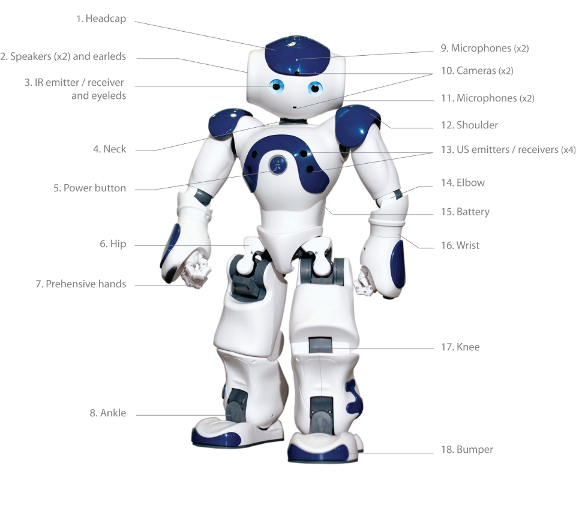
Nao



Contents

1. Ways to work with the Nao 3

1.1. With Choregraphe only 3

1.2. Through SSH connection 3

1.3. Using the SDK 4

# Ways to work with the Nao

First read <http://doc.aldebaran.com/2-1/dev/naoqi/index.html> for a good introduction on the NAOqi framework.

Summary: NAOqi is a framework with many modules (shared objects libraries) with functionality like motion, vision, audio, memory etc. They can be accessed by creating a proxy to this module first. The proxy can run on the robot or remote on a laptop. The proxy will connect through TCP/IP with a broker running on the robot. The broker will have the list of available libraries loaded (in ‘autoload.ini’). The libraries contain modules and methods and the broker will pass the call to the prober library. There are two brokers: the main broker (NAOqi broker) for local calls and myBroker (Python broker) for remote calls. When not running in the NAOqi environment but in a standalone Python script, myBroker() is required. This myBroker connects to the main broker.  
Normally the Nao libraries are developed in C++ and the behaviors in Python.

## With Choregraphe only

In Choregraphe a Nao behavior is created by using boxes and connecting them. One can use existing boxes or create new ones. The types are: Diagram, Timeline, Python or Dialog box. A box can contain classes derived from ‘GeneratedClass’ with standard methods such as ‘onInput\_onStart()’ or ‘’ onInput\_onStop(). When the behavior executes the classes are instantiated and the box methods are called when the input is stimulated. The underlying event mechanism is configured in Choregraphe when connecting two boxes.

One can write a Python script which makes calls to the Nao API through proxies. Example call to obtain a text to speech proxy: ‘tts = ALProxy("ALTextToSpeech")’. Note that in this case no IP address and port is required as the Python module and the ALTextToSpeech module are both in the main broker. This proxy can then be used to make calls to the text to speech API, for example: ‘tts.say("hello World!")’.  
The behaviors are located in ‘/home/nao/.local/share/PackageManager/apps’. Behaviors are in ‘.xar’ files which are ‘.xml’ files containing Python code within <content> tags.

Debugging the Python script can be done printing to the Choregraphe Log viewer like: ‘self.logger.info("\*\*\*\*\*\* This is a info message \*\*\*\*\*\*")’.

## Through SSH connection

With e.g. ‘ssh nao@192.168.1.137’ and password ‘nao’, one can connect with a laptop to the Nao. Using FileZilla it is possible to transfer files between the laptop and the Nao. The home directory is ‘/home/nao’.

One can write a standalone Python script and execute it from the Nao command terminal. Print statements will show in the terminal. One can make calls to the Nao API through proxies. For example a call to obtain a text to speech proxy: ‘tts = ALProxy("ALTextToSpeech", “localhost", 9559)’. This proxy can then be used to make subsequent calls to the text to speech API, for example: ‘tts.say("hello World!")’.  
When transferring this standalone Python script to a Choregraphe Python script inside a box, please note the following:

* The main function of the standalone script has to be moved to onInput\_onStart().
* Put initialization code in \_\_init\_\_(). This function is the constructor and will automatically be called.
* Make sure to put cleanup code in onUnload().
* Print statements will not show but if needed can be replaced with ‘self.logger.info()’.
* In Choregraphe, events are simply connected in the GUI. To make use of events in a standalone Python file one has to create a module class derived from ALModule with a callback function. This callback function is called when the event occurs. Subscribing to events is done via the memory proxy. The module is registered to the broker by calling ‘ALModule.\_\_init\_\_(self, name)’ in the constructor. It is important that before ending the exit() function of these modules is called! Otherwise the next time ALModule.\_\_init\_\_(self, name) will fail because the modules are already registered!

Until now for me this method is the most straightforward and convenient way to develop for the Nao One can develop Python scripts with print statements on the laptop and run it on the Nao. When the script runs fine, it is easy to put in in a Choregraphe Python box. In Choregraphe one can combine it with other boxes.

## Using the SDK

Installing the SDK on a laptop gives the most freedom in working with the Nao. It is possible to develop behaviors in languages like C++, Java or Python. The behavior can be run from the laptop. This means the Nao is controlled from the laptop which makes calls to the Nao API through proxies. Example call to obtain a text to speech proxy: ‘tts = ALProxy("ALTextToSpeech", “192.168.1.137", 9559)’. The IP address and the port number are of the remote broker ‘myBroker’.

At the moment OS X El Capitan is not supported.