

WS 2025/26

LAB COURSE:
HUMAN ROBOT INTERACTION

TASK 5: VERBAL INTERACTION

Lehrstuhl für Autonome Systeme und Mechatronik
Friedrich-Alexander-Universität Erlangen-Nürnberg

Prof. Dr.-Ing. habil. Philipp Beckerle
Martin Rohrmüller, M. Sc.

Introduction

In this task, the NAO robot as shown in Figure 1 is used. NAO is a 58 cm tall humanoid robot and is used in different areas of research and education. NAO has various sensors such as two cameras, microphones, speakers and touch sensors. Through the sensors, the robot can interact with the environment, understand people and communicate with them. More information about Nao can be found on the manufacturer's website (Aldebaran) ¹.

The possibilities of graphical programming are taught and thus a simple dialog interaction is built up. This is then extended by means of a dialog language in order to perform a more extensive interaction task. The preparation part focuses on the graphical programming interface, the execution part then on the implementation of the dialog on the real robot.



Figure 1: NAO robot

¹<https://www.aldebaran.com/en/nao6>

Task: Preparation

To prepare for the two tasks with the NAO robots the Choregraphe suite will be used. This is the development environment provided by the manufacturer.

First steps with Choregraphe

Choregraphe is a block-based programming environment, which facilitates an easy entry into the programming of the robot. More advanced programmers can also program the robot in Python or C++. For the introduction to Choregraphe there are videos² from the University of Marburg and the Federal Ministry of Education and Research.

Exercise 1.

Starting Choregraphe on the stick and connecting to the robot.

- a) Run the Choregraphe suite with the terminal command

```
1 $ cd naosoftware/choregraphe-suite-2.1.4.13-linux64  
2 $ ./choregraphe
```

on the Ubuntu stick.

- b) Watch and follow the video tutorials in the playlist to get to know how to use the interface. You can skip the video about the language choice, since the default English settings will be used.

The simulation environment is very suitable to take the first steps to learn about the possibilities of the humanoid robots. The first exercises are intended to help you work better with the real robot later on and you will notice that there are some big differences between reality and simulation.

Starting with a simulated robot

In the following description you will learn the first steps to operate the robot via Choregraphe. As shown in the video tutorials, you can start the virtual or simulated robot in the program environment. For the following, we encourage you to try other examples as well, which you can find in the documentation³. Note: Not all examples can be tested on the simulated robot.

Exercise 2.

Use the following boxes from the box library on the simulated robot. Feel free to try even more and to put together program sequences.

- a) 
- b) 

²<https://www.youtube.com/playlist?list=PL7Sf9IQyLv99iM40aDDSSKR93kx8jQA7Y>

³http://doc.aldebaran.com/2-8/software/choregraphe/choregraphe_first_steps.html

- c) **[Speech] > Creation > Choice**. Double click the box to edit it. The robot will query the defined text, for the simulated one this appears in the dialog window of Choregraphe. If this is not visible yet, tick "Dialog" in the "View" settings in the menu bar. Note that it can take a few seconds until the answer appears at the "answer" output of the box and until the question appears in the Dialog window.
- d) **[Movement] > Posture > Sit Down**
- e) **[Speech] > Creation > Speech Reco.**. This will throw an error. Why?

Add screenshots to the submission file.

In the Animated Say box an animation can be set to be executed during the speech output. The online documentation⁴ lists the possible commands. Those will be important for the execution part on the real robot, but not for the simulated one. This gives the speech output in a speech bubble however, does not display the animation at all. Some features such as the eye LEDs are executed on the simulated robot but not displayed as well. In addition to that, some things are not even executed on the simulated robot in the first place.

Creating a simple dialog graphically

The graphical interface and the presented functions can be used to compose a dialog. The blocks must be assembled accordingly. Another box needed for the next exercise is the "Choice" box which can give several outputs based on the user's input. If you need more information about how to set it up you can look at the online example⁵.

Exercise 3.

Create a dialog with the boxes. Follow the structure as shown in Figure 2. You can use other sentences than in the example and also extend it. Execute it on the simulated robot and use the dialog window for the inputs.

Add a screenshot including the dialog window to the submission file.

As you may have noticed, the graphical user interface is very simple and clear, but the functionality is quite limited and especially more complex dialogs are difficult to implement. To overcome this, a syntax based way to program dialogs can be used which will be shown in the next steps.

Dialog box in Choregraphe

In the graphical programming window of Choregraphe, dialog boxes can be created. Those boxes can then be combined with the other boxes and functions that the program provides. In the next sections, this dialog boxes will be expanded with code in order to

⁴http://doc.aldebaran.com/2-1/naoqi/audio/alanimatedspeech_advanced.html#animated-speech-list-behaviors-nao

⁵<http://web.eecs.utk.edu/~lepark/Courses/CS494-529-fall11/NAOs/interaction.html>

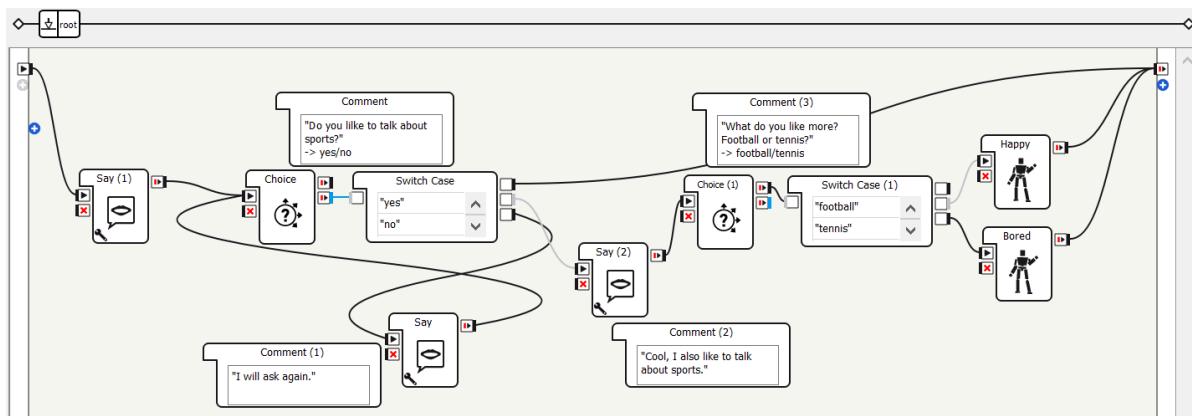


Figure 2: Example dialog implemented graphically.

implement a whole dialog for interaction with the robot. But for now to start, use the online tutorial⁶ to set up the boxes.

Exercise 4.

Follow the steps in this exercise to create the dialog box and refer to the online tutorial for all the necessary information.

- Open a new project and create a new dialog box. (You don't need the Set Language box)
- Create a new dialog topic (Note, that the notation "topic" here has nothing to do with ROS Topics).
- Edit the `..._enu.top` file and add the greeting example from the online tutorial.
- Run it on the virtual robot and try it with the dialog window.
- (Voluntary) Continue with the rest of the tutorial. However, the further steps will be covered in the following exercises.

Again, add a screenshot including the dialog window to the submission file

Dialog creation with QiChat

QiChat is a simple programming language for dialog creation on the NAO. It is a syntax driven language and it can also be very powerful and complex for advanced usage. The basic idea is to relate human input to robot output. As shown in the tutorial, the dialog is implemented in the `topic_name_enu.top` file that comes with the creation of the dialog box. The two lines

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

⁶http://doc.aldebaran.com/2-1/software/choregraphe/tutos/dialog_topic.html

on top of this file define the name of the box and the language, and are generated automatically.

Important elements of the QiChat syntax:

- **User rule:** It defines the responds by the robot to human inputs and starts with the keyword `u:`. In the round bracket the input that the robot listens to is defined and after that the answer as output from the robot.

```
u: (Hello) Hi
u: (Let us talk about sports) Ok
```

- **User subrule:** With subrules, a hierarchical structure can be created. They are activated one after the other and defined with sequential numbers:

```
u: (Hello) Hi
u1:(Goodbye) Bye bye
u1:(How are you?) Good, what about you?
u2:(Good) I am glad to hear that.
u: (Let us talk about sports) Ok
```

- **stayInScope:** Staying in the current subrule level using the keyword `^stayInScope`.

```
u: (Hello) Hi
u1:(Goodbye) Bye bye ^stayInScope
u1:(How are you?) Good, what about you?
```

- **Choice:** []: The words in the square brackets can be substituted for each other. In the output, the elements are selected one after the other in case of multiple calls.

```
u:([hi hello]) [hello hi] human
```

- **Phrase:** "": Allows to place a phrase instead of a single word within the delimiter.

```
u:([{"hello how are you" "hello are you OK"}) ["I am fine" "I am
OK"]
```

- **Optional part:** { }: In the middle of the input sentence there can be an optional word or phrase. This only makes sense in the middle of a sentence, otherwise you could simply omit this rule.

```
u:(hello {buddy} how are you) hello I am fine
```

- **Concept:** Collects synonyms or words that trigger the same reaction in a static list. It can contain **choices**, **optional parts** and **phrases**.

```
concept:(greetings) ~rand[hi hello "hey there"]
u:(~greetings) ~greetings
```

The concepts are defined before the dialog block consisting of rules.

- **Proposal:** Robot output that is activated by a function from a robot output.

```
proposal: First do this
proposal: Then do that

u:(Give some instructions) Ok. Just say next.
u:(next) ^nextProposal
```

- **Input storing:** Safe human input for the following output of the robot.

```
u:(sentence _[word1 word2]) answer $1

u:(sentence _~concept) answer $1

u:(sentence _*) answer $1
```

The underscore `_` is the keyword for the input storing and `$1` refers to the first caught word (`$2` to the second and so on if multiple `_` are used). The reference to a concept allows catching any word from it, whereas `*` allows any said word.

- **Conditions:** The last example shown here is the condition, which is also a bit more complex and combines several other concepts. It can be used, for example, to compare an input and generate an output accordingly

```
u:(I want some _[chocolate cheese]) OK, you want some $1
    $askedFood=$1
u:(can I have more)
    ["$askedFood==chocolate sorry, too much chocolate could hurt
        you"
    "$askedFood==cheese yes, please take more $askedFood"]
```

Here, at first a word input is stored to a variable declared with the key `$`. Next, the output depends on matching the variable. The condition could also be used in the user input to trigger the rule, if the condition is met. Of course, there are also other logical operators than `"=="`.

As already mentioned, the individual elements can be combined in all possible ways. The list shows only an excerpt of the possible elements that the QiChat syntax offers.

Exercise 5.

Implement a dialog for the simulated NAO robot. You can use the code elements from the explanations above. After each step, you can try the code by running the program in Choregraphe and typing the user inputs into the Dialog sub window (If this is not there yet, activate it with View>Dialog in Choregraphe). Following those steps:

- a) Create the basic structure of the dialog with hierarchical **rules** and **subrules**.
In the dialog, there should be a greeting and then talk about a subject such as sports. Define at least three rule branches and three hierarchical subrule levels for each rule branch. Figure out if you can access another branch while being in a subrule. Use **stayInScope** at least once in the dialog.
- b) For each rule of the dialog, replace the single words with at least one **choice**, **phrase** or **optional part** if it makes sense.
- c) Use a **concept** for the greeting. Add a dialog **rule** to say goodbye at the end and create a **concept** for that as well. What is the **^rand** keyword for?
- d) Put at least three **proposals** above the **rules**. Create them so that you can call them one after the other within a subrule.
- e) Add an **input storing** element to your dialog. This could for example be to input a particular sport, that appears with the robot's output again.
- f) Now, add a similar element but store the input into a variable first. Then use a **conditions** for comparison and different possible outputs.
- g) Run a final test of the dialog. Test every functionality you created. Save the Choregraphe project.

Add a screenshot of the code and of the dialog output.

More information to the syntax of QiChat and advanced functionalities are in the documentation online⁷. However, the QiChat elements presented in this section are enough to carry out the next exercises in the lab.

⁷http://doc.aldebaran.com/2-1/naoqi/audio/dialog/dialog-syntax_full.html#dialog-rules

Task: Execution

In this part of the task the elements learned in the preparation are applied to the real robot. However, before starting the robot must first be put into operation.

Exercise 6.

Prepare the robot.

- a) Start the NAO robot with pressing and holding the button on the robot's chest.
- b) Open Choregraphe on the lab computer.
- c) Connect the computer to the NAO via the WIFI connection. The robot should automatically connect to the network.
- d) In Choregraphe, connect to the real robot within the Connection menu (name: Noah or Robin). If no robot is listed, press the button on the robot's chest once and enter the IP address manually.

After the robot is connected, you will create a small speech recognition system with individual boxes first and then the dialog language is used on top of it. For this task's project use the directory *Desktop/NAO-files/<group-name>* or create, if not there yet.

Speech recognition with the NAO robot

On the virtual robot, the speech input box could not be used because it requires access to the robot's microphones. This will now be made up for in an introductory way to get to know this possibility of graphical programming and to get a feeling for the capabilities of the robot, how it perceives speech.

Exercise 7.

Follow the provided instructions in *speech_recognition.pdf* and use them to program a simple speech recognition script in Choregraphe. You can also use your program from Exercise 3 from the preparation as a template. This would basically mean to replace the Choice boxes with Speech Recognition boxes. Run it on the robot. Each participant from the group should test the program sequence. Try out how the parameter *Confidence threshold* of the *Speech Reco.* box affects the results. Add a sit down box in the beginning of your program to make sure that the robot is in the sit position.

This and the following exercises are to be performed with the robot in the sitting position, as this allows it to be operated at a comfortable dialog height on the table. If you later want to run programs on the robot while it is standing, then you can do this on the floor with the underlay mats.

Dialog situation with the NAO robot

In the next parts a dialog situation should be implemented and tested using the dialog language QiChat. As you could see in the exercises before, a graphical structure of extended dialog is difficult. Therefore, the following exercise applies the script-based implementation on the robot. The idea of this task is to think about, how to write a dialog that a user can have with the robot, not knowing how it was implemented. Therefore, each person from your group is supposed to implement a short dialog themselves and these will then each be tested by the other two.

The implementation of the dialog should remain linear in this task. In this case, the dialog is structured with rules and subrules and these are run through to the end one after the other. However, QiChat would also offer further possibilities to jump from and to different dialog sections.

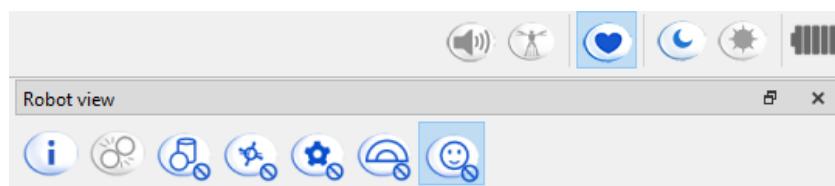


Figure 3: Robot settings in Choregraphe.

Exercise 8.

In this exercise, a dialog for the interaction between humans and the NAO robot should be implemented using the QiChat language. Imagine the situations from the gray box below.

- Each person of your group, choose a dialog situation from the gray box.
- Implement the dialogs for the situations on separate computers. Before you start, sketch the hierarchical structure of the dialog in order to realize it with user rules and subrules afterwards. Note this structure as a cheat sheet for the others later. While implementing keep in mind to create it for users that are unfamiliar with it.
- Save the projects.

Below are the dialog situations that you can use to guide the implementation. Use the elements for the QiChat syntax from the preparation. It is helpful to first break down and structure the dialog into the rules and subrules.

Dialog situation:

- Discussion on where to go for lunch:** Imagine meeting your robot friend NAO on campus at noon. After a greeting, you want to know from it what food options are available. Thereupon the robot asks what you feel like. Depending on your answer, the robot should react accordingly. For example, it should advise you to go to another place, agree with you, or react neutrally. Think about how one can react to it in each case. The goal should be to agree

on one possibility at the end.

- **Discussion on having a coffee somewhere:** Implement this in such a way that the robot first asks whether you would also like a coffee. If not, it should convince you or offer alternatives. As soon as you agree, NAO should make two suggestions. You should be able to ask it what the advantages and disadvantages are and then agree on a place.
- **Ask the robot for directions.** Imagine you are on the red square of the technical faculty and you have to go to the Chair of Autonomous Systems and Mechatronics for the lab course, but you do not know the way. You meet the NAO robot and ask it for directions. You should be able to ask it about the possibilities first. NAO should then ask whether you have a car or a bicycle at your disposal. For each option, it should be able to answer you how long the trip will take. After explaining the way, it should also be able to tell you where exactly the room is located.
- **Find out which degree program NAO is studying.** At the beginning of the dialogue you should be able to ask what NAO is studying. The robot should then respond with a counter-question, for example what you are studying yourself. In turn, it should be able to react to the answer, depending on whether you are studying the same course of study or a different one.

Exercise 9.

Experiments with the dialogs.

- a) Copy the projects to the computer connected to the robot.
- b) Switch off the autonomous life and the animation mode of the robot with clicking the heart symbol and the person symbol above the *Robot view* window of Choregraphe as shown in Figure 3. You are supposed to do this so that the dialog takes place purely via speech recognition and output.
- c) Run all dialogs on the robot and test the dialogs that are still unknown to you. So, those who did not write the dialog are supposed to interact with the robot. Go through the entire dialog in each case. If you get stuck, ask the author.

Please note, that the default value for *Confidence threshold* of the dialog box is 50 %. You can adapt it by copying the Python script box from the provided project file into your project and placing it before the dialog box.

Extension of the dialogue with non-verbal elements

For the next exercise, the autonomous life of NAO will be turned on. Thus, the robot makes small movements throughout to create the impression of liveliness. This function generates lively behavior even if no additional animation is currently running. Additionally, nonverbal elements should be integrated into the dialog. This means that the robot reacts to the dialog content with facial expressions and gestures as far as the hardware

of the NAO allows. Possible animations for NAO are listed in the documentation⁸⁹. Important elements to integrate those animations of the QiChat syntax for non-verbal interaction are:

- **Annotated text:** It is a string for the robot output combining text and functions for robot animations.

```
u: (Hello) "Hello! ^start(animations/Stand/Gestures/Hey_1) Nice  
to meet you ^wait(animations/Stand/Gestures/Hey_1)"
```

The start function starts the animation while NAO is talking and the wait function keeps it running until it ends, even if the text was already spoken.

- **Tags:** The related animations are also collected into tags which can also be seen on the documentation.

```
u: (Hello) "^startTag(hello) Hello Paul, nice to meet you.  
^stopTag(hello) My name is Nao."
```

This uses the same functions but the animations are chosen randomly from the tag.

- **Run, Start, Stop and Wait** Using the keyword "start" starts an animation while NAO is talking and "stop" stops it of course. The "run" keyword suspends the speech, runs an animation and resumes the speech, whereas "wait" suspends the speech when there is already an animation ongoing until the end of the animation. This can be added to the end of the string to let the animation be finished.

Note that not all animations are available in the sitting position. Therefore, use the provided list for animations and tags for the next exercise.

Exercise 10.

Extend a dialog to include the nonverbal elements explained above.

- Choose one of the dialogs from Exercise 9
- Get an understanding of what the difference is between animations and tags.
- For each robot output, add at least one nonverbal element. Therefore, choose animations and tags that match what the robot is saying.

Run it on the real NAO:

- Turn on the autonomous life and the animation mode in Choregraphe.
- Talk through the dialog together with the robot.

Exercise 11.

Read chapters 6.2 (Nonverbal Interaction), 7.2 and 7.3 (Verbal Interaction) of the book *Human-Robot Interaction*¹⁰. Answer the following questions:

⁸<http://doc.aldebaran.com/2-5/naoqi/audio/alanimatedspeech.html>

⁹<http://doc.aldebaran.com/2-5/naoqi/motion/alanimationplayer-advanced.html>

- a) What parts of this chapters align with your experience with the NAO robot?
What did you experience different in the interaction?
- b) How did you experience the interaction without non-verbal elements compared
to with non-verbal elements?

¹⁰Bartneck, C. Human-Robot Interaction: An Introduction. Cambridge: Cambridge University Press. (2020). <https://www.human-robot-interaction.org/>