interaction modalities. In fact Pepper can exploit its several sensors (rbg cameras, depth camera, microphone, speakers, touch sensors, tablet) to reach

a high level of "sociality", in the sense of social skills. Thanks to human-like actuators and vision sensors it is also possible to accompany verbal communication to enable social signal processing, and further, with the use of touch sensors and the tablet, multi-modal interaction can be designed, handling multiple modalities [9]. Our project was developed mainly in Python using NAOqi1, a SDK from Softbank Robotics, to connect directly to the "robot", which provides the needed APIs.

3 Solution

In this section, we will first introduce the basic idea behind the implementation, as well as the used tools.

3.1 Setup Pepper Environment

In order to properly create the project, we installed docker on our computers using the following commands:

```
curl -fsSL get.docker.com -o get-docker.sh
sudo sh get-docker.sh
rm get-docker.sh
sudo usermod -aG docker $USER
```

After that, we cloned the pepper_tools file, and downloaded all the required Python libraries. In order to create connection between tablet and robot, we had to download MODIM files:

```
git clone https://bitbucket.org/mtlazaro/modim.git
```

In order to run our project, we had to first build image using the following command:

```
./build.bash
```

and after that, to run the image using docker environment:

```
./run.bash
```

and entering the console within the docker container in order to run the code.

```
docker exec -it pepperhri tmux
```

After following the instructions above, go ahead on this file to download and test the android SDK emulator Open the Emulator: create a new project on android studio (Robot application) and go to `Tools->Pepper SDK->Emulator`'

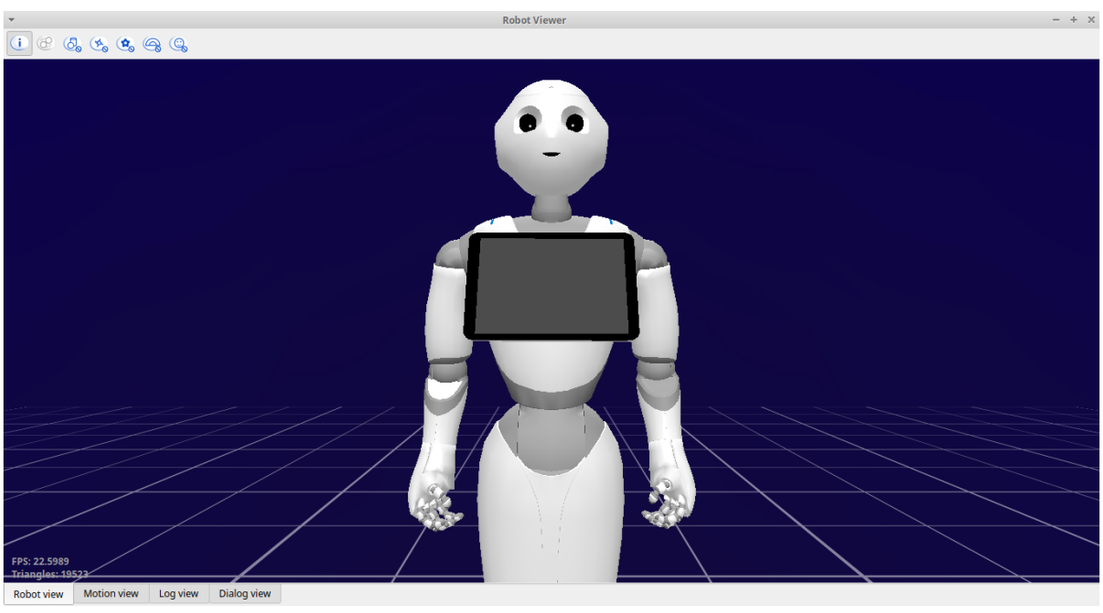

Figure 4: Pepper Emulator

Overall architecture of the system can be illustrated with a following figure.

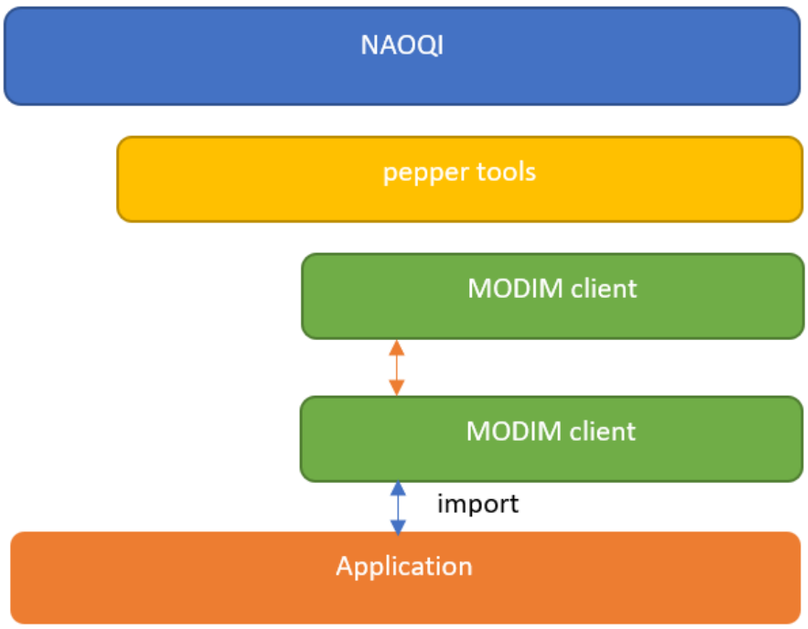

Figure 5: Architecture of the project

The project was created on a Linux environment using Ubuntu. To establish one or more separate containers where it is feasible to operate independently without changing the overall system, we utilized Docker. By using virtualization at the operating system level and adding an extra layer, it is made feasible. To do this, run the following instructions. run a docker image containing all the libraries you've thought about, and use that image to learn the architecture of your robot. Python was used to communicate with the robot (explicitly to handle the libraries made accessible by NAOqi), together with JavaScript and HTML to build several tablet windows.

In order to best serve the demands of Softbank Robotics' robots, the Pepper robot runs on the NAOqi operating system. The NAOqi APIs are a collection of programs and libraries that enable the robot to interact with humans in a variety of ways, such as through speech, gestures, robot motions, and more. This serves as the foundation around which our software architecture is constructed. The NAOqi server must be started in order to connect with the robot; in our instance, it is started when we run the SDK for Android emulator. Then, owing to the accessible tools – the pepper tools – and the documentation of Softbank, we can implement the complete interaction with the robot.

ALMotion

The ALMotion module offers techniques that make moving the robot easier. There are four main categories of techniques for regulating the following:

1. joint stiffness (basically motor On-Off)
2. joint position (interpolation, reactive control)
3. walk (distance and velocity control, world position and so on)
4. robot effector in the Cartesian space (inverse kinematics, whole body constraints)[12]

The ALMotion module also carries out a few "reflexes," including Self-collision avoidance, External-collision avoidance, Fall manager, Smart Stiffness, and Diagnosis effect. Reflexes are automatically activated and change how a robot behaves. More particularly, they

1. alter the motion to avoid self or external collision,

2. adjust the stiffness to spare power when the robot is not moving,

3. trigger a specific task when a fall is detected, or when issues or errors are raised on a device potentially critical.

Along with giving access to essential information about the robot hardware, such as the number of joints, their name, limitations, the available sensors, and so forth, the module also offers tools for defining the robot's behavior in the absence of human commands[12].

ALDialog

The ALDialog module enables you to provide your robot conversational abilities by making use of a set of "rules" that are properly defined and arranged. ALDialog controls the communication between the human and the robot using a set of specified rules. There are two categories of these regulations: proposal rules and user rules [12].

A User rule links a specific user input to possible robot output.

```
u: (Hello Nao how are you today) Hello human, I am fine thank you and you?
```

A Proposal rule triggers a specific robot output without any user output beforehand.

```
proposal: Have you seen that guy on the TV yesterday?
u1: (yes) He was crazy, no?
u1: (no) Really, I need to tell you.
```

In order to properly manage the conversation between the human and the robot, the rules are grouped by Topics.

```
topic: ~greetings
language: enu

u: (Hello Nao how are you today) Hello human, I am fine thank you and you?
u: ({"Good morning"} {Nao} did you sleep * well) No damn! You forgot to
switch me off!
proposal: human, are you going well ?
u1: (yes) I'm so happy!
u1: (no) I'm so sad
```

A Topic can either be created and edited directly in any text editor or through a qiChat Choregraphe box.

ALTextToSpeech

The robot can speak thanks to the ALTextToSpeech module. A text-to-speech engine receives orders from it, and voice customisation is also permitted. The robot's loudspeakers receive the synthesis's output. Based on speech synthesizers, sometimes known as speech engines, is ALTextToSpeech. The audio stream that is output can be changed. For instance, the following effects are accessible: pitch shifting effect modifies the initial pitch of the voice, and double voice effect produces a "delay/echo" effect on the main voice [12].

ALMemory

ALMemory is a centralized memory used to keep track of all the important data pertaining to the hardware setup of your robot. More particular, ALMemory offers details on the Actuators and Sensors' status right now. In addition to serving as a central location for the dissemination of event alerts, ALMemory may be used to store and retrieve named data. A mutexed and unordered boost map is ALMemory. The map includes different (ALValue). A MicroEvent called an Event records a person's history in ALMemory. Although they essentially overlap, MicroEvent is quicker. It can be accessed to event history using ALMemoryProxy::getEventHistory.[12]

Furthermore a Module can autostart when someone subscribe to an Event. This feature is accessible using ALMemoryProxy::declareEvent with two parameters.

4 Implementation

We handled the interaction across 5 communication channels, including textual, verbal, tablet, visual, and touch interactions, and gave additional tools to make the robot even more sociable. In particular, we made the interaction as multimodal as feasible.

4.1 Connecting

In our project we included only the second method, being more readable and tidy. Moreover, MODIM allows to render on our local web page - through the im.displayLoadURL() command – all what we have implemented on its "touch screen", thus having a visual feedback of what just done[9]. However to do this work we needed another tool: a web server that connects our localhost HTML page with our architecture. So, it is here that the importance of the HTML code developed comes out, in which you can define the layout of the Pepper tablet according to your liking. Finally, to let the NAOqi server and MODIM work properly, we need to execute the following steps:

1. To activate the NAOqi server start the emulator: go to Android Studio > Tools > Pepper SDK > Emulator;
2. open the html file $HOME/playground/PepperHRI/tablet/index.html on the browser
3. To enable the MODIM server, go to $HOME/src/modim/src/GUI and run python ws server.py

It is provided a (fantasy) sonar-based human detection system for our robot Pepper. In specifically, before beginning the interaction, Pepper recognizes every person in the space (for simplicity, we assumed that the robot can only sense people in front of it) and calculates the distance between them. Meanwhile, Pepper turns its head left and right in a gesture that mimics the motion of looking for someone. Pepper opens the conversation with a welcome message that uses the TTS modality after it has determined who it wishes to interact with. While this is happening, Pepper makes an asynchronous "Hello" gesture with its right arm, wrist, and hand. The following piece of code demonstrates it[9].

4.2 Planning

Planning part if fully done using two files: main.top and story.top. This is mostly realated to the first part of the project, after the robot starts to search for human, and analysing who is child, and who is not. There can be potentially two problems when Pepper enters the room:

1. The person Pepper sees first is not a child
2. The person Pepper sees first is a child, but not the child who is sick, and needs help

In order to allow Pepper to only work with sick children, Pepper asks the following question:

1. Hi, are you over 18?

If the answer on the first question is 'yes', the Pepper writes the following sentence, and keeps searching for another person. If the answer is 'no', the Pepper proceeds with the conversation.

Planning part coding example is given below:

```
topic: ~example-assistant()
language: enu

concept: (myname) ["My name is" "I'm" ]

proposal: %name What's your name?
u1: ({~myname} _*) Hi $1 ^goto(age)
```

```
proposal: %age Do you have less than 18 years?
u1: (no) Oh, I am sorry, I only play with kids! Bye!
u1: (yes) ^goto(question)

proposal: %question Oh, that is great! Would you like to play for a bit?
u1: (yes) ^goto(question2)
u1: (no) ^goto(question3)

proposal: %moves What kind of moves would you like to see? Cats, dogs, or dance?
Type "cats", "dogs" or "dance"
u1: (dogs) Oh, let's see some dogs moves!
u1: (cats) Oh, let's watch some cats moves!
u1: (dance) Oh, let's watch some dance moves!

proposal: %question2 Ok, we need to use my tablet so we could play. Is that
fine with you?
u1: (yes) Let's turn on the tablet!
u1: (no) Are you sure about that?
u2: (yes) Bye!
u2: (no) You're confusing. ^goto(question)

proposal: %question3 Ok, Let's start with my tablet. Tell me "stop" when you finished
with the tablet
u1: (stop) Ok, Do you want to do something else?
u2: (yes) ^goto(question4)
u2: (no) Bye!

proposal: %question4 I understand! Would you like to watch me make different moves?
u1: (yes) ^goto(moves)
u1: (no) Do you want to leave?
u2: (yes) Bye!
u2: (no) ^goto(question)

u:(["hi" "hello"] {_*}) ^goto(name)
```

Examples of Planning

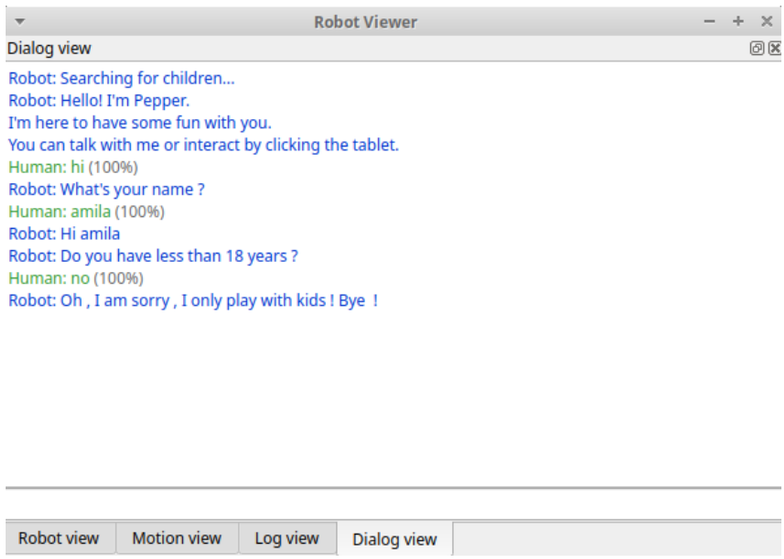

Figure 6: Planning in the case of answer 'no'

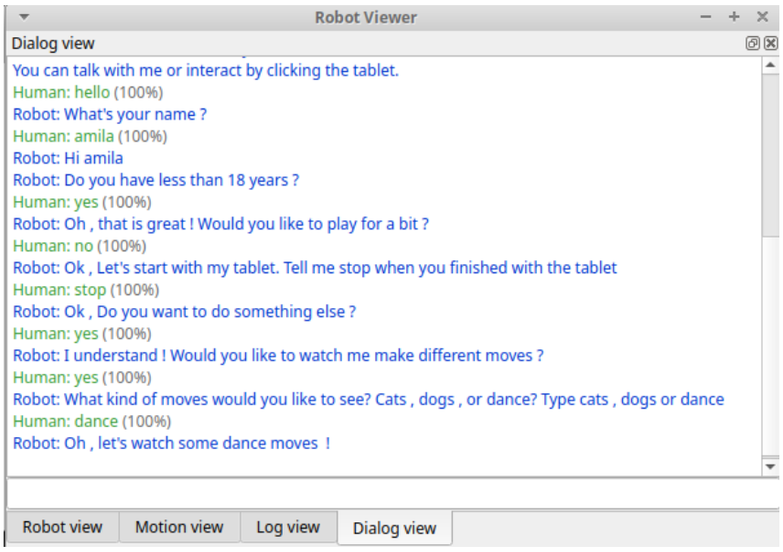

Figure 7: Planning in the case of choosing dancing moves

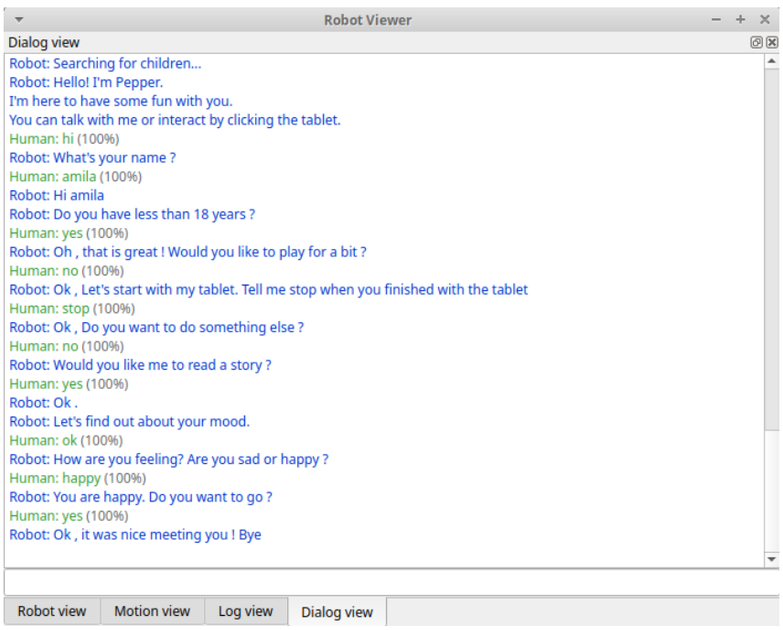

Figure 8: Interaction using two .top files

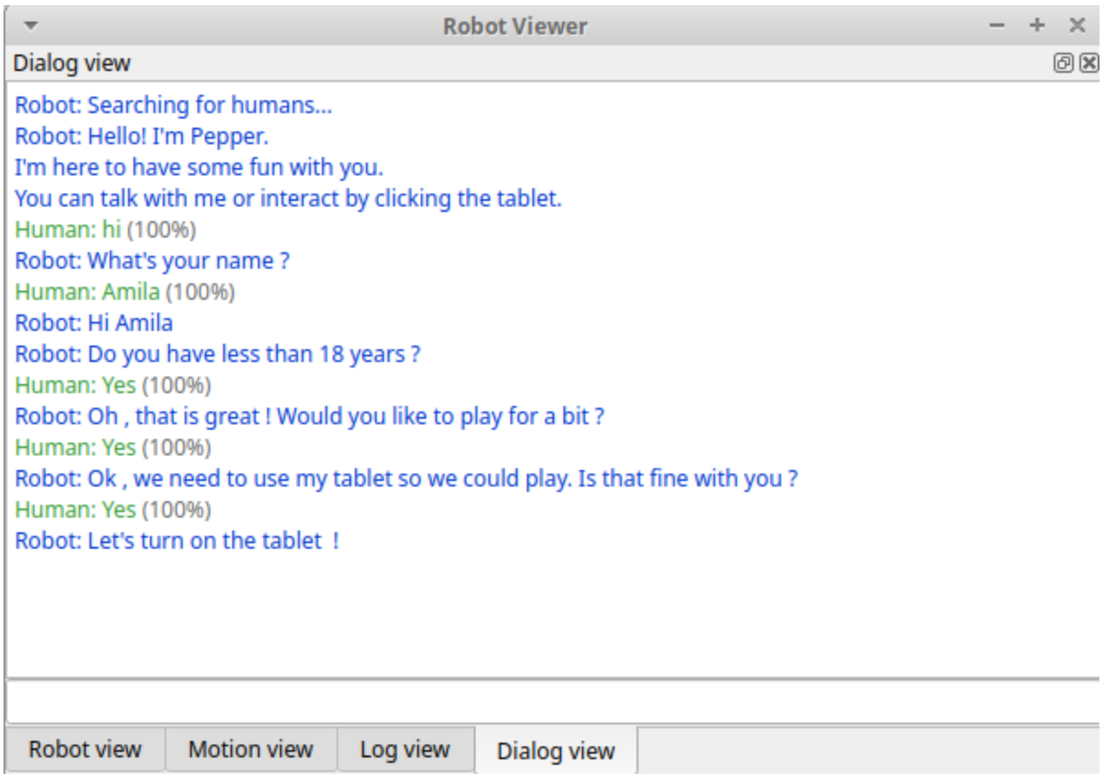

Figure 9: Starting the tablet view

4.3 Main code

The main code is given in *main.py* file and it implements the overall behavior of the system. Furthermore, some aspects of the robot behavior, such as gestures and touch control, are implemented in other Python files for improved readability of the code.

The following code segment implements interaction from the *main.top* file. Furthermore, we instance the gesture and touch control interfaces so they can be used within the application. The content of the *main.top* is given before and it implements the question/answer based interaction between the robot and a human.

```
ALDialog = session.service('ALDialog')
ALMemory = session.service('ALMemory')
ALMotion = session.service("ALMotion")
tts_service = session.service("ALTextToSpeech")
ALDialog.setLanguage('English')
topic_path = project_path + "/topicFiles/main.top"
topf_path = topic_path.decode('utf-8')
topic_name = ALDialog.loadTopic(topf_path.encode('utf-8'))
ALDialog.activateTopic(topic_name)
ALDialog.subscribe('pepper_assistant')
gesture = Gesture(ALMotion, doGesture, tts_service, )
touch = Touch(ALMemory)
```

After this, the code runs the *main()* function where the *ALDialog* service subscribes to the *LastInput* topic with *lastInputHandle* function called whenever the user input is received.

There are several things that this *lastInputHandle* does. First it handles different possible inputs from the user (different string entered in robot dialog simulator). The following code is used for re-running the *main.top* interaction file in a case that humans answers that he/she is not less than 18 years old after starting the interaction. This is due to the requirement that robot is only for interacting with children. Moreover, we start the tablet interface only after the contact with a human is initiated (by saying "hi"). This is what first *if* clause is doing, where *test_ interaction* function uses MODIM interface to send an initiation signal via websocket.

```
if "hi" in lastInput.lower():
mws.run_interaction(test_interaction)

if len(inputs_list) >= 3:
```

```
print(inputs_list)
if "no" in inputs_list[-1] and "hi" in inputs_list[-3]:
print("start over")
ALDialog.unsubscribe('pepper_assistant')
ALDialog.deactivateTopic(topic_name)
ALDialog.unloadTopic(topic_name)
topic_path = project_path + "/topicFiles/main.top"
# Loading the topic given by the user (absolute path is required)
topf_path = topic_path.decode('utf-8')
topic_name = ALDialog.loadTopic(topf_path.encode('utf-8'))
print(topic_name)

# Activate loaded topic
ALDialog.activateTopic(topic_name)

# Start dialog
ALDialog.subscribe('pepper_assistant')
tts_service.say("Hello! I'm Pepper.\nI'm here to have some fun with you.\
    nYou can talk with me or interact by clicking the tablet."+" "*5, _async
    =True)


if "cats" in lastInput.lower():
gesture.doGesture = True
print("doing cat motion...")
gesture.doCat()

elif "stop" in lastInput.lower():
for i in range(1000):
audio_player_service.stop(i)
gesture.doGesture = False
index += 1
```

Furthermore, if user asks a robot to imitate a cat or a dog or a dolphin, the code parses that input and provides the motion for the desired task to cheer the child up. To further implement the interaction between the robot and the human, we added another *.top* file.

```
if "stop" in inputs_list[-3] and "no" in inputs_list[-2] and "yes" in
    inputs_list[-1]:
print("go to story")
ALDialog.unsubscribe('pepper_assistant')
ALDialog.deactivateTopic(topic_name)
ALDialog.unloadTopic(topic_name)
topic_path = project_path + "/topicFiles/story.top"
# Loading the topic given by the user (absolute path is required)
topf_path = topic_path.decode('utf-8')
topic_name = ALDialog.loadTopic(topf_path.encode('utf-8'))
print(topic_name)

# Activate loaded topic
ALDialog.activateTopic(topic_name)

# Start dialog
ALDialog.subscribe('pepper_assistant')
tts_service.say("Let's find out about your mood."+" "*5, _async=True)
```

The content of this *story.top* file is given bellow.

```
topic: ~example-assistant2()
language: enu

concept: (myname) [""]

proposal: %first How are you feeling? Are you sad or happy?
u1: (happy) You are happy! Nice to hear. Let me read you some funny story so we
can laugh together. TEXT OF THE FUNNY STORY.
u1: (sad) Oh, you are sad. Let me read you one happy story to cheer
you up. TEXT OF THE HAPPY STORY.
```

```
u:(["ok" "fine"] {_*}) ^goto(first)
```

4.4 Pepper's Tablet

After the child chooses that he/she wants to proceed with Pepper's tablet, and the robot is connected with the .html file, the following window is opened:

Figure 10: Original html site for introduction

A child can choose among the following options: playing a game, reading a story, or playing a video. All these options have the same goal / interacting with the child, in order to help him overcome his/her issues. In the case of choosing a game, the window is opened, where the child can choose how to play the game: against the computer, or against doctor or somebody else!

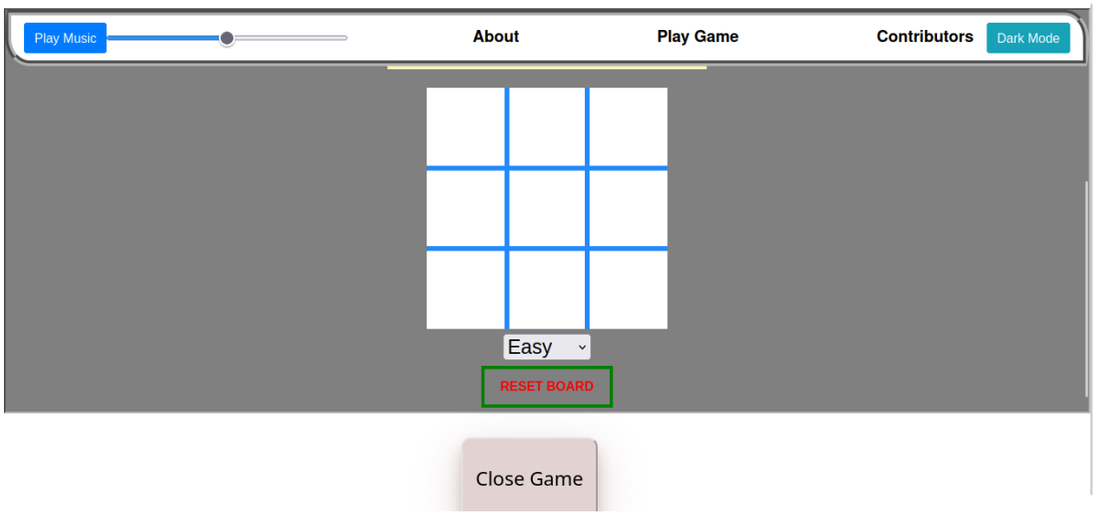

Figure 11: Playing the game

When the child is tired of playing game, he/she can choose an option to close the game, and continue doing something else. If they choose the option 'Read story' they get the following window opened (shown on picture below):

Figure 12: Page for choosing the story

On picture below is shown how one of the stories looks like. The child is able to scroll down, and read this story.

Figure 13: Pepper's story

The last option in the tablet part is watching videos. It is implemented fully using .html and .js files, and the following window is opened after choosing the option 'Watch video' in the main section. The child can choose between three different videos given in the list.

Figure 14: Page for choosing a video

In the picture below is given a view of one of the videos.

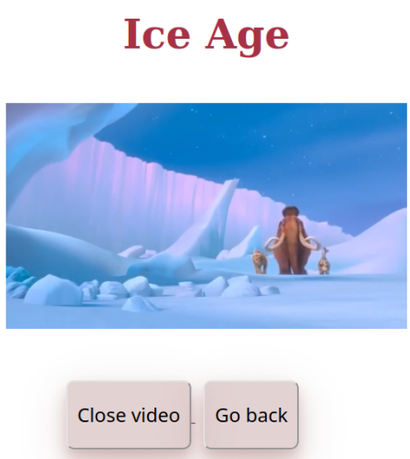

Figure 15: Pepper's Video Ice Age

4.5 Robot movements

The following code implements giving a standing and crouching postures predefined in NAO API.

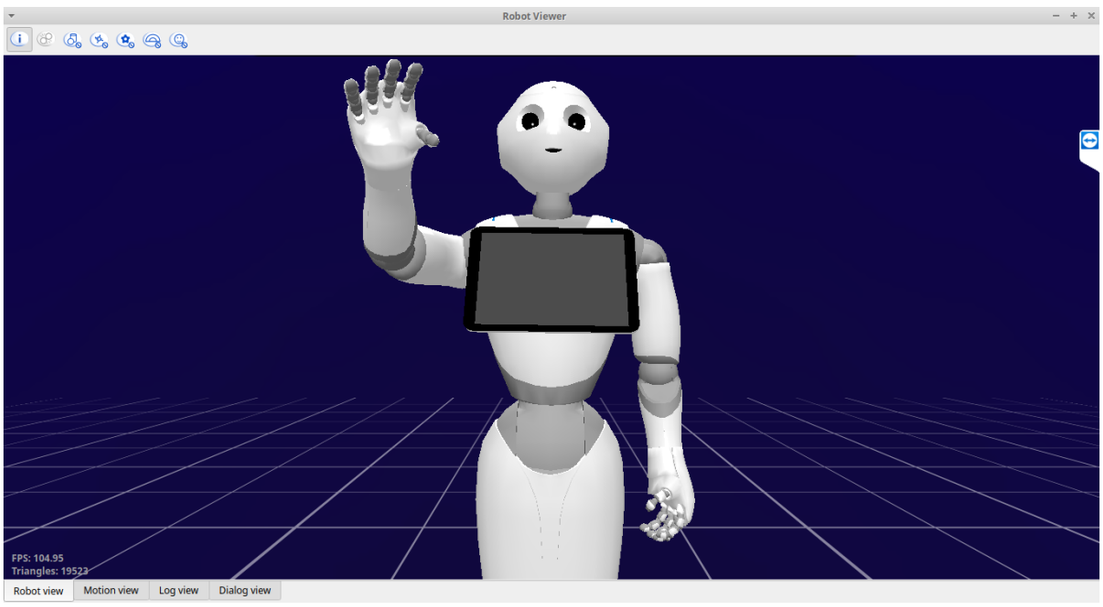

Figure 16: Pepper move - Hi

```
def stand(self):
self.posture_service.goToPosture("Stand", 0.5)
print("Robot is in default position")

def rest(self):
self.posture_service.goToPosture("Crouch", 0.5)
print("Robot is in resting position")
```

As mentioned before, when a child asks the robot to imitate a cat the following function is called. It implements a simple shoulder movement (please note that this is only for illustration purposes, the code to implement real cat like movement would require much more time to implement).

```
def doCat(self):
names = list()
times = list()
keys = list()
```

```
names.append("LShoulderPitch")
times.append([1.0, 2.0, 3.0])
keys.append([1.0, -0.2, 0.8])

names.append("RShoulderPitch")
times.append([1.0, 2.0, 3.0])
keys.append([1.0, -0.2, 0.8])

self.ALMotion.angleInterpolation(names, keys, times, True)
return
```

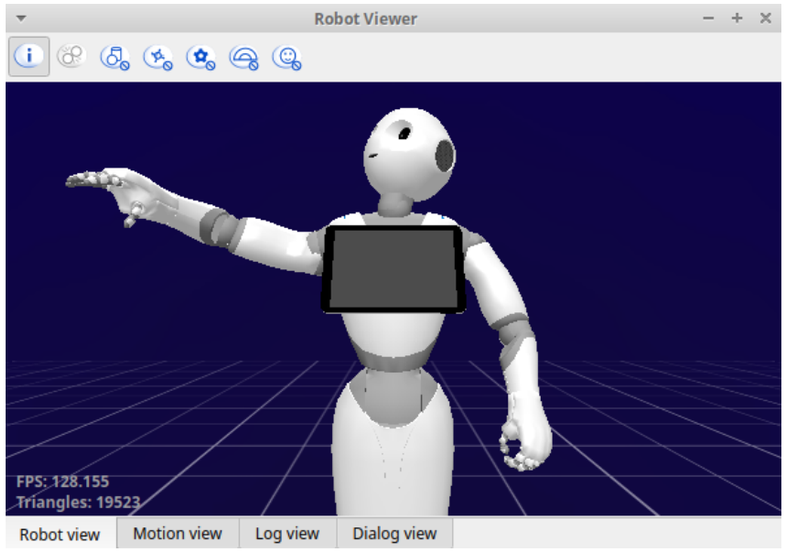

Figure 17: Dance move example 1

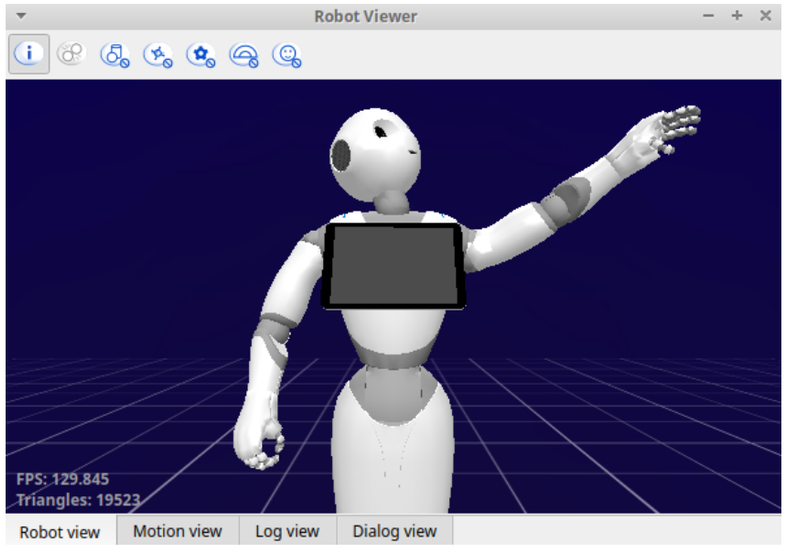

Figure 18: Dance move example 2

5 Final result

In this section, different interactions are shown. We would like to highlight that different interactions can be seen also throughout the whole report, especially in the Planning and Implementation chapters.

In the video, we can see starting of the project, and checking different scenarios. For example, cases of different planing scenarios, as well as story telling. Also, we can see how the child can interact with the table through its screen. Final result of the project can be found on the following youtube link:

https://youtu.be/OhmKaP3XdDY

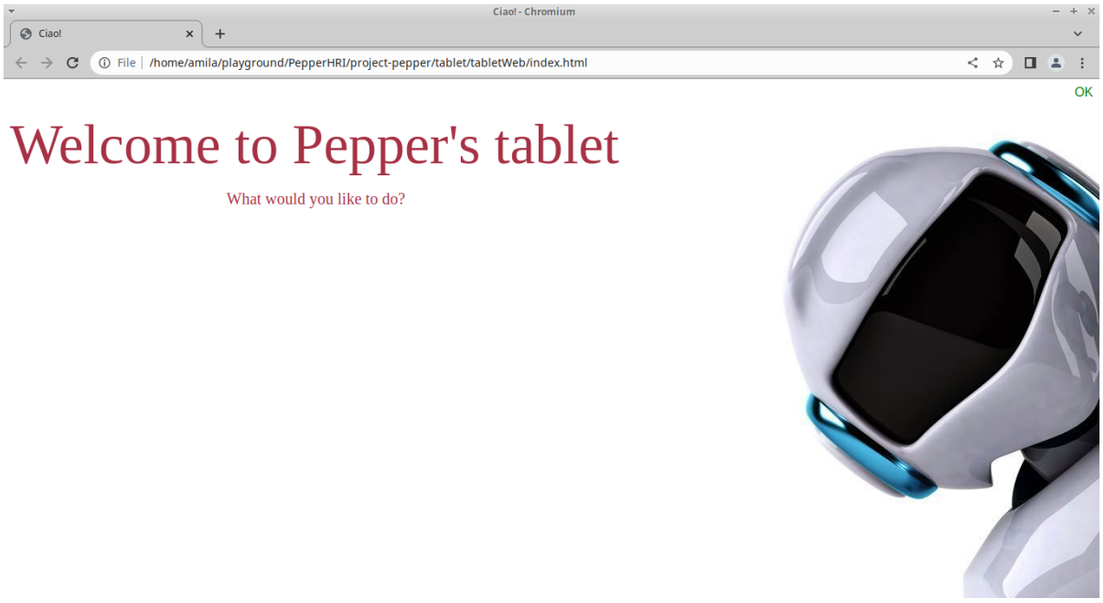

Figure 19: Buttons are hidden before human-robot interaction

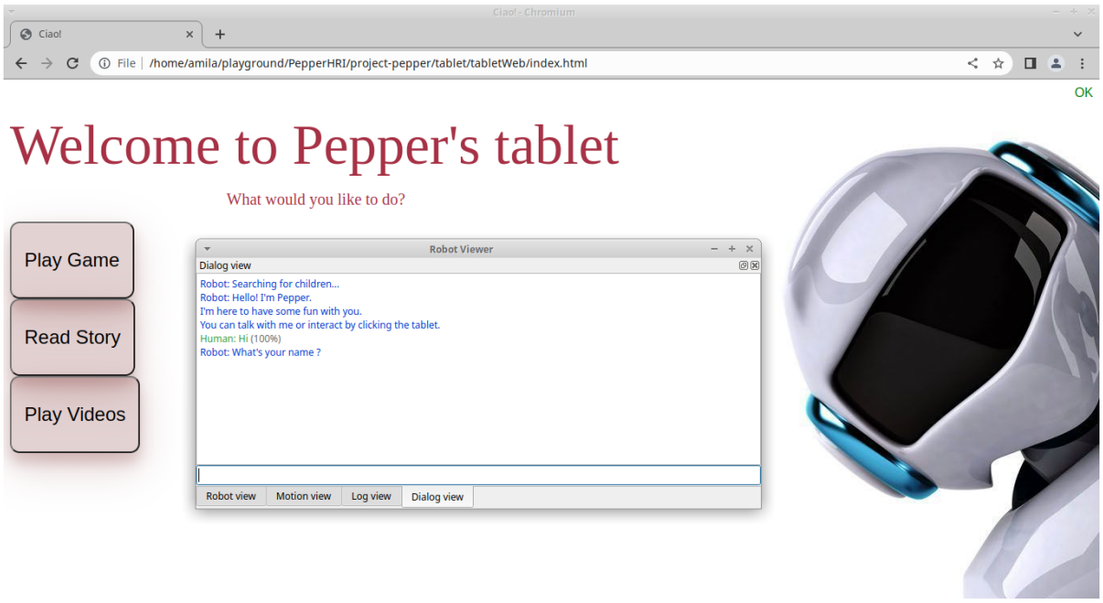

Figure 20: Buttons are shown after human-robot interaction

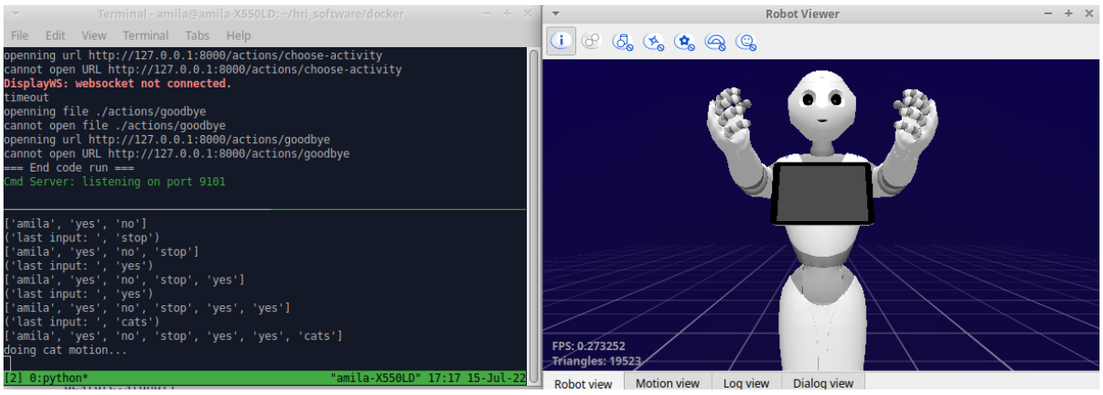

Figure 21: Interaction with cat motion - part 1

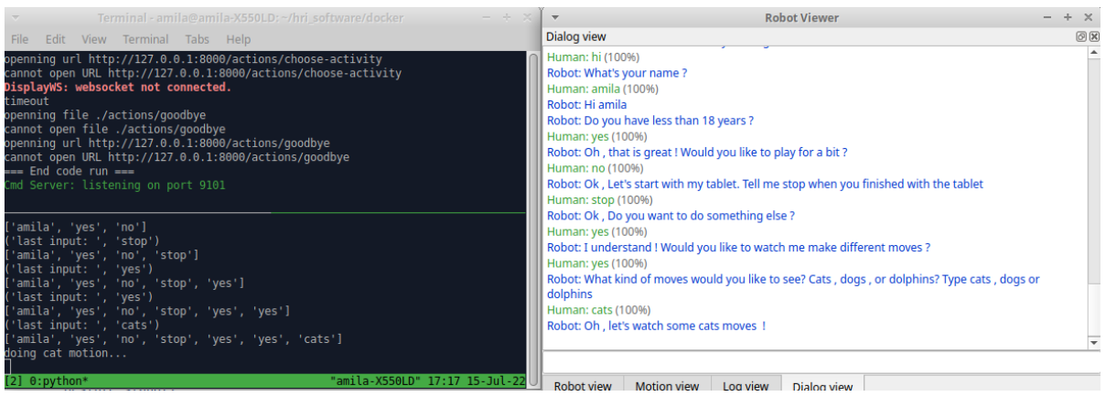

Figure 22: Interaction with cat motion

6 Conclusion

Interaction between robots and people are becoming more complex. This is due to both improvements in the algorithms in the fields of natural language processing, computer vision and control, and in psychology and sociology. The main purpose of this project is to allow interaction between the robot and children for the purpose of animating and improving their interaction. We used Pepper robot simulator and NAO API to implement this task. The code gives the robot basic capabilities to interact with humans, and specifically with children. However, only basic showcases of these interactions are implemented, since the more complex interaction is very time consuming to implement since there are many ways this interaction can go, but verbally and non-verbally. Due to such complexity, and lack of understanding of the sociology of these interactions, we skipped the implementation of some interesting parts, such as emotion recognition, non-verbal communication recognition etc. Finally, for future work, it would be very interesting to implement (and possibly research) some aspects of interaction between robots and children, since social skills of little children is rather basic, can be easily implemented for many scenarios, and this interaction can be very interesting in social sense to better understand first steps of human behavior.

Bibliography

[1] MICHAEL ARGYLE AND JANET DEAN *Eye-Contact, Distance and Affiliation.* Oxford University.

[2] *SoftBank Robotics documentation* Il robot Pepper, un umanoide a disposizione degli anziani e dei medici - Reccom Magazine

[3] Zhou, Y., & Thompson, S. (2008). *Quality assurance for interventional pain management procedures in private practice.* Pain Physician, 11(1), 43.

[4] Cohen, L. L., Blount, R. L., & Panopoulos, G. (1997). *Nurse coaching and cartoon distraction: An efective and practical intervention to reduce child, parent, and nurse distress during immunizations.* Journal of Pediatric Psychology, 22(3), 355-370.

[5] Klassen, J. A., Liang, Y., Tjosvold, L., Klassen, T. P., & Hartling, L. (2008). *Music for pain and anxiety in children undergoing medical procedures: a systematic review of randomized controlled trials.* Ambulatory pediatrics, 8(2), 117-128.

[6] Cassidy, K. L., Reid, G. J., McGrath, P. J., Finley, G. A., Smith, D. J., Morley, C., ... & Morton, B. (2002). *Watch needle, watch TV: Audiovisual distraction in preschool immunization.* Pain Medicine, 3(2), 108-118.

[7] Tanya N Beran, Alex Ramirez-Serrano, Otto G Vanderkooi and Susan Kuhn. *Humanoid robotics in health care: An exploration of children's and parents' emotional reactions.* Journal of Health Psychology 2015, Vol. 20(7) 984–989.

[8] *Social robots to make kids' physiotherapy more fun | TheMayor.EU* https://www.themayor.eu/en/a/gallery/social-robots-to-make-kids-physiotherapy-more-fun-8563?item=0

[9] Sveva Pepe and Simone Tedeschi. *MARIO: Pepper assistant for patients with senile dementia.* Sapienza, Human-Robot Interaction.

[10] Julia Dawe, Craig Sutherland,Alex Barco,Elizabeth Broadbent. *Can social robots help children in healthcare contexts? A scoping review* BMJ Paediatrics Open

[11] B. Robins, K. Dautenhahn, R. te Boekhorst, and A. Billard. *Robotic Assistants in Therapy and Education of Children with Autism: Can a Small Humanoid Robot Help Encourage Social Interaction Skills?* Adaptive Systems Research Group The University of Hertfordshire, School of Computer Science College Lane, Hatfield, Hertfordshire AL10 9AB, U.K

[12] *SoftBank Robotics documentation* http://doc.aldebaran.com/2-5/index.html