Pepper in hospital: multitasking management

An info point robot with a night supervision role

Nicolò Mantovani **1650269**

Lorenzi Flavio **1662963**

Human Robot Interaction

Both students contributed equally to the project

Github link: https://github.com/FlavioLorenzi/Human_Robot_Interaction-Pepper

Abstract

Nowadays, the incessant search for progress aims to improve the human condition, making deep changes to the society in which we live and, in the case of Robotics field, aiming at the insertion of service or social robots into everyday life. Service robots are technical devices that perform tasks useful to the well-being of humans in a semi or fully autonomous way. Recent studies have demonstrated that their inclusion could in the future bring great improvements to our society, helping in those "delicate" sectors where today there are still difficulties, like hospitals, factories or any dangerous-to-humans area. In this work we will see the insertion and programming of a Pepper robot in a healthcare environment and in a simulated way,aiming to contribute to research in this important field.

Contents

1. Introduction

1.1 Idea of the project and purpose

In this work we tried to focalize on several aspect of the Human Robot Interaction field and its importance in social environments, especially when a service robot is inserted in a healthcare environment with the aim to contribute to the improvement of hospital service, bringing enormous help to the whole medical staff and becoming part of it.
In these cases it is important that the human-robot relation is well defined smoothly, such that humans can get the maximum benefit from this interaction. Therefore, the robot must be programmed to ensure smooth operations most of the time and always be able to offer what it was programmed for, thus embodying its role to the best.
The robot implemented in this project is Pepper (Fig. 1), a semi-humanoid robot manufactured by SoftBank Robotics (formerly Aldebaran Robotics), designed with the ability to read emotions, interact with humans and much more.

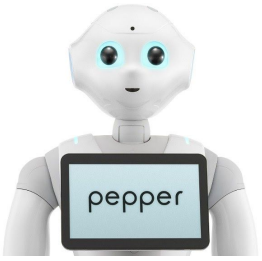


Figure 1: Pepper robot

Our idea has been to insert Pepper in a hospital, assigning it an "**info point**" role: one can asks informations about doctors, patients, hospital departments and other useful informations, but also to play whenever someone is bored.
The interaction occurs in different ways: through the tablet (placed on the robot) it can be seen each written sentence or question, and it's simply to interact thanks to the available interface with buttons and images. Moreover, it is possible to talk with robot thanks to the automatic speech recognition function, always active for voice recognition when the user asks some questions.
As additional implemented task, during the night Pepper could be placed in the corridors of a specific department in the hospital, where it wanders as a "**supervisor**": if it finds patients out of their bed, it blocks them and asks how they feel. Furthermore, if someone shouts for help, it can also intervene and call the medical staff as soon as possible. So the robot is able to understand situations and give advice depending on patient behaviour of a specific moment.

1.2 Overcome difficulties: importance of our work

Our project is developed entirely in a simulated environment, due to the health emergency that the world is facing right now (with Covid-19) which made it impossible for us to have direct experience with the real robot. However this problem was easily solved using some important tools which lead us to obtain acceptable results directly on our laptop.
First of all we handled with NAOqi, an *SDK* official remote service that allows to connect us to the robot (the same used with real robot) and to use all its predefined functions to perform our work. For instance, we can use the sonar function of the robot contained in the "pepper_tools" folder, which allows us to simulate someone's perception changing sonar values, with the *"sonar_sim.py"* script.
Then, MODIM server was used to run interactions with both real and simulated robot, by using some predefined functions (Automatic Speech Recognition and so on..) and observe their execution when possible. This can be done in a simulated way just by running an external web server ('nginx') that is able to link Pepper touchscreen with our localhost and by plotting its content on a web page of our laptop: in this way we can see in real time some Pepper functions (especially those related to the tablet).
All these instruments have allowed us to perform our work in a totally homemade way, but working a little with fantasy with the feeling of being together with the real Pepper.

Working on this project made us understand how much service robots and HRI are important today; in particular in a **healthcare environmen**t where there are people who really need help (patients or visitors too), a service robot can make excellent progress and lead the system to better results. Indeed it brings a lot of advantages: as we said before, it can be used like an info point robot without never being tired (providing and ensuring maximum efficiency), but in general it can be used in several types of roles, especially in this period where the risk of contagion is high and a robot could move freely and interact with patients without any problem. Hence, HRI research needs to be constantly improved and in our small way contributing to it is stimulating and fascinating.

2. Related Work

Although two papers concerning the human-robot interaction in the healthcare system have been the most inspiring ones for us, this section concerns previous works and scientific publications in HRI field which have influenced the overall project, transmitting us ideas and different points of view.

"*Increasing trust in human–robot medical interactions"* [1] is one of the main references for our project: it highlights the importance of transparency and adaptability conditions for social robots, conducting several laboratory experiments.
Transparency is a very complex phenomena which includes on the human side the ability to understand the robot's behavior, also having its performance or failure rate in advance; on the robot side instead for each movement transparency implies the utmost clarity of intent, also announcing its moves before carrying them out.
In this work it was tested the ability of a service robot (Care_O_Bot 3) in a healthcare environment, with the aim to perform a blood pressure measurement over 85 people of any age. These experiments showed that if the robot switch on the two conditions described above, it benefited on the patient's trust and comfort. In particular it turned out that transparency made possible the creation of a certain human-robot bond, transmitting a greater sense of general safeness, because the human believed to have control over the situation.
So transparency concept has been taken up in our work as one of the key concepts: especially to make complete the role of infopoint where the robot kindly asks what the user needs anytime during the interaction (e.g. "I'm done here, do you want to know more?"). Furthermore, as long as the user needs something, the robot remains at his disposal with the maximum clarity of intent.

"*State Of The Art: A Study of Human-Robot Interaction in Healthcare"* [2] is the second main reference: it proves the increasing importance of the employment of the robot technology in the healthcare system, due to natural consequences such as the high cost of the healthcare and the lack of healthcare professionals, taking into account all the challenges that must be faced out (both ethical and usability). The paper shows several task that are carried out by different healthcare robots (surgical, rehabilitations, companion etc…) and their related challenges. For instance, consider privacy issues that a telepresent robot could trigger without having the right permissions (spreading informations to people who are not involved in the care of a patient) or a deterioration in a patient's health due to a failure or a malfunctioning of a robot during the healthcare delivery process. The paper shows also that a key role is played by the acceptability of users, that is the willingness of people to integrate a robot into an everyday social environment, which is strongly conditioned by their culture (think about how robots are differently seen by European and Japanese).
Other works that inspired us are [7] and [8]: the former is about Tega, an expressive and child friendly robot designed for long-term deployment in various educational settings, such as

children's homes, school and therapeutic centers whose task is to model children's mindset in such a way that it can be considered as a "growth mindset", since mindset has been shown to have a large impact on people's achievements; the latter focuses on the recognition of the six human facial expression (happiness, anger, disgust, sadness, fear, surprise): the capability to understand human emotions is indeed crucial for a robot in order to properly interact with human beings.
We notice that in several papers [1][2][9] there is a final questionnaire, about the experiment just carried out. In particular, it was interesting to observe the use of questionnaires as a form of self-assessment and improvement [1]. Thanks to a very short and simple questionnaire, it is possible to have an average of people's thoughts about the robot: its reliability, its behavior and to know if these people felt fully at their ease in front of it. Therefore, we took all these notions and inserted a final questionnaire after the infopoint service: it could be a very good instrument for us to receive a feedback of our work and in case of bad opinions, understand whether and where we went wrong and improve for the future.

In addition to those mentioned, several papers have been read deeply to find inspiration and we have to say that each of them has contributed in its own way to the final development of the project, providing us with new knowledge on this important area of HRI. In particular these papers have helped us to better understand the importance of this area in the real world, applied in various social realities such as "children and autism" [4], "development and fun with AI" [5][8] or "perception of human feelings" [6].

In the end we have the video "*Towards Robotic Systems for danger recognition and alert"*[3] that shows Pepper in a private clinic with an intelligent supervisor role; it can recognize if a patient feels bad not only through voice commands (answer and questions modality), but also if he is passed out on the ground thanks to a cutting-edge recognition system, immediately calling for help. This and many other related videos [11][12][13] about service robots influenced our thought, especially about Pepper in hospitals and the additional *supervisor script* developed in our work.

3. Solution

Software components

Our work is made possible thanks to the use of several tools: some of these belong to the official Pepper system, while others are used mainly to work remotely (designed for simulations) with the robot.
The development environment is mounted on the Linux distribution Ubuntu, where with the open-source Docker software it is possible to create one or more isolated containers in which we can work independently without changing or overloading the system. This is possible by providing an additional abstraction through an operating system-level virtualization: we just need to build a docker image (with all the predefined libraries) and run it to go inside the robot architecture. The entire project runs on python code which is enabled to communicate with each part of the system and with html/css files too.
The Software Architecture on which our project is built up is the following:

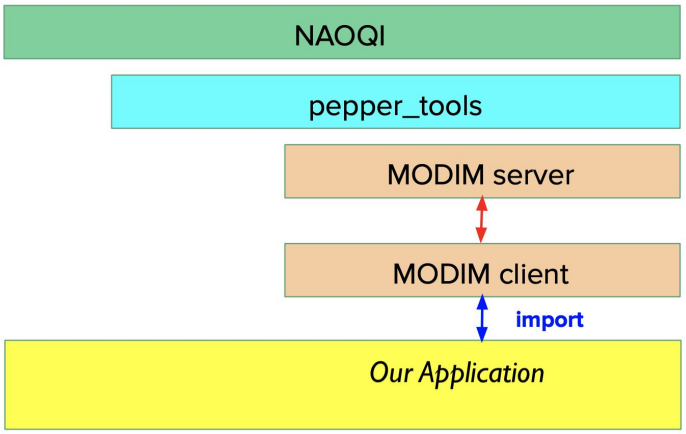


Figure 2: Software implementation: NAOqi dependencies

Pepper's operating system is **NAOqi**, that is an embedded GNU/Linux distribution specifically developed to fit the SoftBank Robotics robot needs. It provides and runs numbers of programs and libraries, among these, all the required one by NAOqi, giving life to the robot [10]. This is the base on which is built most of our software architecture (Fig.2).
Only by running NAOqi server it is possible communicate with a simplified version (given to us a priori) of tools -- ***Pepper_tools*** -- and make them work just importing *pepper_cmd* in the code. Furthermore we handled with **MODIM** service, a system that ensure a set of services for the robot, both for real and simulated one (for the real it's enough to change the ip address respect to our work). MODIM works thanks to the definition of some interaction functions, which are available pre-implemented ready to be inserted into the code and called.
There are two ways to proceed here: the first is "*by direct call*" where each function can be called directly using **im.executeModality('INTERACTION', 'user need')**, by specifying the interaction needed (e.g. TEXT_TITLE) and what we want to perform (e.g.'HRI_project').

The available interactions are the following : TTS (*Text To Speech*) that allows the robot to speak, ASR (*Automatic Speech Recognition*) that allows a speech interaction with the robot, TEXT_DEFAULT to write something on the touch screen, TEXT_TITLE to write something as title on touch screen, BUTTONS to generate buttons on touch screen layout, IMAGE to upload whatever image we need in the center of the touch screen.
The second way is "*calling actions*", where with simple commands like **im.execute('action')** or **im.ask** we can work in an indirectly way with 'actions' implemented separately.
An example of action is the following:

```
TEXT_title          #first MODIM interaction function
<*,*,it,*>: Alt
<*,*,*,*>:  ALT
----
TEXT_default        #second MODIM interaction function
<*,*,it,*>: Oi tu, FERMATI!
<*,*,*,*>:  Eih you, STOP!
----
...new...           #new MODIM interaction function
```

Each action can be characterized by any MODIM predefined function, as we want. Note that with actions we can also define the user profile changing the spoken language for the current interaction ('it' stands for italian, while english is by default) and this could be a very important feature if we place Pepper, for example, in a multicultural environment.
In our project we have included both these methods: the result is the same, but with the second, the code is more readable and ordered since it is only another way to run commands.
Moreover MODIM lean on *pepper_tools* (both run on NAOqi), so each Pepper predefined function (sensorValue, say, raiseArm, takePhoto…) can be called with *im.robot.function*().
Finally MODIM allows to render on our web-page (*im.displayLoadURL()*) everything we have implemented about its touchscreen, having in this way a feedback of what we just done.
However to do this we need another important tool: **Nginx**, a web server that connects our localhost html page with our software architecture. Thus, it is precisely here that the importance of the developed html code takes over, in which it is possible to define the aesthetic layout as we like.

To conclude this section, it is important to show how each part we've discussed before is combined together contributing to a very efficient mechanism. We have to type in the terminal step by step the following commands:

Nginx server running:

```
% cd hri_software/docker & ./run_nginx.bash
```

Robot activation (entering into the docker image, *robot*):

```
% cd hri_software/docker & ./run.bash 0.4.2
```

And once we are into the robot:
-Launch NAOqi server

```
%cd /opt/Aldebaran/naoqi-sdk-2.5.5.5-linux64 & ./naoqi
```

-Launch MODIM server for Pepper

```
%python ws_server.py -robot pepper
```

-Pepper Tools for simulations (e.g. ASR and front sonar):

```
%cd src/pepper_tools/asr
%python human_say.py --sentence xxx

%cd src/pepper_tools/sonar
%python sonar_sim.py --value xxx --duration xxx
```

So in this manner it is possible to combine each part and make the simulation work somehow.

4. Implementation

4.1 A multitasking management

In this section there is a detailed explanation of the practical part and implementation of all our work; in particular, two folders divide our project:

- **html:** the container of all the layouts of the several html pages, css files, images and javascripts;

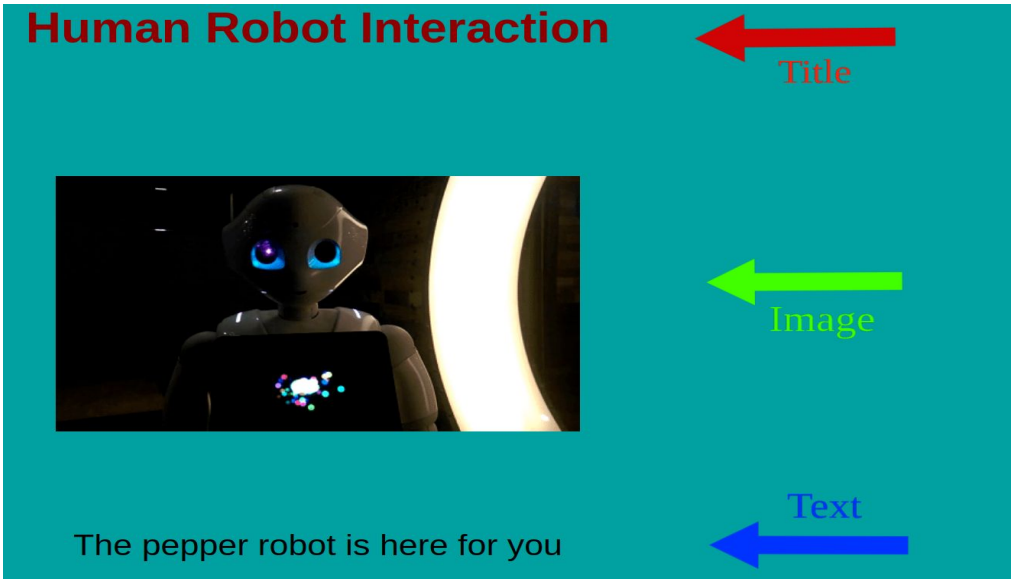
- **modim:** the core of the project. Both python scripts and predefined actions can be found.

Actually, two python scripts have been created and they are:

- ***"infopoint.py":*** during the day we consider Pepper as stuck in the reception interacting with people;

- ***"supervisor.py":*** during the night Pepper is able to wander inside the wall of the building (we assumed the map is given) to check if someone is getting sick or need some help.

For what concern the layout the scripts, each of them has a different *html* file, respectively ***"layout_infopoint.html"*** and ***"layout_supervisor.html"***. Furthermore, we have moved the internal styles of the html to different CSS files (put in a folder named "*css"*), respectively ***"style_infopoint.css"*** and ***"style_supervisor.css"***. Since we also have created a questionnaire to check users' experience, we gave it a different aspect through another layout and another css file, ***"layout_questionnaire.html"*** and ***"style_questionnaire.css"***.
Basically, all the *html* files share the same common structure :



- Title: it is placed in the top part of the webpage;
- Image(s): in the middle;
- Text: always below the image(s);
- Button(s): they are not required in all the interactions and are placed below the text.

Furthermore, in all the pages there is a small script that connects the pages to the MODIM server. The *css* files are of course different from each other since the style is different from page to page and we can access to the different html elements style either by using *'#id'* if they have been created with the *'id'* property or by *'.class'* if have been created with the *'class'* property.
Before showing how the scripts work, we need to mention that even if we didn't have the possibility to work with the real robot, we anyway worked like if we would have it, meaning that interactions like *TTS* and *ASR* have been used. For time reasons it was not possible to implement a simulation for the TTS function, therefore in the code this is always flanked by a TEXT_DEFAULT function, just for visual feedback.

4.2 Technical details

The first of the two python script is composed of three interactions (defined as methods):

1. *"**hellothere":*** the aim is to sense people that are within robot's engagement zone (*im.robot.startSensorMonitor()*). We used front and rear sonar (threshold between 0.5m and 3m) and touch sensors to perceive people around it; once someone is recognized, the html file *"layout_infopoint.html"* is forthwith launched on the screen of the tablet (*im.robot.loadUrl()*) and the person is welcomed through a predefined action (*im.execute('welcome')*). Then the robot introduces itself and prepares for the next task (*'Hi, I am Pepper and like my fellow doctors I am here to help people'*);

```
def hellothere():
        im.init()
        im.robot.startSensorMonitor()
        sensors = im.robot.sensorvalue()
        frontSonar = sensors[1]
        rearSonar = sensors[2]
        headTouch = sensors[3]
        if ( frontSonar > 0.1 and frontSonar < 3) or (rearSonar > 0.1
        and rearSonar < 3) or headTouch:
                im.display.loadUrl('layout_infopoint.html')
        [...]
```

2. *"**infopoint":*** this interaction is basically the body of the script. A big loop is started to allow our robot to be always working and is asked to the user which kind of help (let's define it as a subtask) is needed. Another loop allows the user to stay in the same kind of subtask (i.e.: see information about hospital), unless it explicitly press

the button “no” when the subtask is ended and it is asked to the user if it is needed to perform again that subtask. We have defined the following subtasks:

- *“Looking for a patient”:* who has never gone to visit a relative or a friend and did not have any issue in finding him? Pepper has several databases (actually python dictionaries, not real ones): match people and places where they hospitalized (DATABASE_WHERE), how to get a specific pavillion (DATABASE_GET_TO_PLACE), surname and name of people hospitalized (DATABASE_SURNAME_NAME), and one for the password (DATABASE_PSW). An example:

```
DATABASE_WHERE = {"lorenzi":"urology","mantovani":"psychiatry",
"nicosanti":"cardiology","roberto":"urology","john":"[...]"}

DATABASE_GET_TO_PLACE = {"urology":"If you don't know how to
get urology pavillion, just get out of where you entered. Then
turn left and on the left you will see a big blue building.
That building is urology.","psychiatry":"If you [...]"}

DATABASE_SURNAME = ["lorenzi","nicosanti","mantovani"]
DATABASE_SURNAME_NAME={"lorenzi":'Flavio',"mantovani":"Nicolo",
"nicosanti":"Gabriele"}
DATABASE_PSW = ["nm95","fl95","gn95"]
[...]
```

  The password database has been developed in order to deal privacy issues (i.e.: consider a patient who want to be found just from a restricted group of people). The user is asked the surname (both through the tablet and TTS) of the person and then, if the latter is in the hospital through a quick check in the database, to insert a password to unlock the chance to know where his friend/relative is currently located and it is shown how to get it. If the password is wrong the user is asked to try again;

- *“Looking for a doctor”:* in this section the user is asked for which doctor is looking for (both through the tablet and TTS). Some doctors are fixed somewhere in the hospital and others can be called by Pepper (not a real call at the moment);

```
im.executeModality('IMAGE','img/madrob.jpg')
im.executeModality('TEXT_default','Who are you looking for?')
im.executeModality('TTS','Who are you looking for?')
im.executeModality('BUTTONS',[['Doctor Burioni','Doctor
Burioni'],['Doctor Merlini','Doctor Merlini'],[...])
[...]
```

- *"Info about hospital":* here Pepper is asked to show how the hospital is currently structured. Its map is shown at the center of the screen and the different buildings are displayed below. To do this we worked especially with *im.robot.say()* function and *im.executeModality()* with IMAGE as parameter.
  In the end, given the period we are facing now, Pepper reminds us of some useful guideline to counter the diffusion of the Coronavirus, by rendering an image with some practical tips.
  A practical example is:

```
im.executeModality('IMAGE','img/tips.jpg')
im.executeModality('TEXT_default','Remember some useful tips to deal with this particular period')
im.executeModality('TTS','Pay attention')
[...]
```

- *"Info about Pepper":* this section has been created to allow the user to read some basic information about pepper. If further informations are requested, it is also displayed on the tablet the link that redirects to the SoftBank webpage. Its structure is very similar to the previous subtask one, with the difference you can decide to stop the robot, by interacting with buttons.

```
while true:
    im.executeModality('TEXT_title','Here you can read my story ')
    im.executeModality('TEXT_default','I am Pepper, one of the first social humanoid robots able to [...] ')
    im.executeModality('TTS','Here can you read my story')
    [...]
    im.executeModality('TEXT_default','In this hospital[...]
    time.sleep(20)
    im.executeModality('TEXT_default', 'Then, during the night I am also on the first floor, where [...]')
    [...]
```

- *"Fun with Pepper":* our starting idea was to redirect the user to a real Internet webpage and entertain him with a game. Unfortunately, we got several issues to redirect (loss of original Pepper interface) and then go back to the script, thus we decided to put some riddles instead of a real game, quite short and simple. The user has 10 seconds to reply (*im.ask(actionname=None, timeout=10)* and can digit between three choices through the available buttons (e.g. *im.executeModality('BUTTONS',[['roma','Roma'],['lazio','Lazio'],[...])*); for each answer Pepper replies with "very good" or "not good" (using TTS and TEXT_Default functions). If the user achieve the maximum score, the robot asks him to shoot a photo (*im.robot.takephoto()*), if the user wishes.
  Following there is a brief example of this:

```
im.executeModality('TEXT_default','Lets shoot a picture to the campion: say OK to accept. . .')
```

```
im.executeModality('ASR',['ok','no'])
g = im.ask(actionname=None, timeout=35)
if g == 'ok':
        im.executeModality('TEXT_default','Taking the picture
        [...]
        im.robot.takephoto()
[...]
```

In the end Pepper exclaims "*Game Over*" and asks if you want to play again.

Finally, when the nth subtask is ended and the user was helped, Pepper asks for a personal feedback of the interaction: "*Sorry, are you satisfied of my services?*". The user can reply yes or not (through buttons or voice) and the robot will act accordingly. If the opinion is negative, Pepper invites him to dwell on a short questionnaire.

3. ***"questionnaire"*:** once the infopoint interaction is terminated, the user is asked to fill a questionnaire in order to allow us to improve Pepper's skills. A new html page ("*layout_questionnaire.html")* is displayed, Pepper's eyes became green (*im.robot.green_eyes()*), the head of the robot lowers and turns towards the screen (*im.robot.headPose(0,25,0)*) and some questions are demanded.
   A score from 1 to 5 can be given to each question thanks to five implemented buttons, where 1 means *absolutely not (satisfied)* and 5 means *absolutely yes (satisfied)*; a mean of the scores is performed and a last question, weighted more with respect to others, is asked. If the total score is greater than 3 we conclude the interaction with the user was fine and disastrous otherwise. Once the questionnaire is concluded, the user is thanked and greeted.

Now, we move on the second script of our work *"supervisor.py"*. where we chose to implement short but very important interactions: Pepper is alert at night in the hospital being able to move through the corridors (maybe following a autonomous walk algorithm - not really implemented). Note that in this implementation we often used the indirect calling of "actions", with *im.execute()* command.

Pepper is able to perform three types of subtasks:

- an ASR function is always listening and if someone screams for "help" the robot immediately call the medical staff (*im.execute('nurse')* or *im.execute('doc')*);
  "Nurse" and "doc" are two actions, defined separately in a specific folder (actions) in a very similar manner: each action has a TEXT_TITLE (e.g. "*Alert Alert Alert: need help here"*), a particular IMAGE_Default, a TEXT_Default and a TTS function (e.g. "*Keep calm, I am calling for assistance. A doctor will be here in a minute!"*);

- two emergency buttons are always active; by pressing them you can skip the interaction with the robot and call doctor or nurse (their implementation is the same shown above with *nurse* and *doc* actions);

- if Pepper senses someone in front or behind it, thanks to the sonar sensors (*sonar = im.robot.sensorvalue()*), it immediately blocks him by raising his hand (*im.robot.raiseArm ('R')*) and asking for explanations. It starts then a brief interaction through BUTTONS and ASR function where depending on the patient's responses, Pepper can launch four different actions.
  Patients can choose between predefined answers such as "*I can't sleep*" or click on "*other*": the robot will give some tips on how to deal with the night or call for help, if the problem is serious.
  Following, there is a piece of code relating to the latter situation, where if you click on "*I am sick*" then Pepper asks "how bad you feel" and acts in an appropriate way:

```
im.executeModality('BUTTONS',[["I  am  sick","I  am  sick"],["I  can\'t
sleep","I can\'t sleep"],["I was thirsty","..."],["other","..."]])
im.executeModality('ASR',["I am sick","I can\'t sleep","..."])
c = im.ask(actionname=None, timeout=20)
if c == "I am sick":
       im.executeModality('TEXT_default','How bad you feel? ')
       im.executeModality('BUTTONS',[['1','Alittle'],['5','So-So'],
       ['10','Too much']])
       ...
       if b=='1':
              im.executeModality('TEXT_default','Take a moment and..')
       ...
       else:
              im.execute('doc')
[...]
```

As we already said Pepper is not only a "sentry" calling for help. If the problem is not serious the robot can provide some tips: for instance if the patient click on "*I am thirsty*" so the action tips2 will be executed, in which the robot reassures the patient with sentences like "*near your bed there is the water bottle;now go back to sleep*" and so on, being able to give good advice.

5. Results

In this section we tried to show and comment the most important results we achieved, focusing mainly on the key aspects and the different interactions/behaviours.
Note: for each performed interaction there is a video on the following youtube channel:
https://www.youtube.com/channel/UCsKJESLHc5omHOMhgsrGAcw?view_as=subscriber

5.1 Infopoint

Imagine entering a hospital, the Sacred Heart, because we need some help and all the staff is busy. Thus, we go to the info point and find Pepper (video **HelloThere**). If we approach it or just only touching its head (min. 0:02, we simulate a person in front of it who is perceived by the front sonar sensor), it begins to speak *"I am Pepper and like my fellow doctors, I am here to help people*". The robot proceeds then with greetings and asks if some help is needed, giving the opportunity to respond through two buttons: if "No" is clieck Pepper greets the user (min. 0:25) otherwise the main page will open with all the fields in which it can offer some kind of help. The interactions are the following:

- the "*looking for a patient*" interaction is the longest one (video **Patient**). Pepper asks for the patient's surname (min. 0:26) and for the correctness (min. 0:30). If it's wrong, it apologizes (min. 0:34) and re-asks for the surname (min. 0:40). Once it is sure about the surname, the user is asked for a special password (in our case the psw is composed of the initial of name and surname and the year of birth, i.e.: *"nm95"* for Nicolò Mantovani born in 1995): if it's correct Pepper shows the location (min. 1:06) and how to get him (min. 1:12) otherwise it asks for the psw again (min. 2:04). Finally (min. 3:19), it is shown how Pepper deals with people who are not in the hospital (we've inserted as surname "Rossi", which is not present in the database);

- if a doctor is needed (video **Doctor**), it's possible to choose among some doctors available (min. 0:18). The first doctor is called by Pepper (min. 0:21), for the second doctor it is shown how to reach him (min. 0:39);

- Pepper is able to show you some basic information about the hospital (video **About Hospital**) and its buildings. It also remind people some useful tips to follow during this special time period (min. 1:02);

- if anyone is interested to see some generic information about Pepper (video **About Pepper**), it is possible to click on the button *"About Me"*: Pepper will start then to talk about himself and about some curiosities (from min. 0:14);

- if we get bored, maybe waiting for someone or something, we can type on "*Fun with me*" and play with the robot and as we have already said, a short quiz will start on Pepper touchscreen.
  In the video **Quiz** we can see that questions are very simple and concern general topics (History, Soccer, Music, Cinema) and we can choose between three alternatives, interacting with buttons. Pepper warns we have only 10 seconds (min. 0:14) and each time we give a correct answer it replies with "*very good*", else with "*not good*"; if we achieve the maximum score (min. 0:52), the writing "*you are a master*" appears and Pepper asks if we want a photo to remember the win; to answer we can say just "*ok*" (min. 1:06) with ASR simulation. When the game ends, the Game Over layout appears, asking us to play again. If we type no (min 1:19), there is the exit that takes us back to the home screen of the infopoint;

- finally if there is no more need information, Pepper asks if we are satisfied of the its services, so typing "yes" it thanks and greets us, while typing no (video **Questionnaire**) it invites to fill in the questionnaire (Fig. 6), which will open shortly. So we can accept or decline the request of a personal feedback. If we accept (typing ok) questions like "*how reliable am I, from 1 to 5 ?*" (Fig. 7) are asked and in the end Pepper evaluates our judgment (as positive or negative) thanking us. In the video we gave a positive feedback (min. 1:46).

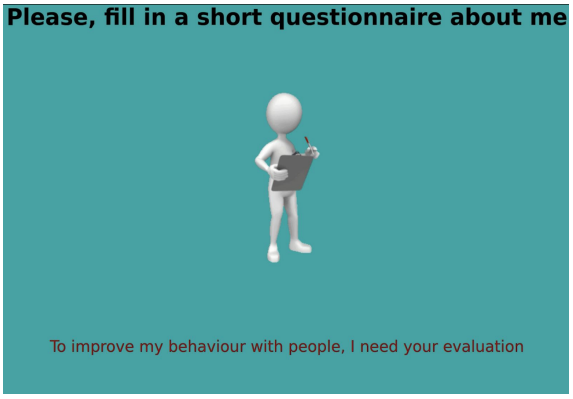


Figure 6

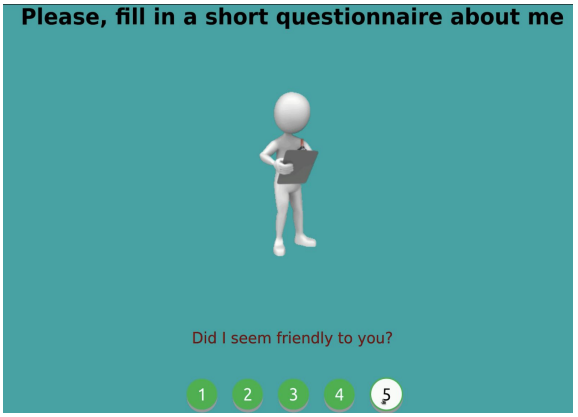


Figure 7

5.2 Supervisor

If Pepper meets someone in the corridors during the night, it is able to stop him immediately. Front sonar and rear sonar of the robot are always used to detect humans nearby; moreover the ASR service is implemented and always kept active with the keyword "*help*": anytime patients need something they just scream help and the robot will arrive (first part of the video **SickTooMuch**). So the robot raises its arm and stop the patient and the interaction starts.
Each time it is called, Pepper asks what is the problem and on its touchscreen three standard options appear (buttons) and a fourth too (Fig. 8) which stands for "*this is beyond my capability*" and it immediately call a nurse (video **Other**).
Note: to answer, the human can also speak and the robot will understand the problem anyway.

Figure 8

If we digit "*I am sick*", Pepper will worry about us asking "*How bad you feel?*": so we are faced with three paths, according to which the robot will behave differently. If we type "Too Much" (second part of the video **SickTooMuch**) (min. 0:35) means that there is a big problem, so Pepper will call immediately Doctors. Moreover it comforts us, saying to keep calm and that he will wait for the doctor with us. When we type "*A little*" (video **SickLittle**) instead Pepper understands that the problem is not big, so it will give us some advice (like take a moment...) or to hold on and wait until the next day.
Every interaction will conclude with the robot expression "*Now return in your room and try to rest*", so Pepper embodies its supervisory role very well.

Everytime the robot is activated (the script is successfully launched) two emergency buttons appear: *'nurse'* and *'doc'* buttons that, if pressed, they will immediately notify the medical staff. These become very important in case the patient is unable to interact with the robot and needs immediate help. Note that their working is always active, but is not shown in any video for time reasons.

6. Conclusion and future works

Robotic technology is experiencing huge advancements in these years providing efficient ways of helping people in many fields, like industries, biomedical, education, healthcare and others, allowing also to carry out tasks that are too cumbersome or dangerous for humans.
The scope of the project was to build a software application that would allows our social robot Pepper to interact with people in an medical environment. Software components made available by our *Professor Luca Iocchi (DIAG, University "La Sapienza", Rome)* and by SoftBank Robotics, have enabled us to achieve this result, previously shown. This experience has been one of the most practical and interesting one we have been so far, considering this is the first time ever we've been working on a real robot (actually, due to COVID-19 pandemic, we still could not have the chance to deal with the real robot).
Due to both time and practical reasons, some parts of the project have been not deeply developed (i.e: calling nurses or doctor, redirecting user to web pages and also further gestures considering we couldn't be able to use the emulator provided by SoftBank Robotics for some issues) and thus we manage to complete these ones and to add further features (i.e: SLAM algorithms for night walking, capability to answer to a wider range of questions, answer to call and book medical visits, make the application scalable) once we will get the possibility to come back to our department. Indeed, we are the opinion that working with the real robot would certainly be even more stimulating, allowing us to develop a smarter and more accurate one.
Being aware of the difficulties and the challenges that affect interactions between humans and robots (concerning mainly ethic and usability), we have designed our application taking into account all these problems and considering that robots must be included in our society with the sole purpose of supporting people, not replacing them, otherwise we will end up in one of those alienated societies that we see in science fiction films.

References

[1] "*Increasing trust in human–robot medical interactions: effects of transparency and adaptability*", Fischer K, Weigelin H M, Bodenhagen L; University of Southern Denmark, 2018

[2] "*State Of The Art: A Study of Human-Robot Interaction in Healthcare*", Olaronke I, Oluwaseun O, Roda I; Department of Computer Science, Adeyemi College of Education & Federal University of Technology, Nigeria, 2017

[3] "*Towards Robotic Systems for danger recognition and alert*", YouTube video, link at: https://www.youtube.com/watch?v=GgLX67rOrDM; Konika Minolta, 2019

[4] "*Discrepancies in a virtual learning environment: Something "Worth communicating about" for young children with ASC?",* Alcorn A, Good J; University of Edinburgh School of Informatics, University of Sussex Department of Informatics, 2013

[5] "*Emotional Robot for Intelligent System, Artificial Emotional Creature Project*", Shibata T, Inoue K, Irie R; AI Laboratory, Massachusetts Institute of Technology

[6] "*Loving AI: Humanoid Robots as Agents of Human Consciousness Expansion*", Goertzel B, Mossbridge G, Monroe E, Hanson D, Yu G; Hanson Robotics & OpenCog Foundation, Institute of Noetic Sciences (IONS) & Mossbridge Institute, LLC, 2017

[7] *"Growing Growth Mindset with a Social Robot Peer.",* Park, Hae Won & Rosenberg-Kima, Rinat & Rosenberg, Maor & Gordon, Goren & Breazeal, Cynthia. (2017).

[8] *"Facial Expression Recognition for Human-Robot Interaction – A Prototype",* Wimmer M, MacDonald B A, Jayamuni D, Yadav A (2008) In: Sommer G, Klette R (eds) Robot Vision. RobVis 2008. Lecture Notes in Computer Science, vol 4931. Springer, Berlin, Heidelberg

[9] "*Questionnaires to Measure Acceptability of Social Robots: A Critical Review*", Krägeloh U C , Bharatharaj J , Kutty S; Auckland University of Technology, 2019

[10] SOFTBANK ROBOTICS DOCUMENTATION, NAOqi OS getting started and Pepper Documentation, at this link: http://doc.aldebaran.com/2-5/home_pepper.html

[11] Care-O-bot 4 - Making-of the new service robot generation; YouTube video, link at: https://www.youtube.com/watch?v=OwAF8hsO58o&t=293s

[12] UBTECH Walker: Intelligent Humanoid Service Robot; Demo CES 2019, YouTube video, link at: https://www.youtube.com/watch?v=lWdcyWwQ4R0

[13] Moxi the Robot - Texas Health Resources, YouTube video, link at: https://www.youtube.com/watch?v=MVC4YAT2dNs