CAIR Software Handbook

ENGLISH

USER GUIDE - VERSION 1.0

Contributors: Lucrezia Grassi, Roberto Menicatti University of Genoa
Email: `lucrezia.grassi@edu.unige.it`

The system described in this document has been re-designed from scratch, by keeping some of the ideas of the original CARESSES system but completely re-writing the code at the University of Genoa. CARESSES[a] is an international, multidisciplinary project whose goal is to design the first robots that can assist older people and adapt to the culture of the individual they are taking care of. The robots can help the users in many ways including reminding them to take their medication, encouraging them to keep active, helping them keep in touch with family and friends. Each action is performed with attention to the older person's customs, cultural practices and individual preferences. CARESSES is funded by the European Commission and the Ministry of Internal Affairs and Communications of Japan. The project started on the 1st January 2017 and ended on the 31st January 2020.

[a]www.caressesrobot.org

Contents

Chapter 1

Download and Installation

This chapter explains how to download and set up the CAIR client.

As first thing, pull from the ***develop*** branch of the **CAIRclient** GitHub repository[1] the whole CAIR software.

The repository contains several applications. To install them on the robot, perform the following steps for each application:

- Download **Choregraphe** version 2.5.10 from this link[2] by clicking on the *Downloads* option in the menu and selecting your OS;
- Open the **pml** file inside the application folder you want to install with Choregraphe;
- Connect Choregraphe to the robot;
- Open the *Robot applications* window and click on the first button "*Package and install current project to the robot*".

Additionally, install the *follow_me* app from the Store (currently available only for Nao), and uploading it to the robot through Choregraphe. If the app *Force perception* is installed on your robot, uninstall it in order to avoid undesired behaviours in the head pose.
N.B. - Make sure to have at least **NAOqi 2.5** installed on the robot. The software may also work with older NAOqi versions, but it has only been tested with NAOqi 2.5 and 2.8.

Once all the applications are installed on the robot, to start using the CAIR system follow the procedure described in Section 3.1.

[1]https://github.com/lucregrassi/CAIRclient/tree/develop
[2]https://developer.softbankrobotics.com/pepper-naoqi-25

Chapter 2

CAIR System

This chapter provides basic information about the interaction with the robot connected to the CAIR Cloud.

2.1 Interaction Modes

The interaction between the user and the robot connected to the CAIR Cloud can be schematized as in Figure 2.1, where the circles represent the three *modes* or *states* that the robot can assume and the rectangles represent the situations which make the interaction switch from one mode to the other.

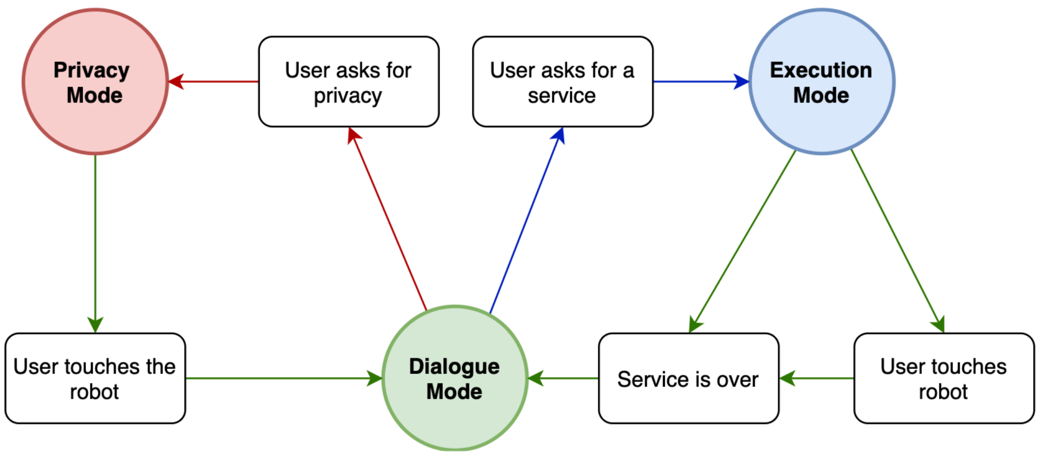

Figure 2.1: State machine representing the three states in which the robot connected to the CAIR Cloud can be during the interaction with the user.

2.1.1 Dialogue Mode

In the ***Dialogue*** Mode (chit-chatting) the robot verbally interacts with the user and tries to detect *triggering keywords* or specific *patterns* in what the user says. The *triggering keywords* lead the conversation to a specific topic, while the *patterns* activate the execution of a specific service/action (*Execution* Mode). If the user explicitly asks for it, the robot can switch to the

Privacy Mode.
The robot's tablet displays Figure 2.2(a) on the screen.

2.1.2 Execution Mode

In the ***Execution*** Mode the robot interacts with the user by executing the requested service. When the service is over, the robot exits this state and switches to the *Dialogue* Mode. The user can force the robot to exit the ***Execution*** Mode by touching one of its sensors or using one of the *Exit Execution-Mode keywords* (see Section 2.3).
N.B. - During the ***Execution*** Mode it is not possible to start other services or a conversation, unless by first quitting the current task.
The robot's tablet displays Figure 2.2(b) on the screen.

2.1.3 Privacy Mode

In the ***Privacy*** Mode the robot turns away and covers its eyes to give privacy to the user. The robot does not listen nor talk with the user until the user touches the robot.
The robot's tablet displays Figure 2.2(c) on the screen.
N.B. - From a technical point of view, the Privacy Mode is a task implemented in the same way of all other Execution Mode tasks, however, it does not terminate unless the user touches the robot.

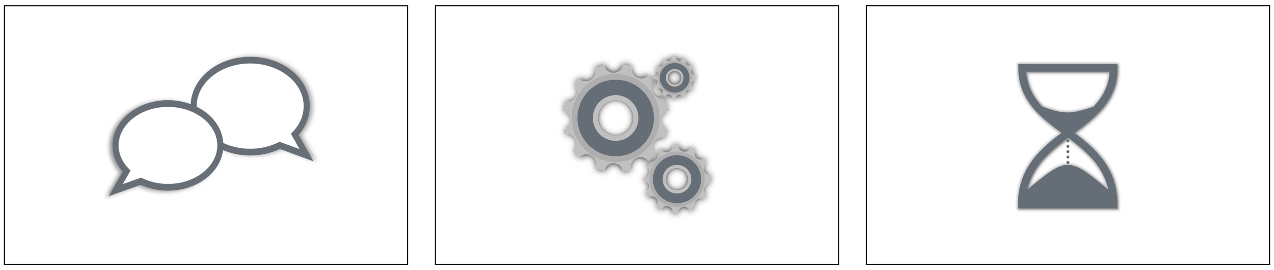
(a) Dialogue Mode screen (b) Execution Mode screen (c) Privacy Mode screen

Figure 2.2: Different tablet screens for the four modes.

2.2 Inputs to the CAIR robot

You can give **five** kinds of input to the CAIR robot. It is **very** important to know ***why***, ***when***, and ***how*** to give these inputs.

2.2.1 Chat Request Input

- **Why:** in order to start chatting about a topic.
- **When: when the user wants to change the conversation topic.**
  Notice that this kind of input can be given **only** when the robot is **in Dialogue Mode.**

- **How:** simply mention the topic you want to talk about. Two example sentences to start talking about *sport* could be: "*Let's talk about **sport!***" or "*You know? I really like **sport***".

2.2.2 Task Request Input

- **Why:** in order to make the robot execute a task.
- **When: when the user wants the robot to execute a task**.
  Notice that the request for executing a task can be given **only** when the robot is **in Dialogue Mode**.
- **How:** use one of the **triggering sentences** mentioned in Section 3.2. Please notice that you don't necessarily have to give all the details of your request at this stage, the robot will ask for any missing information later if necessary (see Section 2.2.5). For example, you don't have to specify immediately the title of the song that you want to listen to.

2.2.3 Open Answer Input

- **Why:** in order to say anything you want concerning the current topic of conversation.
- **When: when invited by the robot** to express your idea on something, for example when the robot says:
  - "*Please, **tell me** something **about** your favourite traditional Indian outfit if you like*".
  - "*Please, **tell me** more **about** Italy. I would like to know about its natural beauties*".

  Notice that **these are not questions**.
- **How:** just talk freely, e.g.: "*My favourite outfit is the kurta because it reminds me of... etc.*".

2.2.4 Yes/No Input

- **Why:** in order to answer questions about your habits and preferences or to trigger the execution of a goal.
- **When: when the robot makes a yes/no question**, such as:
  - "***Do you** live in Genova?*".
  - "***Do you** make flower arrangement sometimes?*".
  - "***Do you** want me to play the song You Really Got Me for you?*".

  Notice that **these are yes/no questions**.
- **How:** reply clearly by saying "*yes*", "*no*" or a similar word conveying a positive or negative meaning such as "*always, never...*". It is better to add some words to facilitate speech-recognition for the robot: e.g., "*yes, sure*", "*no, never*" or "*yes, please*", "*no, thanks*".

2.2.5 Parameter Input

- **Why:** in order to give more details to the robot about the task you want to execute.
- **When: when the robot asks for a specific parameter**, such as:
  - "*Of which **city** do you want to know the weather forecast?*".
  - "*Please tell me the **title** of the song you want to listen.*"
  - "*For which **part of the body** do you want me to show you exercises?*".

  Notice that **the request is specific**.
- **How:** give a specific answer. The more specific is your answer, the sooner the robot will accomplish the task. For example, the answers to the questions above may be: "*Genova*", "*Hallelujah*", and "*Arms*".

N.B. - Be sure that the CAIR robot has finished talking before giving any kind of input. **While the robot is talking, it does not listen to you and cannot be interrupted.**

2.3 Ways to ease the interaction

While interacting with the robot, you can use some *special keywords* and *special sentences* (even within a more structured sentence). **Please notice that the keywords/sentences work only when the robot is waiting for an input from the user and not when it is *doing* something** (e.g., while the robot is talking, moving, elaborating a request, etc.).

- ***Repeat keywords***: repeat, say it again.
  Use these keywords to make the robot repeat its last sentence.

- ***Initiate Execution-Mode sentences***: sentences depend on the task you want to activate, (see Section 3.2).
  Use these sentences to start the execution of a task.

- ***Exit Execution-Mode keywords***: stop, exit, quit.
  Use these keywords to exit the Execution Mode when the robot asks for a missing parameter during the execution of a task (see Section 3.2).

- ***Exit CAIR keywords***: stop talking, quit the application.
  Use these keywords to end the dialogue and exit the Dialogue Mode and the CAIR app.

Chapter 3

Interacting with the System

This chapter is meant to be a guide for the user who wants to interact with the CAIR robot without knowing what is going on "beneath the surface".

3.1 Chit-Chatting

The application that manages the interaction with the CAIR Cloud is the **DialogueManager**.

This application contains the *chit-chatting* behavior that can be launched either from Choregraphe or by saying one of the following trigger sentences:

- Let's talk.
- Talk with me.

To launch the behavior through its trigger sentences, the robot should be subscribed to the **Basic Channel**[1]. Once subscribed, make sure that the robot is listening when saying a trigger sentence (the eyes should be blue and it should emit a "bip" sound. The execution of the behavior is acknowledged by the sentence "*I am connected to the CAIR Cloud!*".

N.B. - Each time the *chit-chatting* behavior is started, to avoid conflicts, it changes the robots settings by disabling the *Autonomous Life* before beginning the interaction. The user will be informed of this operation through the following sentence: "*Give me a moment. I need to change some of my settings before we start.*"

The first time the user runs this behavior, the Cloud will send the initial Client State to the robot, which will be stored in a local file and will be used for all the future requests performed by the same client. This practice is shown in Figure 3.1. To delete all data and start a new conversation, the user can remove and re-install the Dialogue Manager application (through Choregraphe) or simply delete the file in which the client state is stored; in this way, a new initial Client State will be assigned at the following interaction with the server.

During the execution of the chit-chatting behavior, you can ask the robot either to:

[1]http://doc.aldebaran.com/2-5/nao/nao_store_channel.html#nao-store-channel

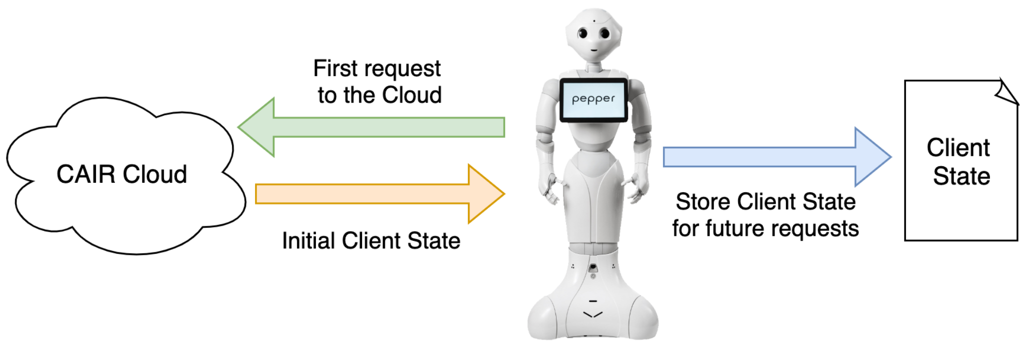

Figure 3.1: First interaction with the CAIR Cloud.

- **chat about a topic** or to
- **execute a specific task** among the ones installed (see following sections).

3.1.1 Conversation Topics

This is a non-exhaustive list of the **topics you can chat about** with the CAIR robot, just to give you some suggestions.

- Events
- Family
- Friends
- Food
- Habits
- Health
- Hobbies (books, music, movies, sport...)
- Religion

3.2 Execution of Tasks

As already mentioned in Section 2.1, during the *Dialogue* Mode (chit-chatting), the robot can detect specific *patterns* in the user sentences that can trigger the execution of a specific task (*Execution* Mode).

The execution of *most* tasks can be **stopped** in one of the following ways:

- Saying one of the **Exit Execution-Mode keywords**: *stop, exit, quit*. This works only when the robot is listening (e.g. when it asks for a missing parameter during Execution Mode;
- Touching the robot either on its head, on the back of its hands or on the bumpers (or feet in the case of Nao), when explicitly told by the robot during a task (see Figure 3.2).

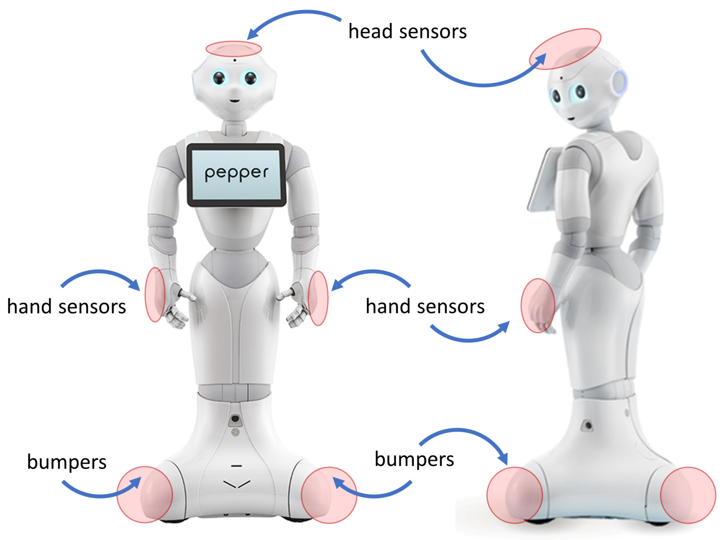

Figure 3.2: Pepper tactile sensors

The following subsections describe all the tasks that can be triggered during the conversation, along with their triggering *patterns*, and the exit modes.

N.B. - All the behaviors that ask for missing parameters can also be launched individually from Choregraphe.

3.2.1 Utility

The **Utility** application makes the robot able to perform two basic operations: set the volume of the robot and set the voice speed.

This application contains two behaviors: *volume* and *voice_speed*.

3.2.1.1 Volume

The *volume* behavior allows the user to set the volume of the robot.

Parameter: the **level** of the volume.
N.B. - The specified level of the volume can be absolute (i.e., high, normal, low) or relative (i.e., louder, lower/quieter).

Trigger Sentences:

- Talk with **high** speed;
- Talk **louder**.

The absolute levels correspond to precise values of the volume (i.e., 100%, 60%, 40%), while the relative ones increase or decrease the volume by 10. The behavior terminates after the volume has been set.

3.2.1.2 Voice Speed

The *voice_speed* behavior allows the user to set the speed with which the robot speaks.

Parameter: the **level** of the voice speed.
N.B. - The specified level of the voice speed can be absolute (i.e., high, normal, low) or relative (i.e., faster, slower).

Trigger Sentences:

- Talk with **high** speed;
- Talk **faster**.

The absolute levels correspond to precise values of the voice speed (i.e., 100%, 70%, 60%), while the relative ones increase or decrease the voice speed by 10. The behavior terminates after the voice speed has been set.

3.2.2 Greetings

The **Greetings** application makes the robot reply to the greetings that are popular in three cultures: English, Indian, and Japanese[2].
This application contains three behaviors: *hello*, *namaste*, *konnichiwa*.

3.2.2.1 Hello

The *hello* behavior makes the robot answer to the English greeting.

Parameter: no parameter.

Trigger Sentences:

- Hello;
- Hi Pepper.

The robot will answer by waving its hand while saying "*Hello*", and the behavior will terminate as soon as the move is finished.

3.2.2.2 Namaste

The *namaste* behavior makes the robot answer to the Indian greeting.

Parameter: no parameter.

Trigger Sentences:

[2]The cultures have been arbitrarily chosen during the H2020 CAIR project (`caressesrobot.org`) as a case study to evaluate cultural competence in robots.

- Namaste;
- Namaste Pepper.

The robot will answer by joining both the palms together in a worshipful pose while saying "*Namaste*". The behavior will terminate as soon as the move is finished.

3.2.2.3 Konnichiwa

The *konnichiwa* behavior makes the robot answer to the Japanese greeting.

Parameter: no parameter.

Trigger Sentences:

- Konnichiwa;
- Konnichiwa Pepper.

The robot will answer by performing a small bow while saying "*Konnichiwa*". The behavior will terminate as soon as the move is finished.

3.2.3 Time Tools

The **TimeTools** application allows the robot to tell the current date and time to the user.
This application contains two behaviors: *date* and *time*.

3.2.3.1 Date

The *date* behavior provides today's date to the user.

Parameter: no parameter.

Trigger Sentences:

- "What day is *today*?"
- "What is today's *date*?"

The behavior terminates as soon as it tells today's date to the user.

3.2.3.2 Time

The *time* behavior provides the current time to the user.

Parameter: no parameter.

Trigger Sentences:

- "What *time* is it?"
- "What's the *time*?"
- "Tell me the *time*."

The behavior terminates as soon as it tells the current time to the user.

3.2.4 Weather Forecast

The **WeatherForecast** application allows the user to get the weather forecast for a specified city.

This application contains only the *weather* behavior.

3.2.4.1 Weather

The *weather* behavior performs all the operations needed to provide the weather forecast to the user.

Parameter: the **city** of which the user wants to know the weather forecast.

Trigger Sentences:

- "What's the weather like in **Genova**?"
- "How's the weather forecast for **Genova**?"
- "How is the weather like?"

If the trigger sentence does not contain a city, the procedure for the recovery of the missing parameter is started: in this case, the user will be asked: "*Of which **city** do you want to know the weather forecast?*". The user can either answer with the name of the city or with one of the aforementioned *Exit Execution-Mode keywords* to terminate the execution of the task.

If a city is provided as input, the task will terminate after telling the current temperature in degrees Celsius, the description of the weather, the minimum and the maximum temperature during the day (if available).

3.2.5 Word Tools

The **WordTools** application allows the user to perform three operations: translate words/sentences from English to other languages, search for the definition of words in the dictionary, search for things on Wikipedia.

This application contains three behaviors: *translator*, *dictionary*, and *wikisearch*.

3.2.5.1 Translator

The *translator* behavior translates words or sentences from English to a language among those installed on the Robot.

Parameter: the **text to translate** and the **destination language** of the translation.

Trigger Sentences:

- "How do you say **dog** in **Italian**?"
- "Translate into **Italian**."
- "Translate **the cat is on the table**."
- "Open the *translator*."

This behavior expects two parameters, however, it is possible that only one or no parameters are found in the trigger sentences. Once the behavior is started, if the text to be translated is missing, the user will be asked "*What would you like me to translate?*"; if the destination language is missing the user is asked "*In which language do you want me to translate?*". In both cases, the user can answer with the required parameter, or with an *Exit Execution Mode keyword* to terminate the behavior.

When both parameters are recovered, the Robot will provide the translation by saying "*The **Italian** for **dog** is cane.*", and terminate.
N.B. - The destination language could either may not be installed, or it could be a not supported language not by *ALTextToSpeech*. These cases are both managed, and the error message given by the robot will be different.

3.2.5.2 Dictionary

The *dictionary* behavior provides the definition of a word from the dictionary.

Parameter: the **word** to be searched on the dictionary.

Trigger Sentences:

- "What's the definition for **car**?"
- "What does **car** mean?"
- "Define **car**."
- "Open the *dictionary*."

If the trigger sentence does not contain the expected parameter, the behavior will start the parameter retrieval procedure by asking the user "*What word would you like me to search on the dictionary?*". As in the previous cases, while the robot is listening, the user can also say one of the *Exit Execution Mode keywords* to terminate the task.

After the parameter has been provided, the behavior communicates the result of the research to the user by saying: "*The definition of* ***car*** *is a motor vehicle with four wheels usually propelled by an internal combustion engine*", and then it terminates.

3.2.5.3 Wikisearch

The *wikisearch* behavior allows the user to search for something on Wikipedia.

Parameter: the **thing** to be searched on Wikipedia.

Trigger Sentences:

- "Search for **Harry Potter** on Wikipedia."
- "Look up **Harry Potter** on Wikipedia."
- "Open *Wikipedia*."

When the trigger sentence does not contain the thing to be searched on Wikipedia, the parameter retrieval procedure is started and the user is asked "*What would you like to search on Wikipedia?*". As always during a parameter request, the user has the possibility of terminating the task by saying one of the *Exit Execution Mode keywords* or provide the requested parameter.

The provided parameter is used to perform a search on Wikipedia and the first sentence is provided as a description: "*Here's what I found on Wikipedia about Harry Potter: Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling*". The parameter may not be found on Wikipedia; in this case, the robot will inform the user with an error message and the behavior terminates.

3.2.6 Music Player

The **MusicPlayer_Pepper** application allows the user to listen to the requested song, with the possibility of watching also the video on YouTube.

This application contains three behaviors: *play-song*, *play-music*, and *play-video*.

The **MusicPlayer_Nao** application contains only the *play-song* and the *play-music* behavior, as Nao is not equipped with a tablet and cannot give to the user the option of watching the video.

3.2.6.1 Play song

The *play_song* behavior is in charge of retrieving the eventual missing parameter and, based on the user intention to watch or not the video on YouTube, calling the appropriate behavior.

Parameter: the **title** of the song that the user wants to listen.

Trigger Sentences:

- "Play the song **Hey Jude**."
- "I want to listen to **Hey Jude**."
- "Play *some music*."

If no song title or artist is specified within the request, the user will be asked "*Please, tell me the **title** of the song you want to listen to*". Again, the user can either answer by providing the title of the song, or with an *Exit-Execution Mode keyword* to terminate the task.

If some parameter is provided, the task will proceed by asking the user: "*Do you want me to show you also the video?*"; if the user answers affirmatively, the *play-video* behavior is started, otherwise, the *play-music* behavior is started.

This behavior will end as soon as the started behavior (*play-video* or *play-music*) is terminated.

3.2.6.2 Play video

The *play_video* behavior can only be called by the *play-song*, to make sure that the title of the song to be searched on YouTube is available. As soon as the behavior is started, it will inform the user by saying "*Give me a moment. I'm looking for the video on YouTube*". The search query on YouTube is performed by adding the word "*lyrics*" after the provided song title/artist, to make sure that the video contains only the music. The first video of the search result is shown on the Robot's tablet. Once the video is loaded, the Robot will inform the user by saying: "*Press play when you're ready. Touch my head, my hands or my bumpers when you want to stop*".
N.B. - The first time that YouTube opens, it may ask you to accept their policies about the use of cookies and data: scroll to the bottom of the page and click on "*I agree*". When the video is over, the user can continue to watch other videos on YouTube while in Execution Mode. To return to Dialogue Mode touch one of the Robot's sensors.

3.2.6.3 Play music

The *play_music* behavior can only be called by the *play-song*, to make sure that the title of the song to be played is available. As soon as the behavior is started, it will inform the user by saying "*Give me a moment. I'm looking for the song. Please, be patient, it may take a while*". Once the song is loaded, the Robot will inform the user by saying "*Touch my head, my hands or my bumpers when you want to stop*", and the playing of the song will begin automatically. When the song is over, the robot will terminate the task, exit the Execution Mode and switch back to Dialogue Mode.

3.2.7 Karaoke Player

The **KaraokePlayer_Pepper** application allows the user to play the karaoke of the requested song on YouTube.

This application contains only the *karaoke* behavior.
N.B. - This application is not available for the Nao robot since the tablet is required.

3.2.7.1 Play Karaoke

The *play-karaoke* behavior is in charge of performing all operations needed to play the karaoke video from YouTube of the song requested by the user.

Parameter: the **title** of the song the user wants to sing.

Trigger Sentences:

- "Play the karaoke for the song **Billie Jean**"
- "I want to sing **Billie Jean**."
- "Open the *karaoke*."

As in the previous case, if no song title is passed as a parameter in the trigger sentence, the user will be asked "*Please, tell me the title of the song you want to sing*". The user can either answer by providing the title of the song or with an *Exit-Execution Mode keyword* to terminate the task.

If the title of the song is provided, the behavior continues by searching a query on YouTube, adding the word "*karaoke*" after the title requested by the user. The first video of the search result is shown on the Robot's tablet. Once the video is loaded, the Robot will inform the user by saying: "*Press play when you're ready to sing. Touch my head, my hands or my bumpers when you want to stop*". Similarly to the *play_video* behavior, when the karaoke video is over, the user can continue to watch other videos on YouTube while in Execution Mode. To return to Dialogue Mode it is necessary to touch one of the robot's sensors.

3.2.8 Movie Player

The **MoviePlayer_Pepper** application allows the user to play the trailer of the requested movie on YouTube.

This application contains only the *play-movie* behavior.

N.B. - This application is not available for the Nao robot since the tablet is required.

3.2.8.1 Play Movie

The *play_movie* behavior is in charge of performing all operations needed to play the trailer of the movie requested by the user on YouTube.

Parameter: the **title** of the movie of which the user wants to see the trailer.

Trigger Sentences:

- "Show me the trailer of "**Titanic**".
- "Play the movie "**Titanic**"."
- "Show me a *trailer*."

As always, if no movie title is passed as a parameter in the trigger sentence, the user will be asked "*Please, tell me the title of the movie*". The user can either answer by providing the title of the movie, or with an *Exit-Execution Mode keyword* to terminate the task.

If the title of the movie is provided, the behavior continues by searching a query on YouTube, adding the keywords "*official trailer*" after the title requested by the user. The first video of the search result is shown on the Robot's tablet. Once the video is loaded, the Robot will inform the user by saying: "*Press play when you're ready to watch the trailer.* "*Touch my head, my hands or my bumpers when you want to stop*". Similarly to the *play_karaoke* behavior, when the trailer is over, the user can continue to watch other videos on YouTube while in Execution Mode. To return to Dialogue Mode it is necessary to touch one of the robot's sensors.

3.2.9 Video Instructions

The **VideoInstructions_Pepper** application allows the user to play the instructions of the requested activity on YouTube.

This application contains only the *show-instructions* behavior.

N.B. - This application is not available for the Nao robot since the tablet is required.

3.2.9.1 Show Instructions

The *show_instructions* behavior is in charge of performing all operations needed to show the instructions of the activity requested by the user on YouTube.

Parameter: the **activity** of which the user wants to see the instructions.

Trigger Sentences:

- "Show me the instructions for **playing the guitar**".
- "I want to know how to **play the guitar**."
- "I need some *instructions*."

As in the previous cases, if no activity is passed as a parameter in the trigger sentence, the user will be asked "*For which activity do you want me to show you the instructions?*". The user can either answer by providing the name of the activity, or with an *Exit-Execution Mode keyword* to terminate the task.

If the name of the activity is provided, the behavior continues by searching a query on YouTube, adding the keyword "*tutorial*" after the activity requested by the user. The first video of the search result is shown on the Robot's tablet. Once the video is loaded, the Robot will inform the user by saying: "*Press play when you're ready to watch the instructions. Touch one of my sensors when you want to stop*". Similarly to the previous behaviors involving playing a video on YouTube, when the tutorial for the activity is over, the user can continue to watch other videos on YouTube while in Execution Mode. To return to Dialogue Mode it is necessary to touch one of the robot's sensors.

3.2.10 Video Exercises

The **VideoExercises_Pepper** application allows the user to play a workout video on YouTube targeting the part of the body requested by the user.

This application contains only the *play-exercise* behavior.

N.B. - This application is not available for the Nao robot since the tablet is required.

3.2.10.1 Play Exercise

The *play_exercise* behavior is in charge of performing all operations needed to play a video on YouTube showing how to perform the exercises for the muscle group requested by the user.

Parameter: the **part of the body** for which the user wants to see an exercise video.

Trigger Sentences:

- "I want to train my **arms**".
- "Show me an **arms** workout."
- "I want to do some *exercise*."

As in the previous cases, if no part of the body is passed as a parameter in the trigger sentence, the user will be asked "*For which part of the body do you want me to show you some exercise?*". The user can either answer by providing the target body area, or with an *Exit-Execution Mode keyword* to terminate the task.

If the name of the targeted area is provided, the behavior continues by searching a query on YouTube, adding the keywords "*home exercise*" after the activity requested by the user. The first video of the search result is shown on the Robot's tablet. Once the video is loaded, the Robot will inform the user by saying: "*Press play when you're ready to exercise. Touch one of my sensors when you want to stop*". Similarly to the previous behaviors involving playing a video on YouTube, when the workout video is over, the user can continue to watch other videos on YouTube while in Execution Mode. To return to Dialogue Mode it is necessary to touch one of the robot's sensors.

3.2.11 Movement

The **Movement** application allows the user to perform the following operations:

- Move slightly forward, back, left or right;
- Move forward, back, left or right until a sensor is touched or an obstacle is encountered;
- Learn the location of a place in the environment;
- Set the position of the robot to a known place in the environment;
- Make the robot go to a known location in the environment;

- Make the robot go to charge (rest) on its docking station;
- Make the robot leave its docking station (wake up);
- Forget the map of the environment (and the transformation matrix).

This application contains six behaviors: *move*, *go*, *learn_place*, *set_position*, *go_to*, and *forget_map*.

Thanks to the behaviors contained in this application, the user can create a map of the environment, by teaching the robot where the places are (see Section 3.2.11.3); this map can then be used to make the robot move to the desired location (see Section 3.2.11.5).

3.2.11.1 Move

The *move* behavior allows the robot to perform a small movement (0.2m) in the required direction (forward, back, right, left).

Parameter: the **direction** in which the robot should perform the movement.

Trigger Sentences:

- "Move **left**."
- "Step to the **left**."
- "Step **left**."

In this case, the task cannot be executed without a parameter: if no direction is provided, the robot will ignore the request. When a direction is provided, the robot will confirm the intention of executing the task by saying "*Ok, I will move **left***". When the move is finished, the task terminates.

3.2.11.2 Go

The *go* behavior makes the robot move towards a specified direction (forward, back, right, left).

Parameter: the **direction** in which the robot should go.

Trigger Sentences:

- "Go **right**."
- "Come **forward**."

Also in this case, the task cannot be executed without a parameter: if no direction is provided, the robot will ignore the request. When a direction is provided, the robot will say: "*OK, I will go **right**. Touch my head, my hands or my bumpers when you want me to stop.*", and it will start moving in the requested direction. The behavior terminates either when the user touches the robot, or when an obstacle is encountered.

3.2.11.3 Learn Place

The *learn_place* behavior allows the robot to learn the location of a place in the environment.

Parameter: the **name** of the place that the robot should learn.

Trigger Sentences:

- "This place is the **bedroom**."
- "This place is my **bedroom**."
- "Learn this *place*."

If the name of the place is not passed as a parameter, the *parameter retrieval procedure* is started and the user is asked: "*Which place is this?*"; as always, the user can decide to terminate the task or provide the name of the place. In case a name is provided, the robot will acknowledge that it has understood and it will add the name of the place and its coordinates to the map.
N.B. - The coordinates are always computed with respect to the environment; for this reason, if the map is not empty, the robot should know where it is in the environment before a new place could be added to the map. If the map is empty, the coordinates of the new place are stored with respect to the robot frame and the transformation matrix will be an identical matrix.

3.2.11.4 Set Position

The *set_position* behavior allows the robot to understand where it is with respect to the environment.

Parameter: the **name** of the place in which the robot is.

Trigger Sentences:

- "You are in the **kitchen**."
- "You are in my **kitchen**."

This behavior is started only if the name of the place is provided as a parameter. The place should be present in the map of the environment, otherwise, the robot will not able to compute the transformation matrix between its frame and the world frame and it will inform the user by saying: "*I'm sorry, I don't know where the **kitchen** is*". If the place is found in the map, the robot will compute the transformation matrix, and before terminating the behavior it will acknowledge the success of the operation by saying: "*My position is now set to **kitchen***".

3.2.11.5 Go To

The *go_to* behavior allows the robot to go to a specified location (some place in the room, among those of the map).

Parameter: the **name** of the place in which the robot should go.

Trigger Sentences:

- "Go to the **bathroom**."
- "Go *somewhere*."

In case the trigger sentence does not contain the name of the place in which the robot should go, the *parameter retrieval procedure* is started and the robot is asked: "*Where should I go?*". If the user does not exit the task by saying one of the *Exit Execution Mode keywords*, the behavior checks if the provided place is present on the map. If not, the robot will say: "*I'm sorry, I don't know where the **bathroom** is*". Otherwise, if the place is in the map, but the robot does not know its position with respect to the environment (because it has not been previously set with the *set_position* behavior), it will say: "*I need to know where I am with respect to the environment before I can go somewhere*".

If the place exists in the map and its position with respect to the environment is known (i.e., the transformation matrix is stored in the memory), the robot will say "*I'm going to the bathroom. Touch one of my sensors if you want me to stop.*", and it will start moving.

The behavior can terminate for one of the following reasons:

- the destination is reached;
- the robot encounters an obstacle;
- the user touches the robot.

3.2.11.6 Rest

The *rest* behavior makes the robot dock on its docking station.

The *go_to* behavior with parameter "docking station" is always called before the *rest* behavior, to make sure that the robot is near the docking station (possibly looking towards it) to increase the probability of finding it. This assumes that the location of the "docking station" is present in the map; if not, the robot will tell the user that it doesn't know where that place is.

Parameter: no parameter.

Trigger Sentences:

- "Go to the *docking station*."
- "Go to *charge*."

- "Go to *rest*."

Known Issues: sometimes the robot is not able to find the docking station. Pepper uses the bottom camera to localize the docking station: apparently, when there is too much light, the robot is not able to localize the station as the color of the blue LEDs is not clearly visible.

This behavior makes the robot start looking for the docking station, move in front of it, and dock. If one of these operations fails, the robot will inform the user that something went wrong; otherwise, it will acknowledge the success of the operation by eventually saying: "*I'm successfully docked!*", and the behavior terminates.
N.B. - It is not possible to interrupt the robot while it is trying to dock, as the operation is blocking.

3.2.11.7 Wake Up

The *wake_up* behavior allows the robot to leave the docking station.

Parameter: no parameter.

Trigger Sentences:

- "Wake *up*."
- "Stop *charging*."
- "Leave the *docking station*."

This behavior terminates as soon as the robot has left its docking station.

3.2.11.8 Forget Map

The *forget_map* behavior allows the robot to empty the map of the environment.

Parameter: no parameter.

Trigger Sentences:

- "Forget the *map*."
- "Reset the *map*."
- "Delete the *map*."

This behavior terminates as soon as the *map.txt* file, in which the known locations of the environment are stored, is empty, and the transformation matrix (if present) is deleted from the memory. The robot will acknowledge this operation by saying "*The map is now empty*".

3.2.12 Provide Privacy

The **ProvidePrivacy** application makes the robot look the other way to provide privacy to the user.

This application contains only the *privacy* behavior.

3.2.12.1 Privacy

The *privacy* behavior makes the robot perform all the operations needed to provide privacy to the user.

Parameter: no parameter.

Trigger Sentences:

- "Provide *privacy*."
- "Give me some *privacy*."
- "I need some *privacy*."

This behavior makes the robot turn the other way, look down, and cover its face with its hands. The behavior terminates when the robot returns to its previous position after the user touched it.
N.B. - Make sure that you authorized the **deactivation of the safety reflexes**, otherwise, the robot cannot turn after it has been touched. To do this, you must go to the *Pepper advanced webpage*[3] and check the checkbox to authorize it. If you don't know the IP of your robot, press the chest button once.

3.2.13 Follow Me

The **FollowMe** application makes the robot move with the user holding hands.

This application contains only the . behavior.

3.2.13.1 Follow

Parameter: no parameter.

Trigger Sentences:

- "Follow *me*."
- "Come with *me*."

[3]robot_ip/advanced/#/settings

As soon as this behavior is started, the robot will say: "*I want to come with you, if I'm too fast for you, you can stop me by touching my head*"; then it will hold out its hand, waiting for the user to take it. The robot will start walking in the direction in which the user pulls its hand, and the behavior will stop the execution when the user touches the robot's head.

N.B. - The trigger sentences of all tasks are recognized by a set of regular expressions associated with each task. Slight variations of such sentences can be recognized, and more sentences can be easily added in the future to provide more variety and the possibility to recognize the user intention to start a certain task.