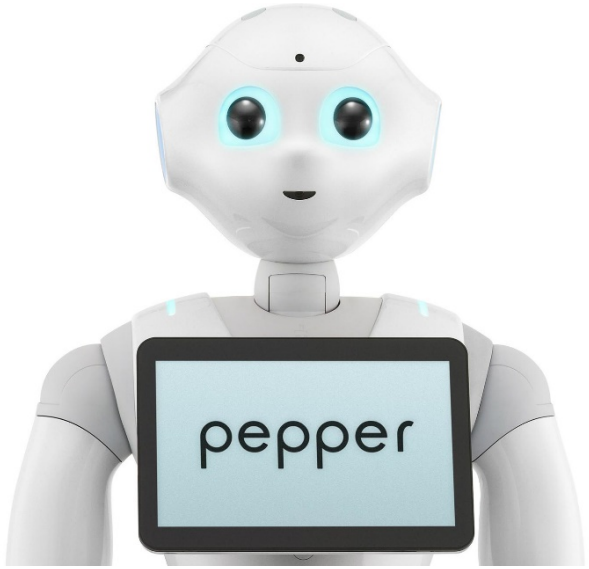
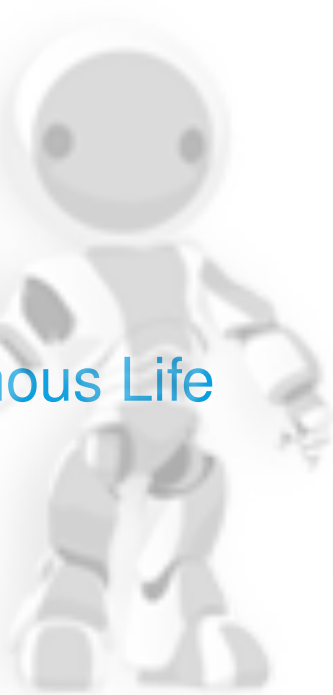
Introduction to Pepper Application Development



SoftBank Robotics

Training Session 2017

Introduction



Outline

Introduction

Presentation:

- Who am I ?
- Who are you ?
- What do you expect to do with Pepper ?

Introduction

Goals of this training:

- Give you a complete overview of Pepper's capabilities
  - Hardware and software
- Configure, connect, update and change settings of Pepper
- Use development tools to build integrated behaviors
  - Create, structure, debug and launch

Introduction
Foreword

What are we going to do during 3 days ?

Day 1

- Hardware and how to wake up your robot
- Setting up tools
- Discover Pepper main features
- Create Applications, Behaviors, Activities and Autonomous Life
- Understand ALMemory
- Some Python and NaoQi APIs

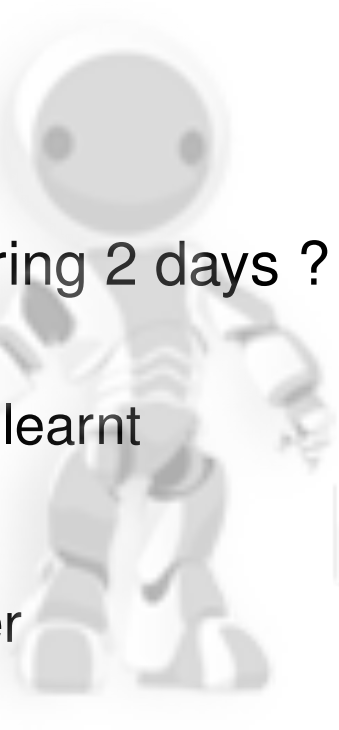
Day 2

- Build dialogs with qichat
- Discover the tablet and javascript
- More Python and NaoQi APIs (Python App Structure)

Day 3

- Build Animations with timeline
- Packaging an app

Introduction
Foreword



What are **you** going to do during 2 days ?

Day 4 and 5

- Consolidate what you learnt
- Build a demo
- Answer a use case
- Build the best app ever

Introduction
Foreword

- Founded in 2005 by Bruno Maisonnier
- World leader in academic humanoid robotics
  - Partnerships with research labs
  - A dynamic programmers community
- Headquarters in Paris, branch offices in China, Japan and United States

SoftBank Robotics

Introduction
Foreword

- Pepper is a humanoid robotic platform
- Project started in 2012
- 121 cm height

SoftBank Robotics

Introduction
Foreword

Pepper robot:

- Announced 2014 June 6th in Tokyo
- Developped for Softbank

SoftBank Robotics

Introduction
Foreword

Pepper robot:

- Emotion recognition capabilities
- Software compatibility with NAO
- First used for customer service in stores in Japan
- Sold in Europe from 2016

SoftBank Robotics

Introduction
Pepper's presentation

- Version: 1.8 (early 2017)
- Height: 1.2 m (max: 1.35 m)
- Depth: 0.425 m (max: 0.65 m)
- Width: 0.48 m (max: 1.20 m)
- Weight: 27.8 kg

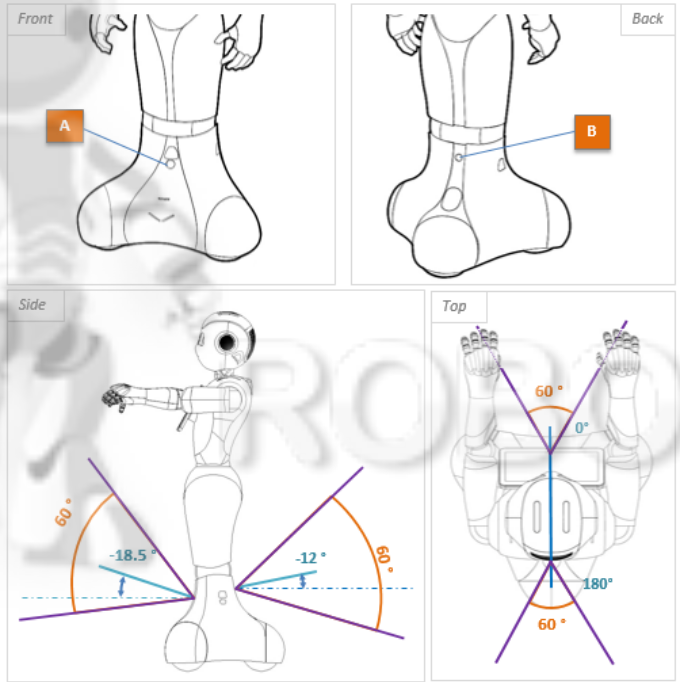
Introduction
Pepper's presentation

Sensors:

- Capacitive sensors on the head and the hands, 3 bumpers
- 4 directional microphones, accelerometer
- 2 RGB camera, 1 depth sensor in the eyes
- Tactile tablet

Introduction
Pepper's presentation

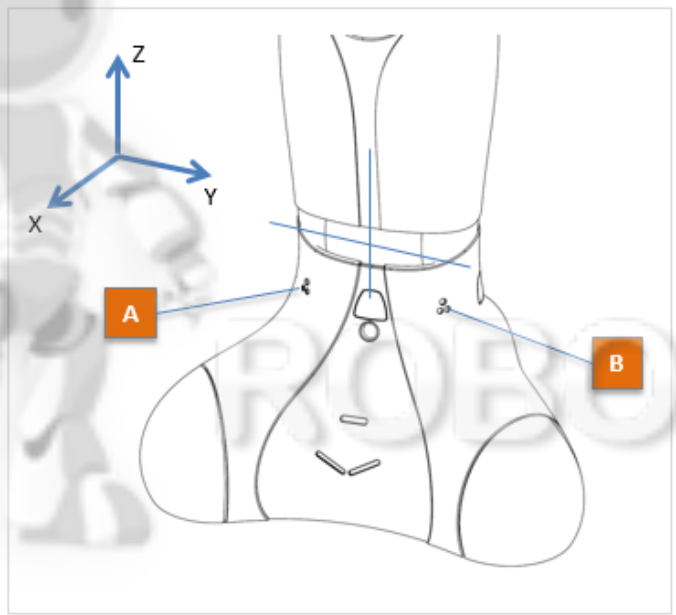
2 Ultrasound distance sensors



SoftBank Robotics

Introduction
Pepper's presentation

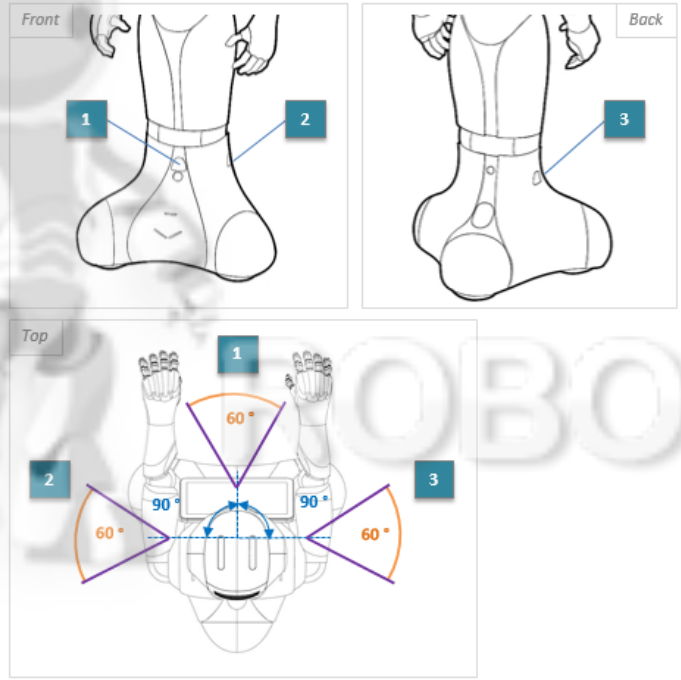
2 Infrared distance sensors



SoftBank Robotics

Introduction
Pepper's presentation

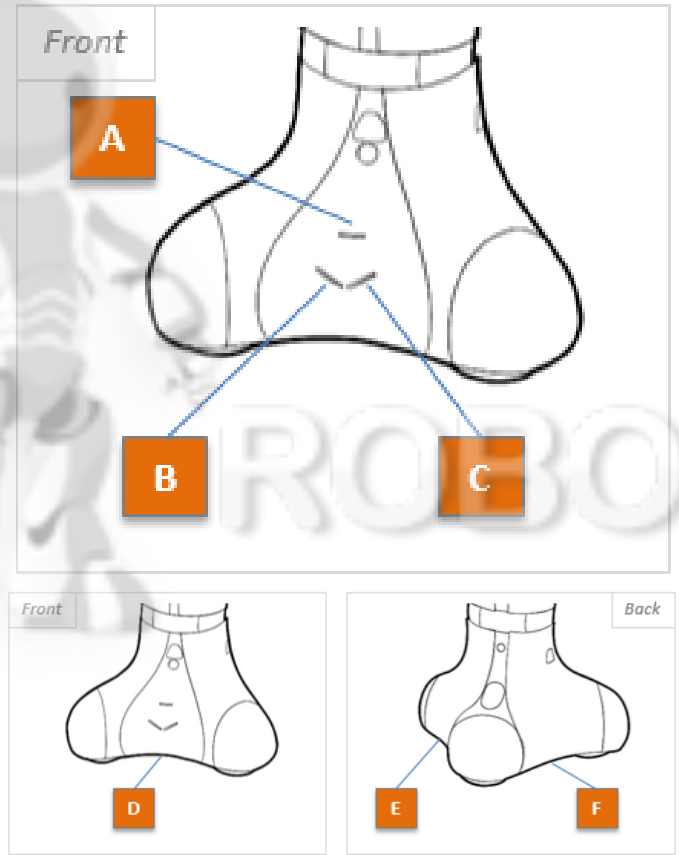
3 laser sensors



SoftBank Robotics

Introduction
Pepper's presentation

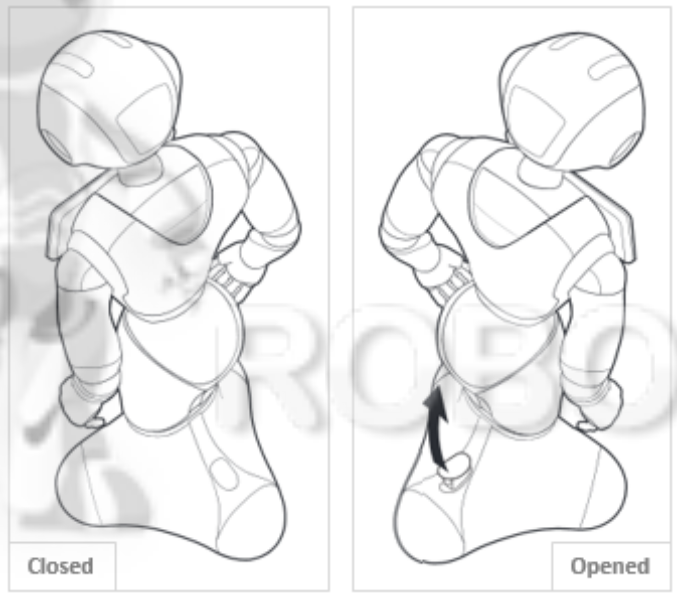
6 laser emitters



Introduction
Pepper's presentation

Power Hatch

- Pepper cannot move if opened
- Always open hatch when moving Pepper by hand



SoftBank Robotics

Introduction
Pepper's presentation

Actuators list:

- Head, arms, speakers
- HipRoll, HipPitch and KneePitch joints
- 3 holonomic wheels

With:

- Real temperature probes for each motor
- Bigger battery (10-12h autonomy)
- Two leds on the shoulders
- Leds around the eyes and ear speakers

SoftBank Robotics

Introduction
Pepper's presentation

- Intel(R) Atom(TM) CPU E3845 @ 1.91GHz with 4 cores
- 4 GB DDR3 of RAM and 8 GB eMMC of Flash memory
- Harddrive 16 GB Micro SDHC
- GPU Intel HD graphics up to 792 MHz
- No legs, only Hip/Knee joints

Introduction
Pepper's presentation

Turning Pepper on, off and change its state

- When Pepper is off
  - Press chest button briefly (led turn on) : let the robot boot ; ready at 'Ognagnouc' sound
  - Press chest button up to 8 seconds, shoulder leds get blue, release : controllers reset, long startup, data is unchanged
- When Pepper is on
  - Press chest button 2/3 seconds : stop, robot say 'nocnoc'
  - Press chest button once : robot says it's name, ip and notifications
  - Press chest button twice : stops/starts Autonomous Life

Introduction
Pepper's presentation

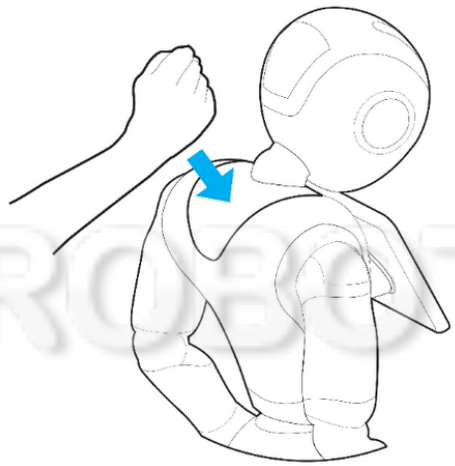
Interrupt an application, go to/leave rest mode (when Pepper is active and Autonomous life is on)

- Cover forehead camera and tactil head :
  - release on first 'blop' sound : stop current running application
  - release after 'blop' 'blop' 'clic' sounds : goes to rest mode
  - touch head sensors to come back to activity

Introduction
Pepper's presentation

Emergency stop, cut power:

- Emergency stop button on the back of the neck
- Press behind the neck on the soft plastic area to cut power off in emergency situation or for transport

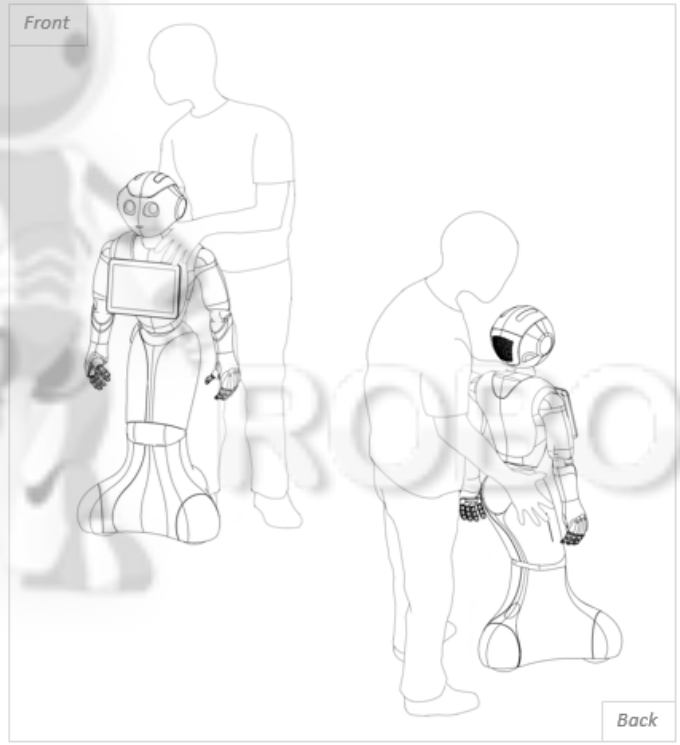


SoftBank Robotics

Introduction
Pepper's presentation

Moving the robot:

- a hand on his shoulder
- and another on its hip

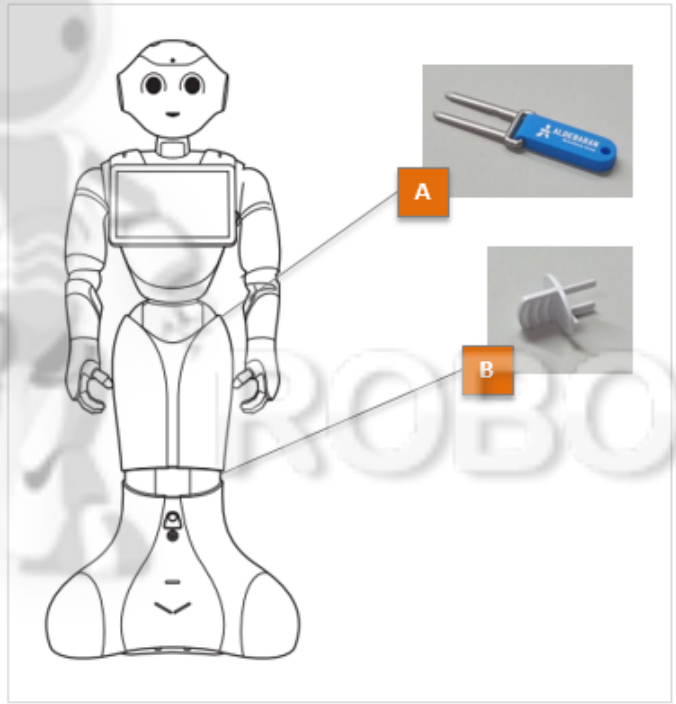


SoftBank Robotics

Introduction
Pepper's presentation

Releasing brakes for transportation:

- always keep a hand on pepper to prevent it from falling
- insert the hip key
- insert the knee key
- put pepper in Slumping posture



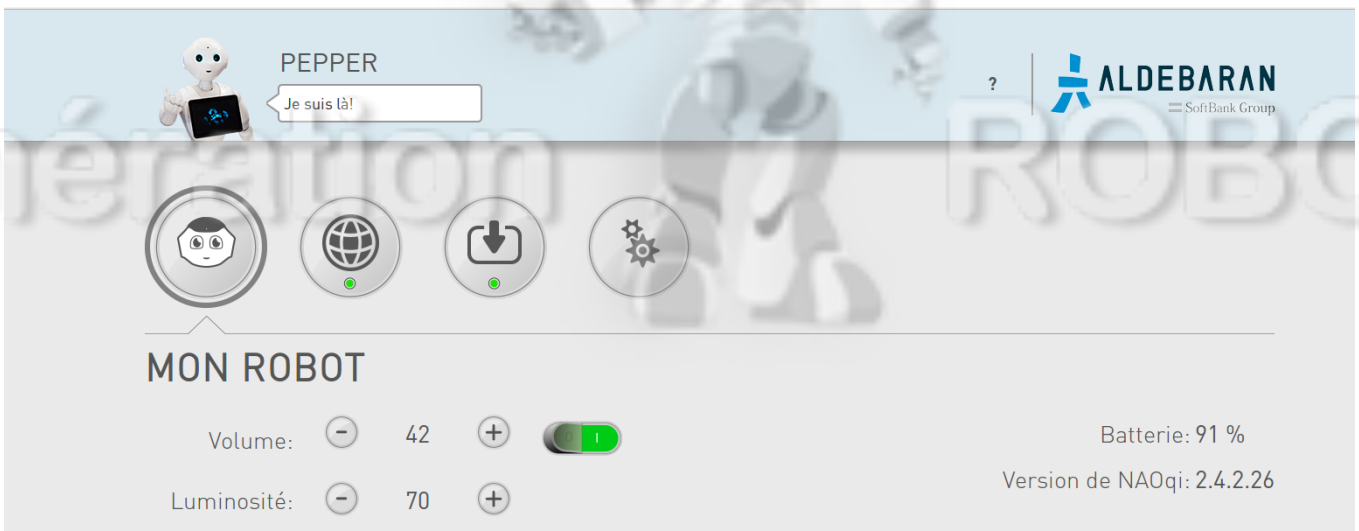
SoftBank Robotics

Introduction
Configuration and Maintenance

- Pepper's webpage allows you to configure your robot
- Access it from your web browser: *http://<robot_name.local>/*
  - Alternative: use IP address *http://<ip_adress>/*
- Log in with user *nao* and pass *nao* (default account)
- Voice command is also possible:
  - "Launch settings" voice command will display the settings on the tablet
  - To turn volume up/down say "Speak louder" or "Speak softer"

Introduction
Configuration and Maintenance

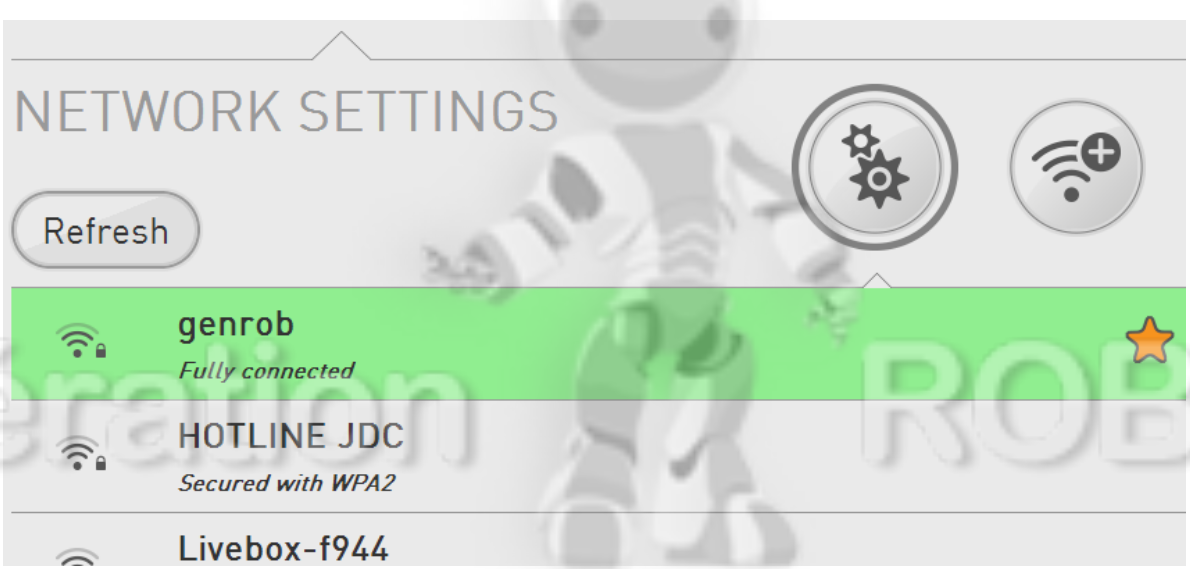
- The upper part shows the name of the robot
- The four icons lead to 'my robot', 'network', 'applications' and 'settings' pages



SoftBank Robotics

Introduction
Configuration and Maintenance

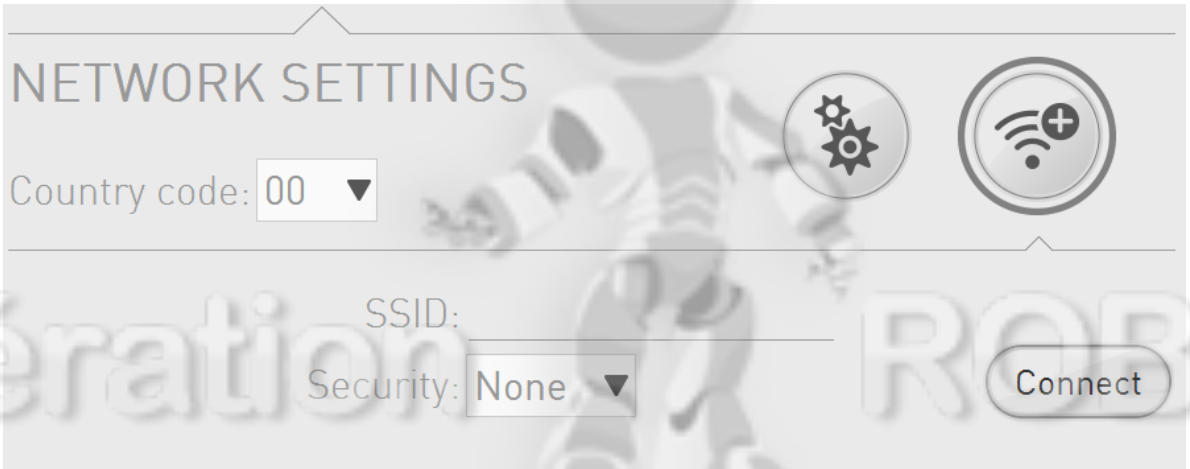
network page:



- Click on a wifi SSID to connect
- The robot will try to connect by itself next time

Introduction
Configuration and Maintenance

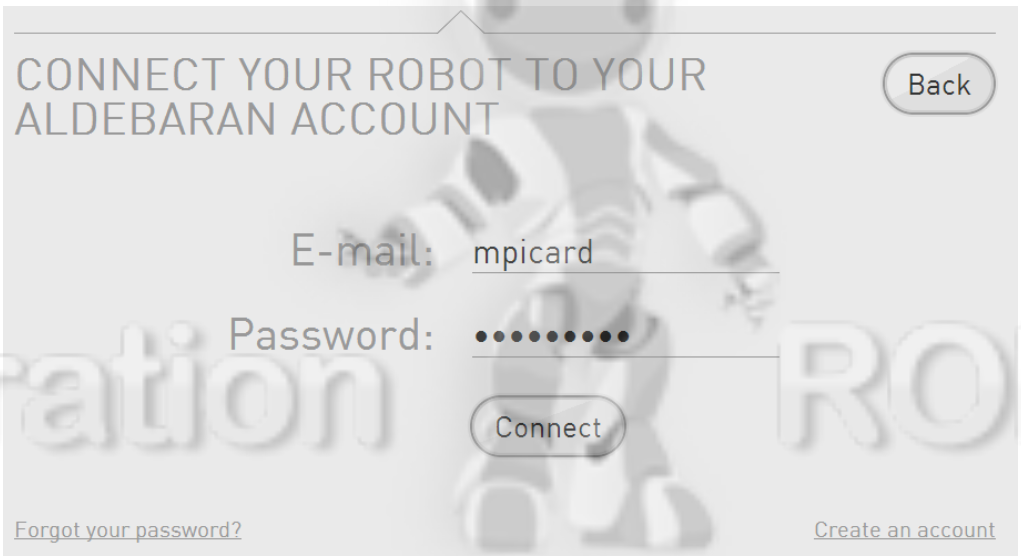
Pepper's wifi:



- Pepper can create its own Wifi network
- Choose the name and security and click connect

Introduction
Configuration and Maintenance

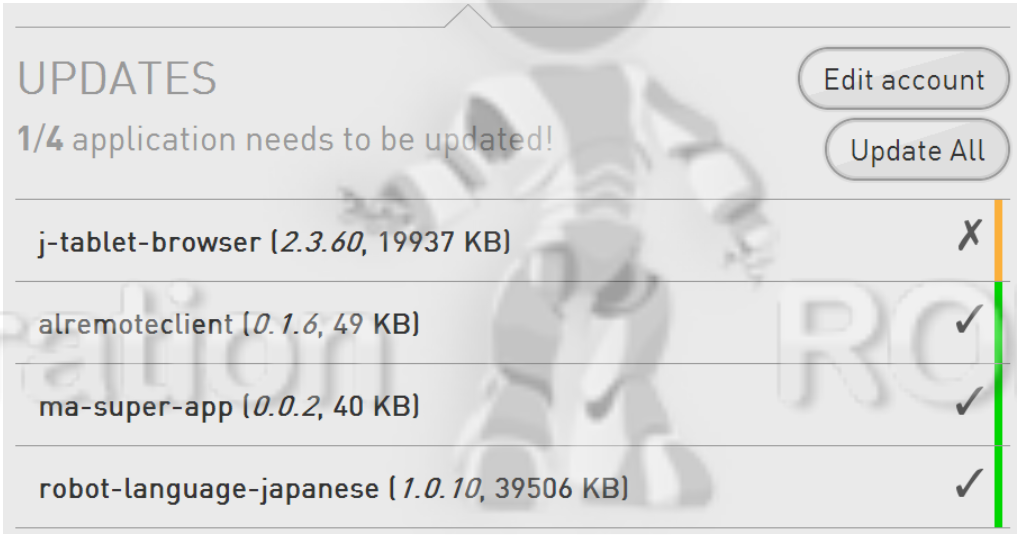
Applications page:



- Connect your robot to your community account
- This will also link the robot to the store

Introduction
Configuration and Maintenance

Applications page:

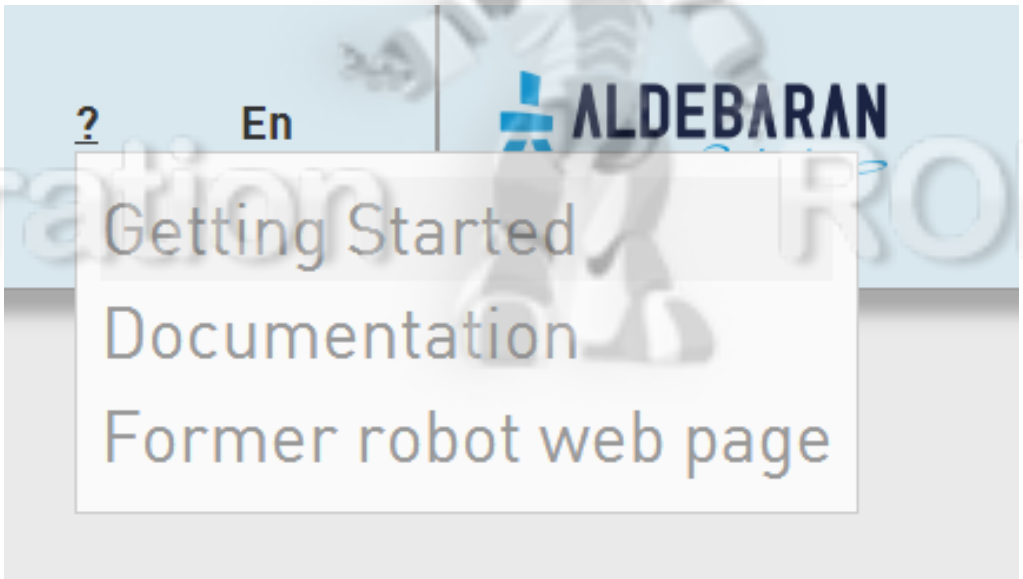


- You can then view the installed applications

Introduction
Configuration and Maintenance

The ? sign is a link to some useful pages

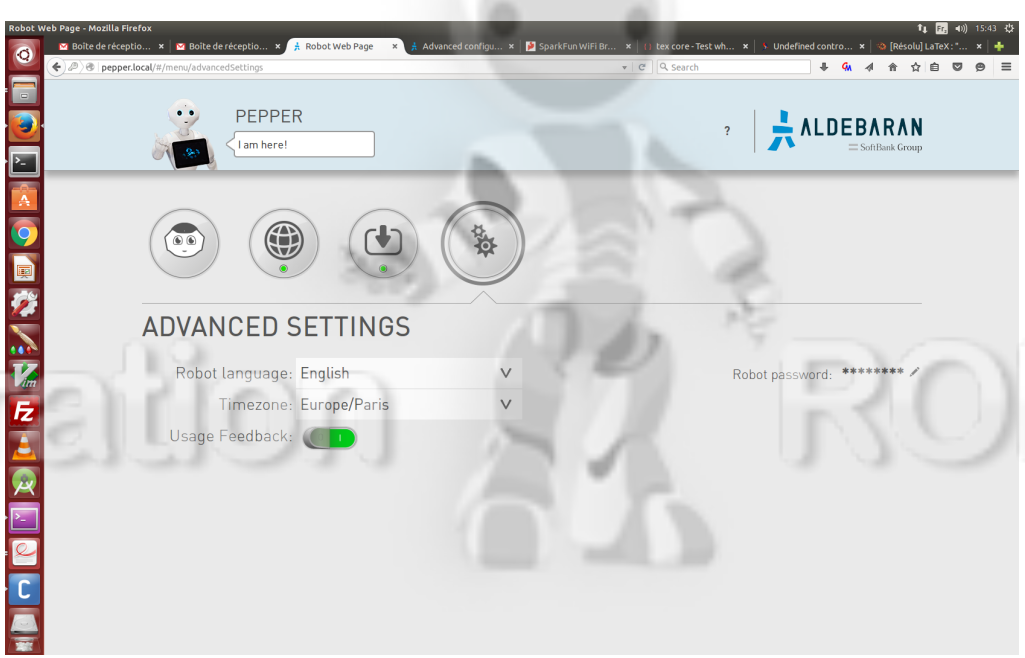
- Including the 'former robot web page' with more info for developpers



SoftBank Robotics

Introduction
Configuration and Maintenance

Settings page:



- You can change the language and timezone

Introduction
Configuration and Maintenance

Flashing the robot:

- Needed if
  - You don't use the auto-update feature
  - Your robot has no internet access
  - The system image of your robot has been corrupted
- Simple reflash will keep your data
- Factory reset will erase everything
  - Including password, robot name, store account...

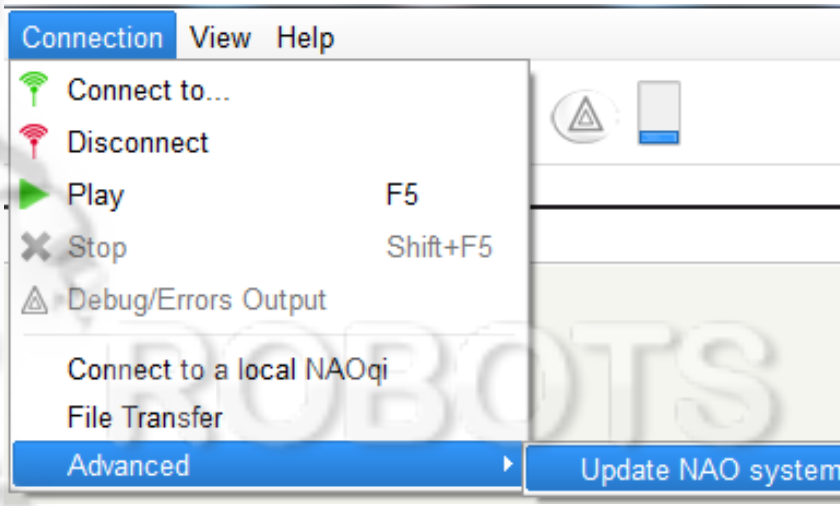
Introduction
Configuration and Maintenance

Updating the robot:

- Current Pepper software version is 2.5.5.5
  - New versions can be downloaded online on `https://developer.softbankrobotics.com/us-en/downloads/pepper`
- Install Choregraphe Software Suite
  - Always use the same version for Choregraphe and NaoQi
- Download Pepper system image

Introduction
Configuration and Maintenance

- Connect Choregraphe to Pepper
- Connection / Advanced / update Pepper system, then select system image file
- Wait for transfer to finish then reboot Pepper
- Flashing lasts 10-15 minutes



Warning

Follow the instructions and be patient. Forcing Pepper to reboot while downloading or flashing may crash the robot.

Introduction
Configuration and Maintenance

You can also flash your robot from the settings page clicking on the Factory reset button

- To display the settings page on the tablet launch BootConfig behavior from the robot application list

Warning

Follow the instructions and be patient. Forcing Pepper to reboot while downloading or flashing may crash the robot.

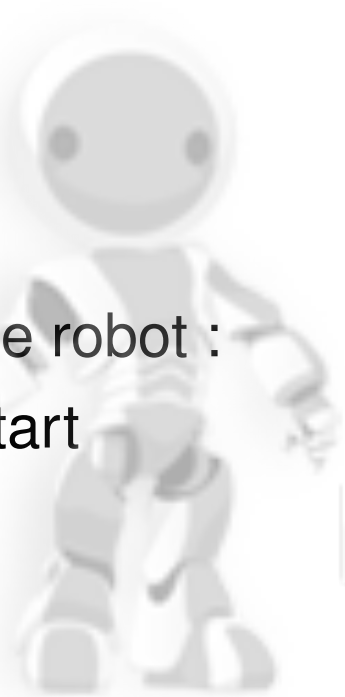
Introduction
Configuration and Maintenance

Connect to the robot :
SSH Commands

- OS on the robot is a custom linux Gentoo distribution
- connection with ssh
  - $ ssh nao@mypeppername.local / ssh nao@<ip_adress>
  - password is 'nao' (if you modify it don't lose it, no root access on pepper to reset it)
- Opens a shell on Pepper's Linux OS, allows to run administrative tasks, troubleshoot issues, etc. (ex: "nao restart", acces logs, ...)
- Requires a good knowledge of Linux

SoftBank Robotics

Introduction
Configuration and Maintenance



Useful shell commands on the robot :

- naoqi restart : $ nao restart
- reboot : $ sudo reboot

Introduction
Configuration and Maintenance

FTP File Server is also available

- Connect with `ftp://pepper.local` in web browser or use other FTP client
- Default user/password: `nao/nao`
- Allows to browse Pepper's filesystem
- Upload and download file between Pepper and computer

Introduction
Configuration and Maintenance

QICLI

- QiCli is installed on the robot and on your Choregraphe install dir
- $ ./choregraphe-suite-2.5.5.5-linux64/bin/qicli
  - use from the robot
  - use from your computer and connect remotely to naoqi
- Command Syntax : qicli cmd args --qi-url IPofYourNaoqi
- For details : qicli --help

Introduction
Configuration and Maintenance

QICLI Examples

- qicli info : display info on services
  - qicli info
  - qicli info ALMemory
- qicli call : execute module/services functions
  - qicli call ALMemory.getData Dialog/RobotName
- qicli watch : listen to signal or events
  - signal => qicli watch PackageManager.onPackageInstalled
  - events => qicli watch --almemory PeoplePerception/JustArrived

`http://doc.aldebaran.com/libqi/guide/qicli.html`

Introduction
Configuration and Maintenance

How to interact with the robot from command line with qicli :
Change your robot name :

- $ qicli call ALSystem.setRobotName peppermint
- $ qicli call ALStore.update
- $ qicli call ALSystem.reboot
- $ qicli call ALMemory.raiseEvent FrontTactilTouched 1
- $ qicli call ALTextToSpeech.say "oh oh !"

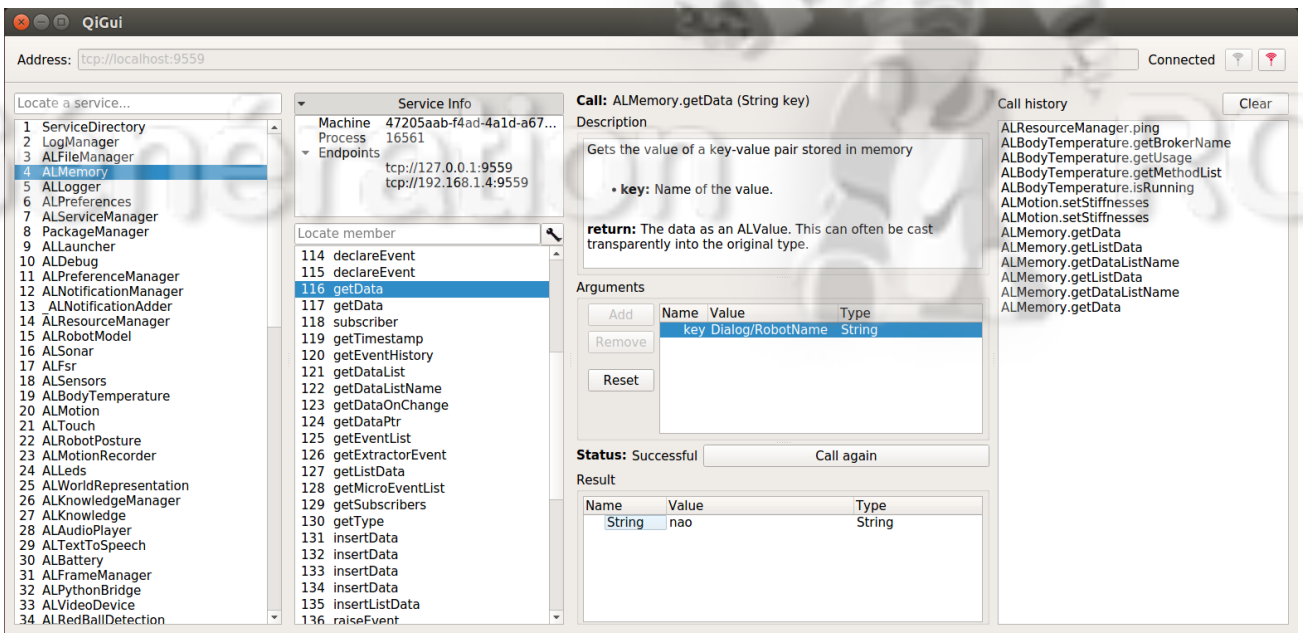
`http://doc.aldebaran.com/libqi/guide/qicli.html`

SoftBank Robotics

Introduction
Configuration and Maintenance

QIGUI is a graphical interface comparable to qicli

- QIGUI
  - located in your Choregraphe installation dir in choregraphe/bin directory
  - not available on mac



Introduction
Configuration and Maintenance

Laptop configuration instructions :

- Set up latest Choregraphe software suite
- Set up python 2.7 from python.org
- Set up naoqi python sdk
- Install an editor for python (ultraedit, pycharm, ...)
- Set up laptop configuration to launch python from command line (PATH/PYHTONPATH ...)

Follow documentation for installation here :

- `http://doc.aldebaran.com/2-5/dev/python/install_guide.html`

SoftBank Robotics

Introduction

Notions summary / Key concepts

- Hardware details
- Move and transport Pepper
- Access setting page of the robot
- Voice commands
- Connect with ftp or ssh

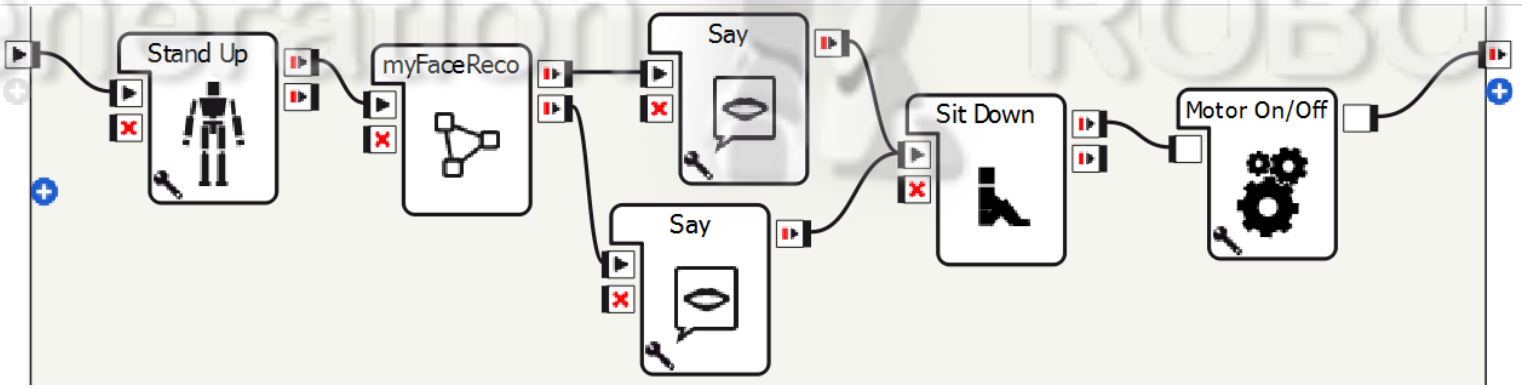
Outline

Applications and Autonomous Life

Applications, Activities and Autonomous Life

We want to create a complete behavior using face recognition.
For example:

- Stand up and use face recognition
- Say 'Hi' if someone is seen and 'I see no one' otherwise
- Sit down and release stiffness

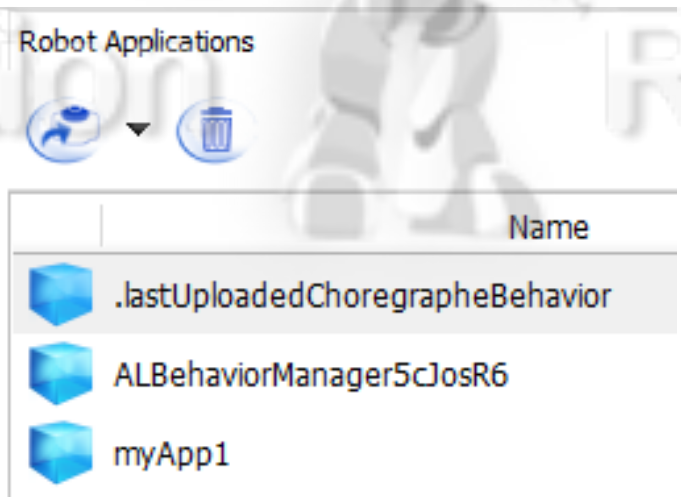


SoftBank Robotics

Applications and Autonomous Life

Applications, Activities and Autonomous Life

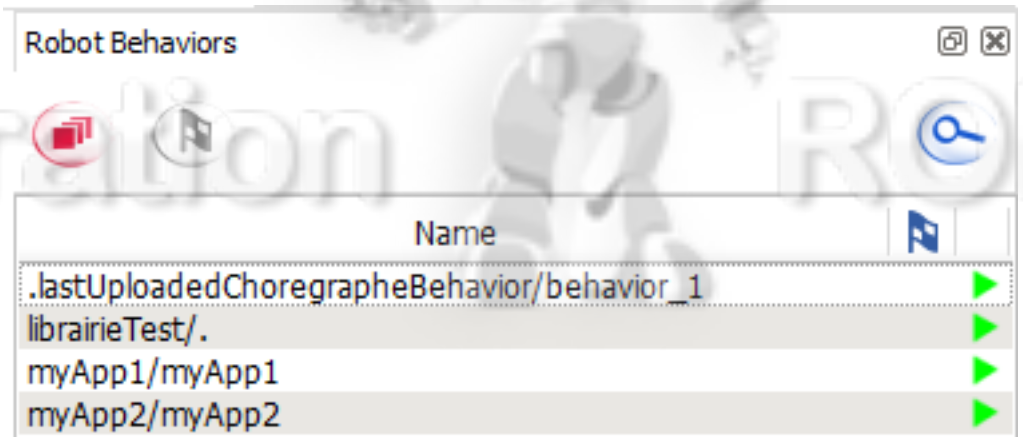
- Go in the Robot Application menu
- Use 'package and install current project to the robot' button
  - This will store your project on the robot
  - You have a bin button to delete an application



Applications and Autonomous Life

Applications, Activities and Autonomous Life

- Then go to the Robot Behaviors menu
  - You have the list of all installed behaviors
  - You can launch them using the arrow button



SoftBank Robotics

Applications and Autonomous Life

Applications, Activities and Autonomous Life

- Goal: use the robot without a computer, make the robot context-aware
  - Launch applications by voice
  - Automatically start applications when the relevant context is met
- Goal: The robot must be alive!
  - Never stay completely still
  - React to the environment
  - Manage the robot's state: low battery...

Applications and Autonomous Life

Applications, Activities and Autonomous Life

- Autonomous Life is a module of NaoQi
- It offers life services and starts them if needed
  - Basic Awareness
  - Breathing
- It conditionally launches **solitary activities** if no one is there
  - Can be interrupted at any time
- It launches **interactive activities** when conditions are true
  - Only one interactive activity running at a time

Applications and Autonomous Life
Applications, Activities and Autonomous Life

Breathing

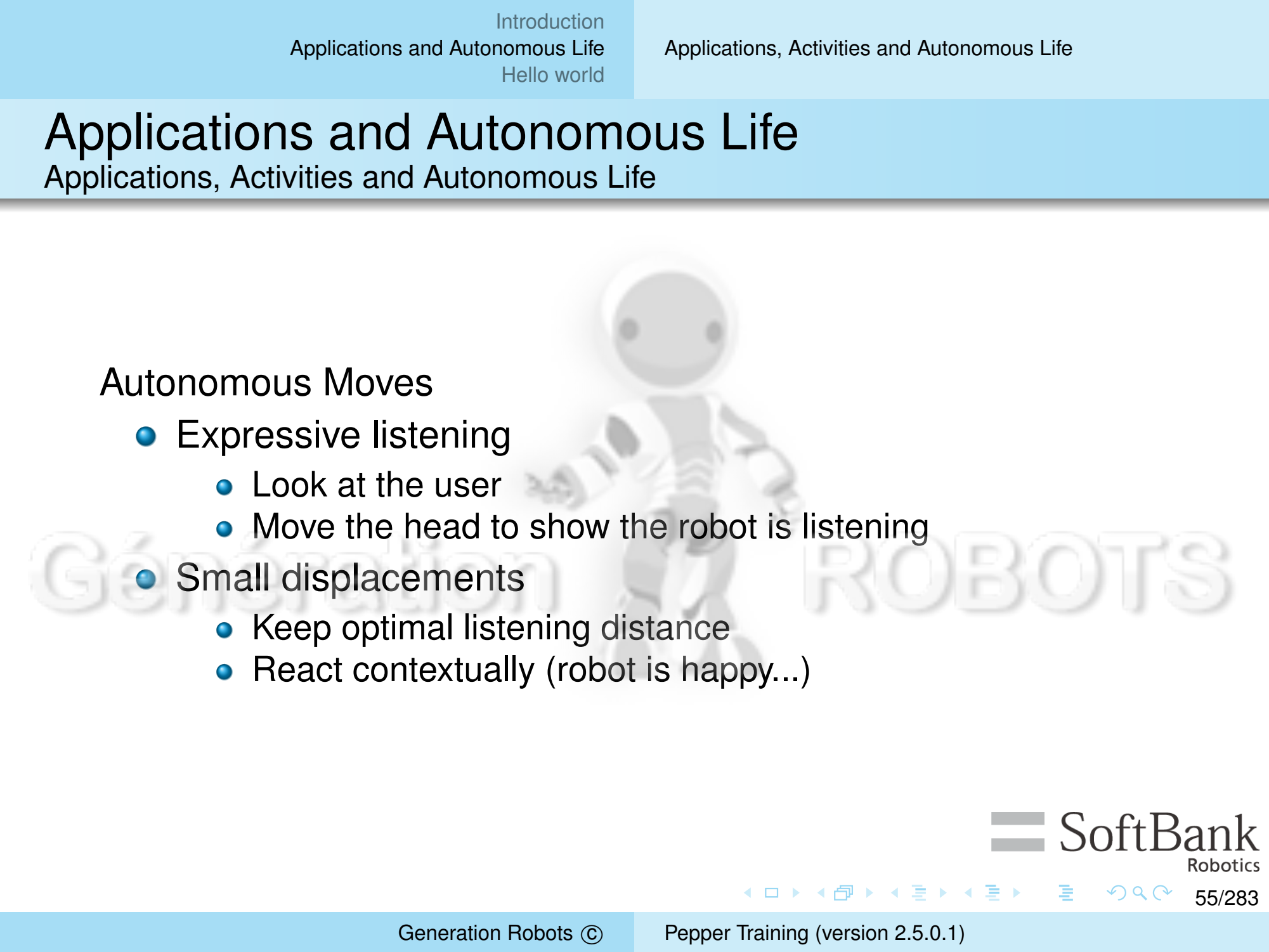
- Service that makes small breathing movements
- Designed to not interfere with ALMotion
- Assume that breathing is started when your application starts

Applications and Autonomous Life
Applications, Activities and Autonomous Life

Basic Awareness

- Module that keeps the robot aware of its environment
- Based on many other modules (ALPeoplePerception, ALMovementDetection...)
- Assume that Basic Awareness is started when your application starts

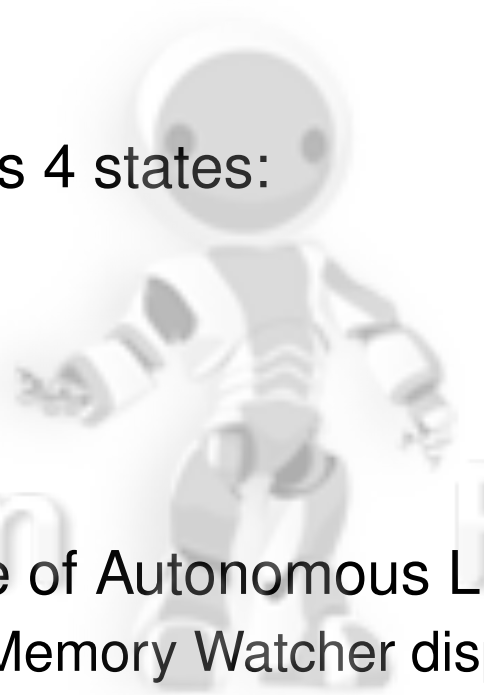
Applications and Autonomous Life
Applications, Activities and Autonomous Life



Autonomous Moves

- Expressive listening
  - Look at the user
  - Move the head to show the robot is listening
- Small displacements
  - Keep optimal listening distance
  - React contextually (robot is happy...)

Applications and Autonomous Life
Applications, Activities and Autonomous Life



- Autonomous Life has 4 states:
  - Disabled
  - Safeguard
  - Solitary
  - Interactive
- Display current state of Autonomous Life
  - in Choregraphe Memory Watcher display AutonomousLife/State memory event
  - $ qicli watch –almemory AutonomousLife/State

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Disabled state:

- Basic Awareness, Breathing stopped
- No activity can be started
- Can get out of this state only programatically

Applications and Autonomous Life
Applications, Activities and Autonomous Life

Safeguard state:

- Activated when critical reflex launched
  - Fall, people too close, battery low...
- Basic Awareness, Breathing stopped
- Current activitie(s) stopped
- Reflex is processed
- Depending on result, next state will be solitary or disabled

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Solitary state:

- The robot maintains a life-like behavior
- Automatically launches solitary activities
- The goal is to attract people to go into interactive state
- Can be interrupted at any moment

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Entering solitary state:

- Activated from code or when the interactive activity stack is finished
- Basic Awareness, Breathing started with parameters reset
- Autonomous Launchpad stopped
- Solitary or interactive activity can be launched
- Get out of this state, if interactive activity is started

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Interactive state:

- The robot interacts with a user
- The current activity is never stopped (only by critical reflex)
- Basic Awareness, Breathing keep the robot's moves natural

Applications and Autonomous Life

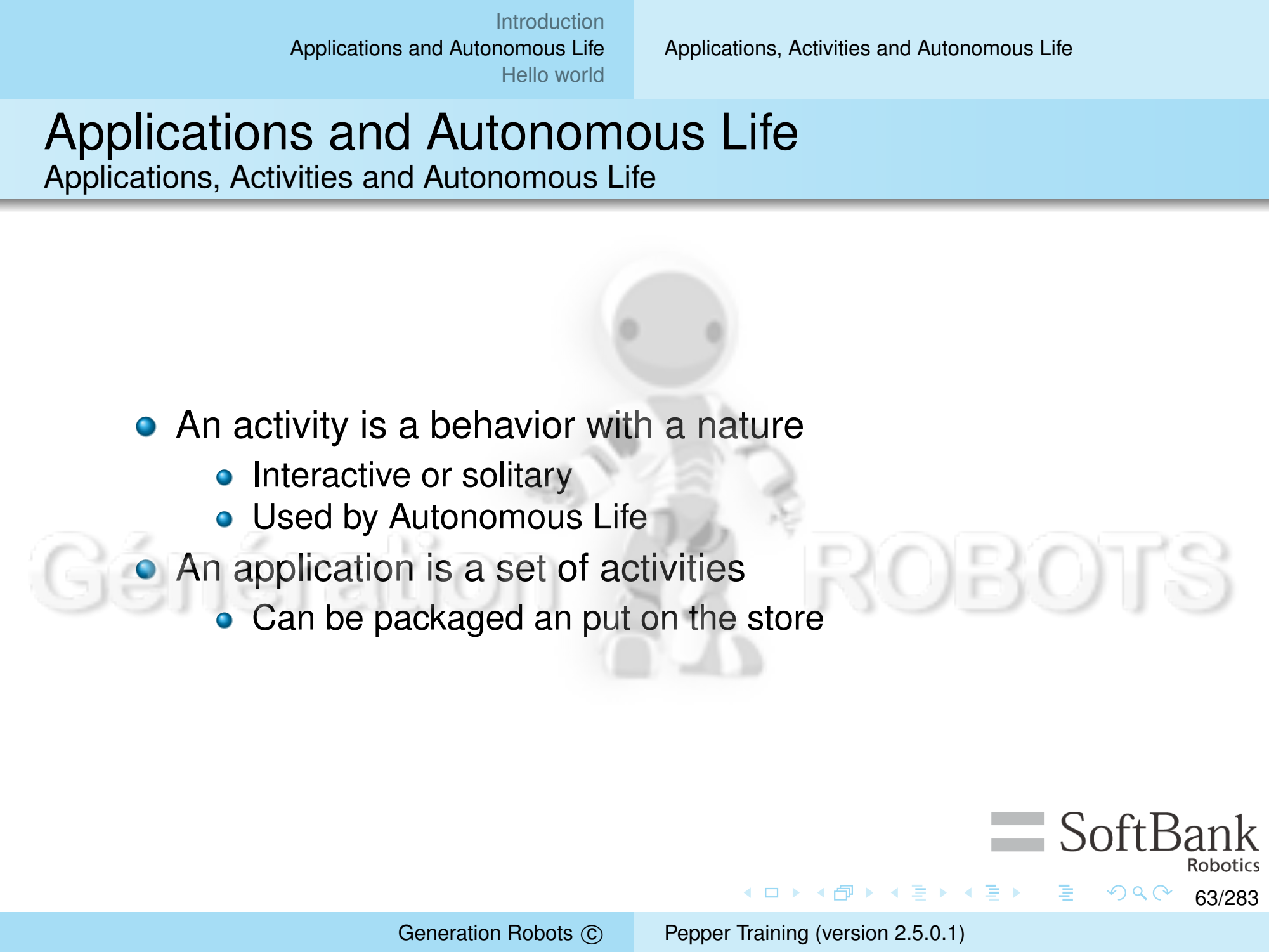
Applications, Activities and Autonomous Life

Entering interactive state:

- Activated from solitary state
- Basic Awareness, Breathing started with parameters reset
- Interactive activities can be launched with switchFocus()
- Get out of this state when all interactive activities are finished

Applications and Autonomous Life

Applications, Activities and Autonomous Life

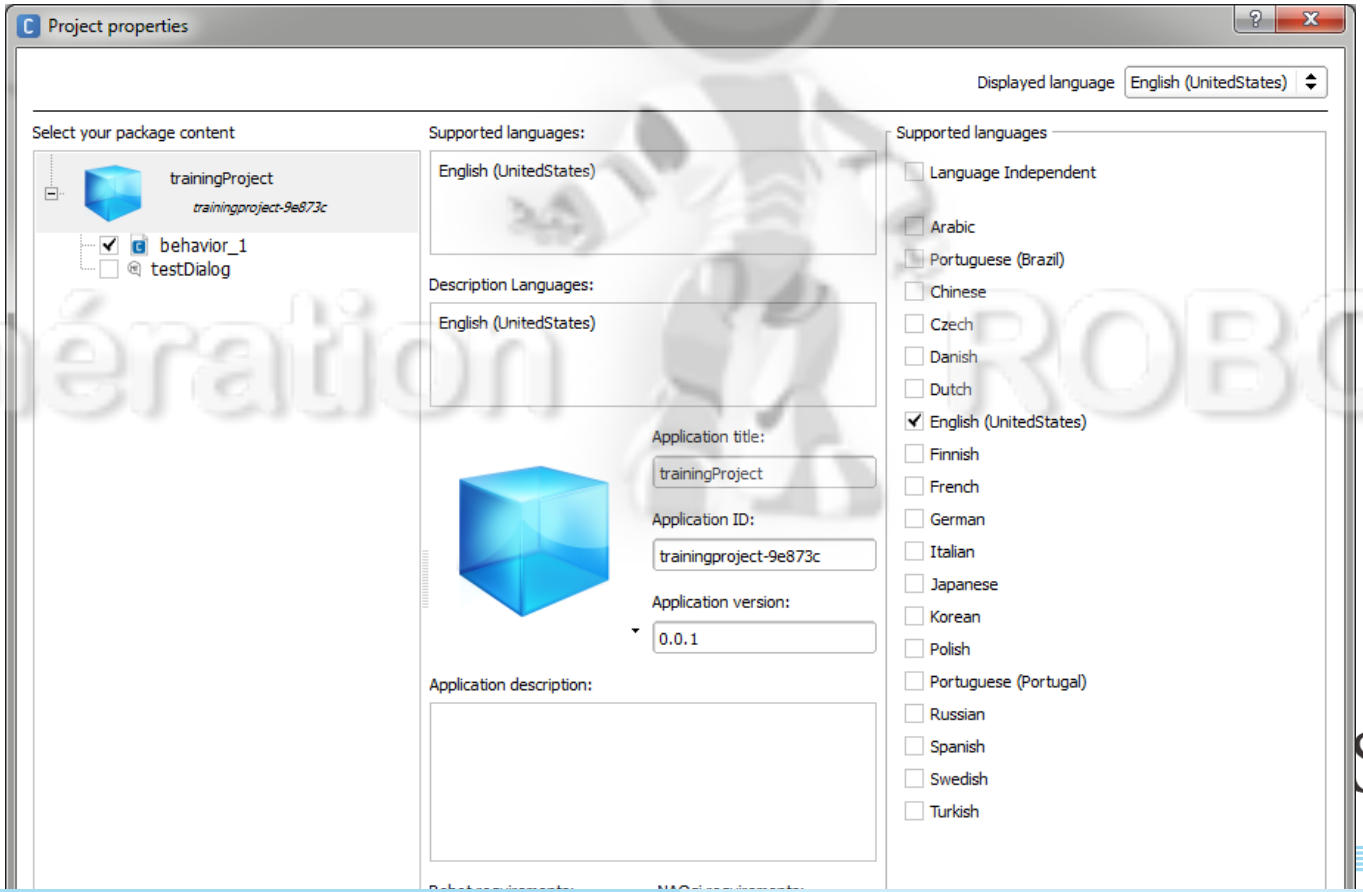


- An activity is a behavior with a nature
  - Interactive or solitary
  - Used by Autonomous Life
- An application is a set of activities
  - Can be packaged an put on the store

Applications and Autonomous Life

Applications, Activities and Autonomous Life

- In the project content menu of your Choregraphe project, click on properties:



SoftBank Robotics

Applications and Autonomous Life
Applications, Activities and Autonomous Life

- In the right part, selects the languages your application is compatible with
  - Then don't forget to fill all fields in all languages
  - For behavior without speech (dances), select 'language independent'
  - Otherwise, English is mandatory
- Select an application title
- Enter a description
  - It should describe clearly what the robot does in the behavior

Applications and Autonomous Life

Applications, Activities and Autonomous Life

- The Application ID has been set automatically
  - Allowed characters: lower case letters, numbers, - and _
  - It should be unique on the store !
- The version number starts at 0.0.1
  - You can change major and minor version number
  - Micro version number will be increased each time you put the application on the store

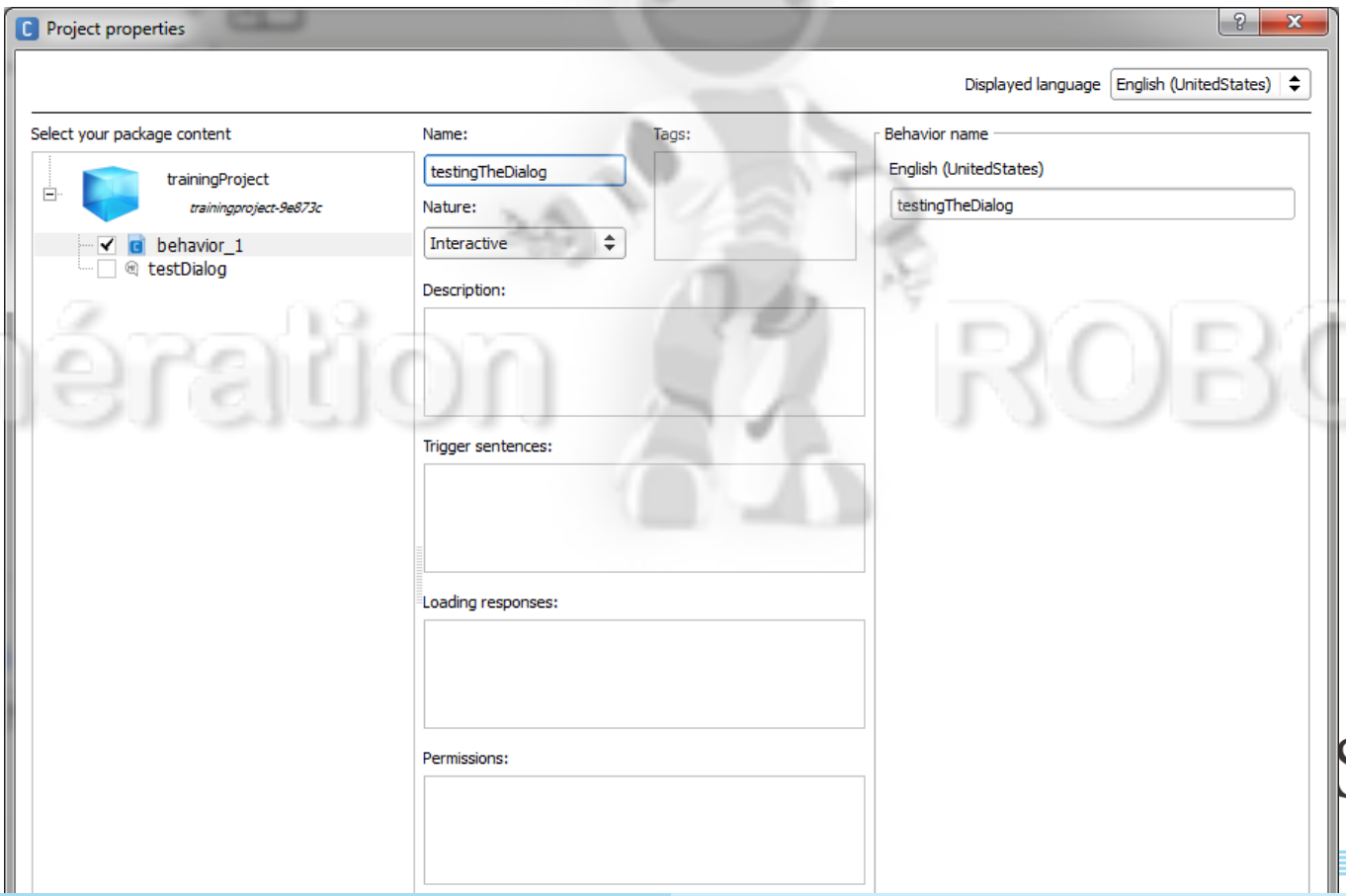
Applications and Autonomous Life

Applications, Activities and Autonomous Life

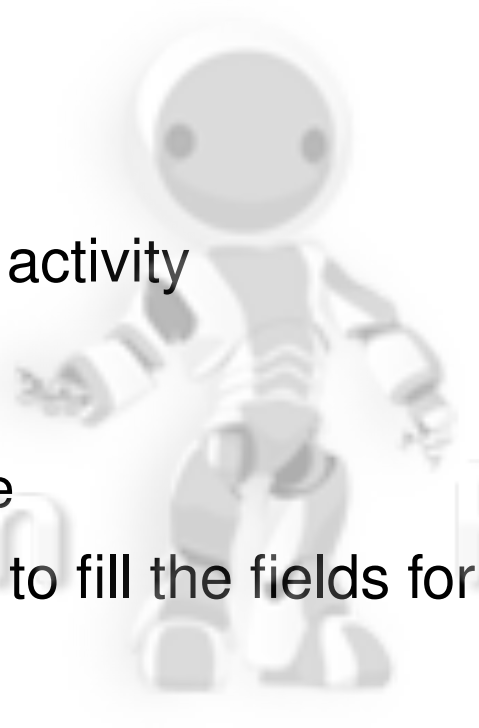
- The blue cube is the default application icon
  - Click on it to change it
- Click on Robot Requirement to restrict your application to some robots
- Click on the NAOqi requirement to restrict your application to some software versions

Applications and Autonomous Life
Applications, Activities and Autonomous Life

- Then you can click on each behavior in the project's contents



Applications and Autonomous Life
Applications, Activities and Autonomous Life



- Set a name for your activity
- Set some tags
  - One word by tag
  - Used by the store
- Again, you will have to fill the fields for all compatible languages

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Select a nature

- No nature
  - This activity will not be called by Autonomous Life
- Solitary
  - Put some life to the robot
  - Attract people to a interactive activity
  - Can be interrupted at any time
- Interactive
  - Won't be interrupted
  - Should be stoppable by user (or if user is gone)

Applications and Autonomous Life

Applications, Activities and Autonomous Life

- Put a description to your activity
- Permissions
  - May the robot stand up / sit down during the behavior ?
  - Can the behavior be run while the robot is on its charging station ?

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Used by Autonomous Life if the robot is in solitary mode:

- Trigger sentences
  - If the human says this sentence, the robot should launch this activity
  - Only if the application dialog_applauncher is installed on the robot
- Loading response
  - Sentence said by the robot to confirm that it launches the activity

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Used by Autonomous Life if the robot is in solitary mode:

- Launch trigger condition
  - Condition based on ALMemory values and time
  - Used to trigger the activity without voice
  - Are evaluated by the Autonomous Launchpad plugin

Applications and Autonomous Life
Applications, Activities and Autonomous Life

Condition example:

- At least 1 person in zone 3 for at least 5 seconds

```
('Launchpad/NumPeopleZone3' >= 1) ~ 5
```

- People are in the closest zone for at least 4 seconds and there is motion seen in that zone, all within the last 5 seconds.

```
((('Launchpad/NumPeopleZone1' >= 1) ~ 4) && ('Launchpad/
    NumMotionZone1' >= 1)) @ 5
```

Applications and Autonomous Life
Applications, Activities and Autonomous Life

In conditions you can use:

- constant floats (1, 1.5, 1e4)
- constant strings, double-quoted ("foo", "bar")
- keys: unquoted or quoted if having / (key12, 'motion/foo')
  - Key type can be event, data or microevent.
- Lists: [1, 2, foo]

Applications and Autonomous Life

Applications, Activities and Autonomous Life

In conditions you can use:

- Operators: + - / * && || == <= < >= > ! != []
- Function calls: substr, strlen, type, size, concat.
- ~: Requires condition to be true for given duration in seconds
- @: Requires condition to have been true at a specific point in the past
- #: Requires an expression to be true at any moment in time

Applications and Autonomous Life
Applications, Activities and Autonomous Life

In conditions you can use keys about:

- Number of people nearby
- Motion nearby
- Waving detection
- Battery status, network, pose, temperature
- Time of the year, time of the day
- Time keys from Autonomous Life, time keys from applications
- QR codes

Applications and Autonomous Life

Applications, Activities and Autonomous Life

What is the store ?

`cloud.aldebaran-robotics.com`

- The store is a website of Aldebaran Robotics
- Allowing to store, share, sell, update applications
- Manages versioning, beta versions
- Each user can have one or several robots linked to its account
- Previously named Application Delivery Engine (ADE)

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Link a robot to your account

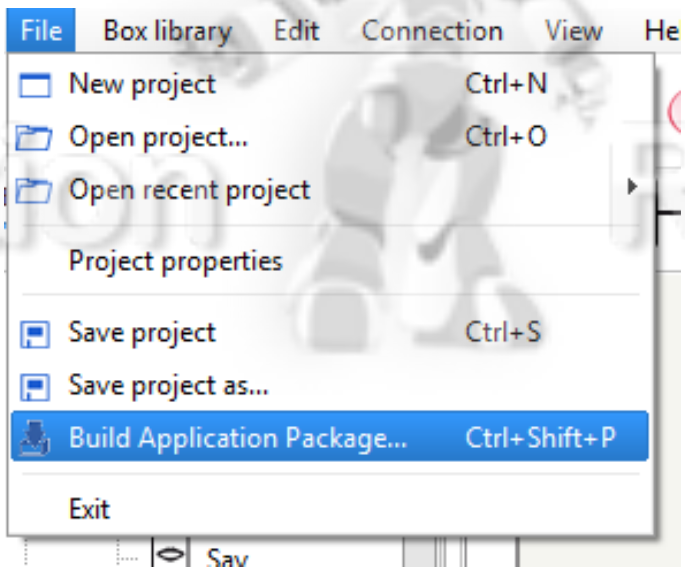
- Go to the robot's website
- In the web service feature, enter your login and password
- From the store, you can install applications on your robot
- The robot automatically updates its current applications

Applications and Autonomous Life

Applications, Activities and Autonomous Life

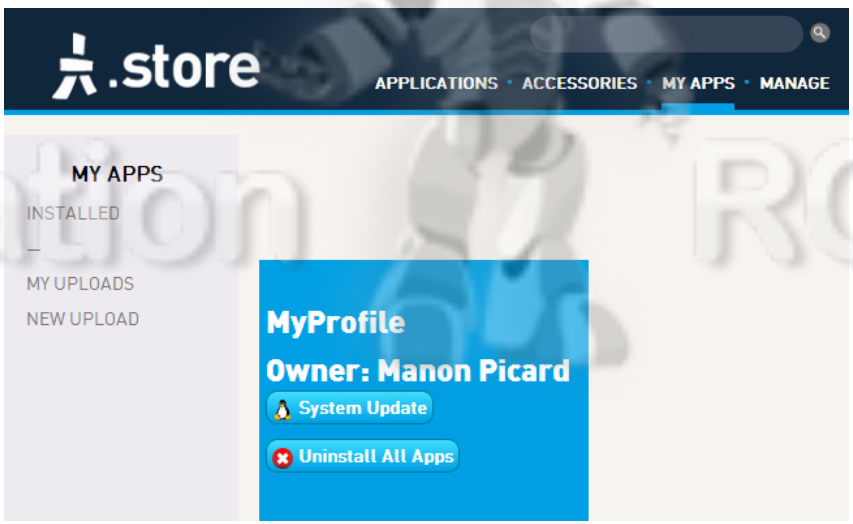
To put you application on the cloud:

- Package it (file -> Build Application Package)
- This will create a .pkg file



Applications and Autonomous Life
Applications, Activities and Autonomous Life

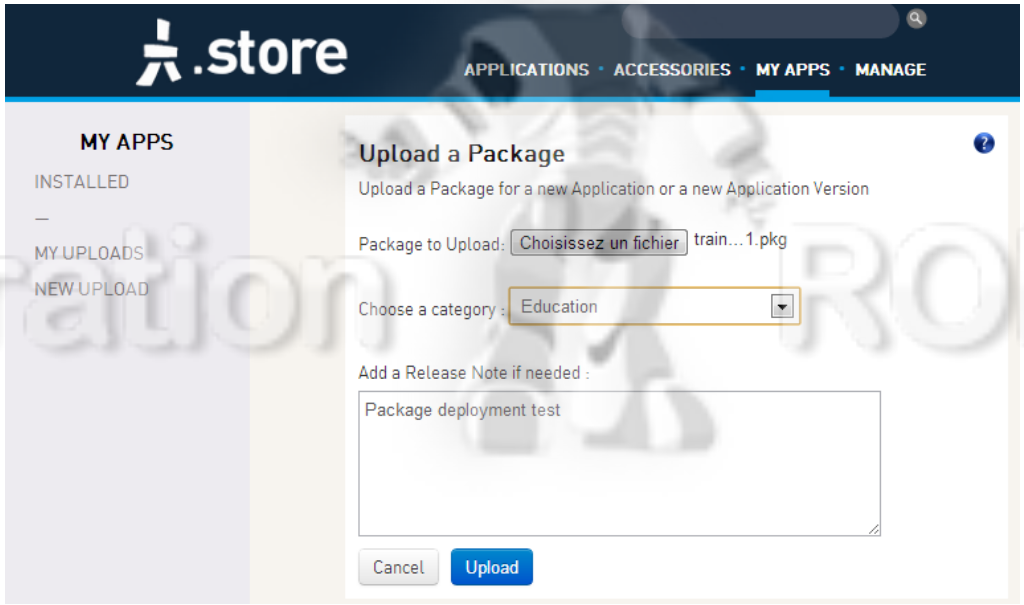
- Now go to the store (cloud.aldebaran-robotics.com)
- Go to 'my Apps' then 'new upload'



SoftBank Robotics

Applications and Autonomous Life
Applications, Activities and Autonomous Life

- Select your .pkg file and upload it

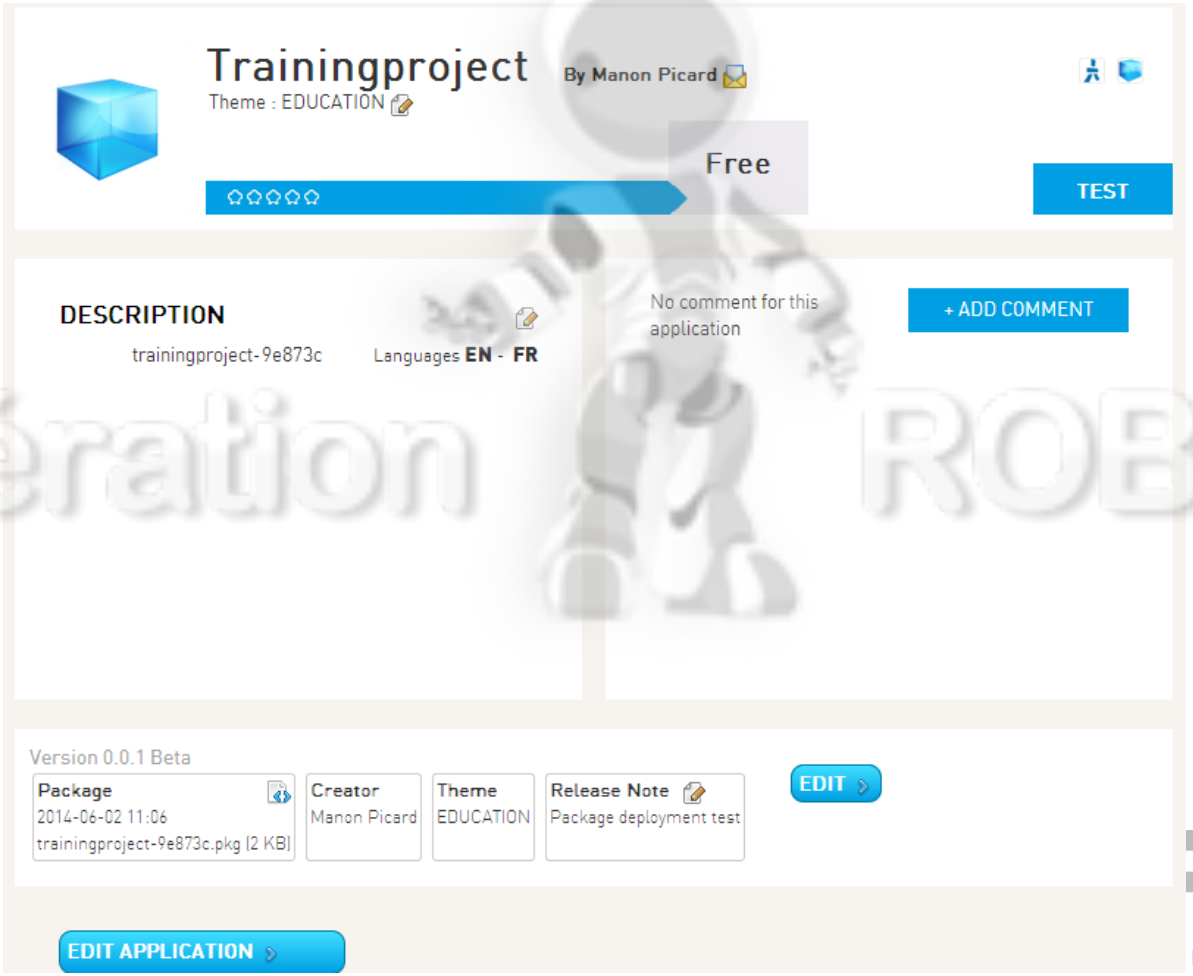


SoftBank Robotics

Applications and Autonomous Life

Applications, Activities and Autonomous Life

Your application is now available for download and test



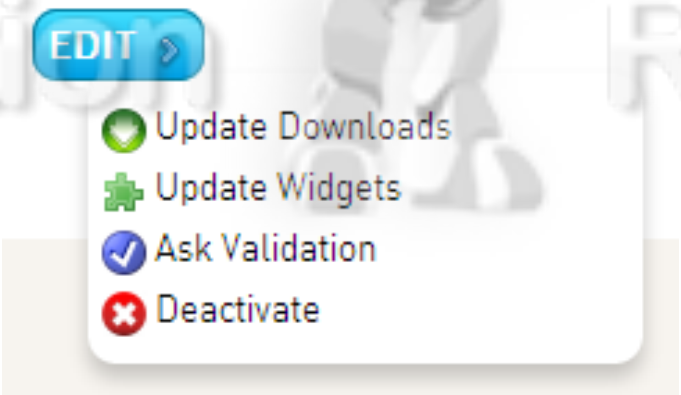
SoftBank Robotics

Applications and Autonomous Life

Applications, Activities and Autonomous Life

For the current version, you can:

- Update the downloads
- Update the widgets (not ready yet)
- Ask for validation

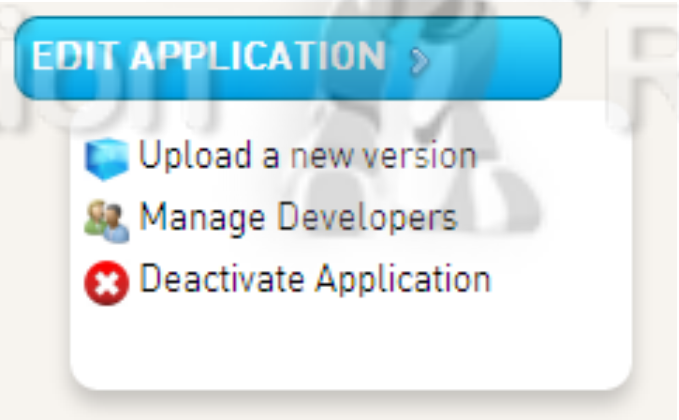


Applications and Autonomous Life

Applications, Activities and Autonomous Life

For the application, you can:

- Upload a new version
- Add developers to your team
- Deactivate the application



Applications and Autonomous Life
Applications, Activities and Autonomous Life

- Validation process:
  - You ask for validation
  - Validation team (in Paris) test and check some requirements (safety...)
  - You get a message saying if your application is validated or not and why.

Applications and Autonomous Life

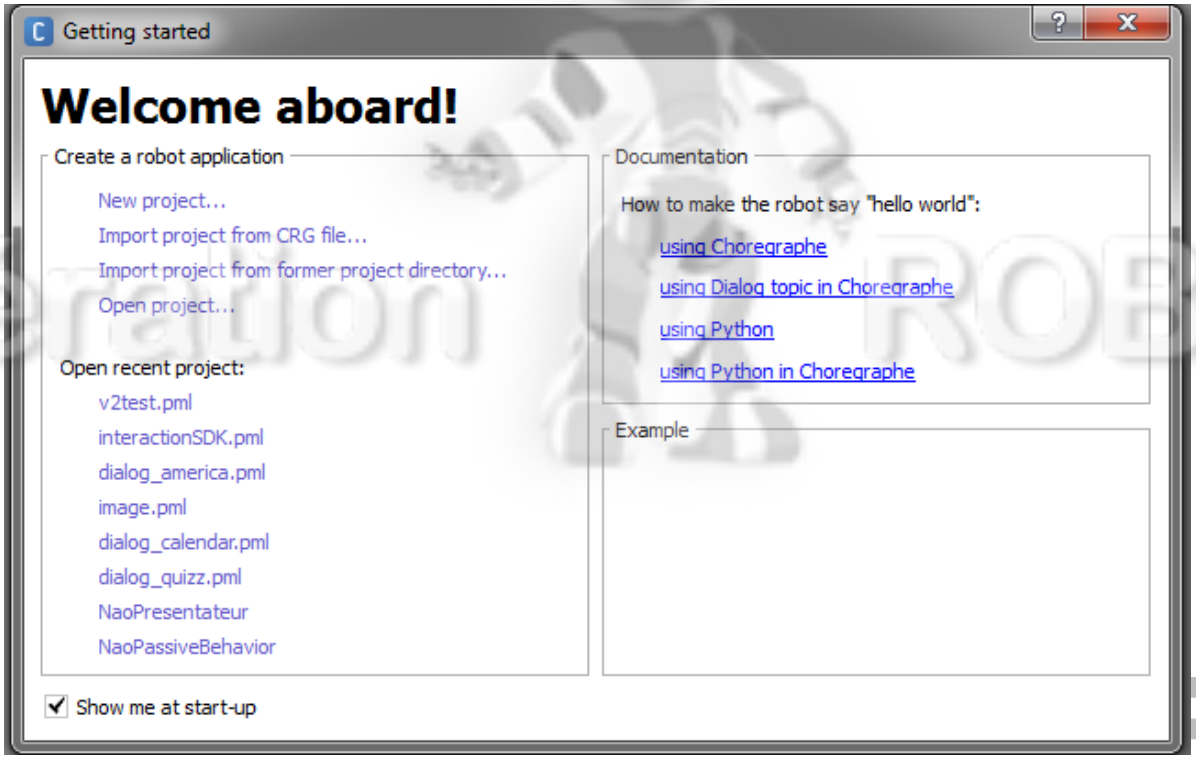
Summary

- Application / Behavior / Activity
- What is Autonomous Life
- States of the robot : disabled, safeguard, solitary, interactive
- Nature of activities : solitary, interactive, no nature
- Create a project (parameters, trigger sentences or events)

Outline

Hello world
Create your first behavior

- Open Choregraphe
- Select new project

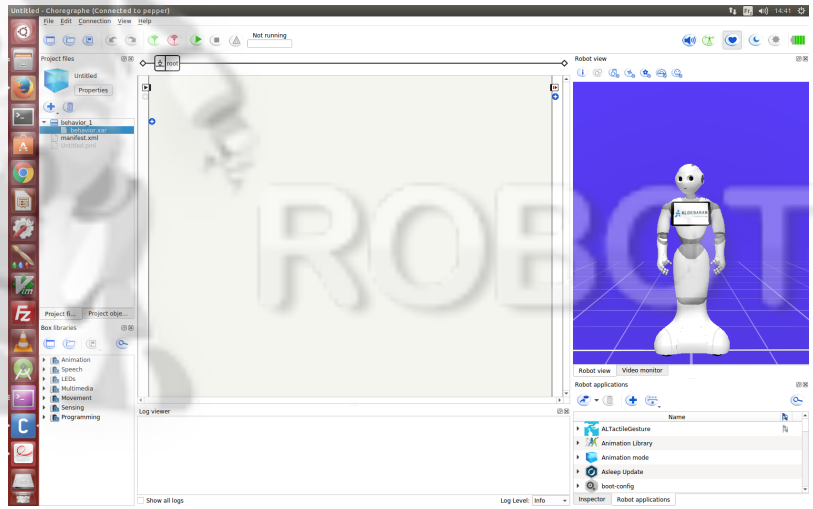


SoftBank Robotics

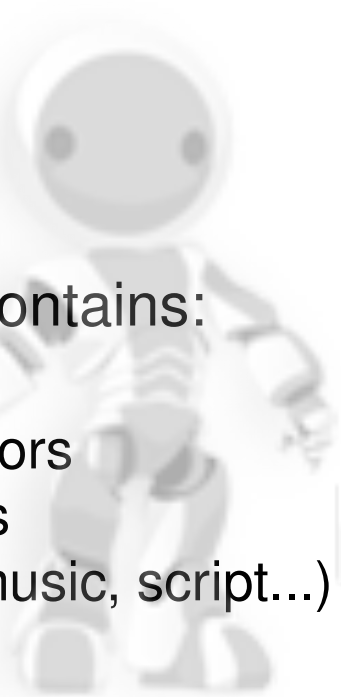
Hello world
Create your first behavior

Choregraphe's main panels:

- Project content
  - List every behavior in your project
- Default box library
  - Ready-to-use boxes
- Flow diagram
- 3D robot view



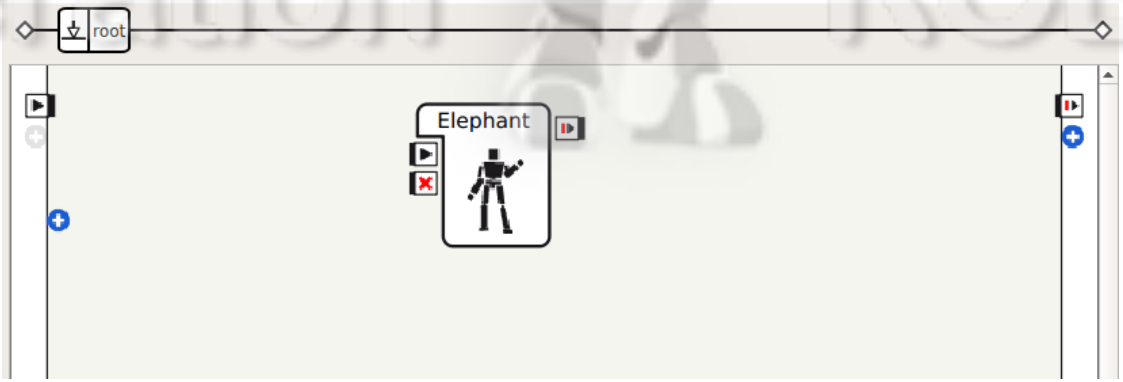
Hello world
Create your first behavior



- A Choregraphe project contains:
  - A manifest
  - Any number of behaviors
  - Any number of dialogs
  - Resource files (text, music, script...)

Hello world
Create your first behavior

- In the box library, find the 'Elephant' box
  - In the motion/animation folder
- Drag and drop it to the flow diagram
  - The box in your flow diagram is a copy from the one in the library
  - It can be moved, copied, pasted...



SoftBank Robotics

Hello world

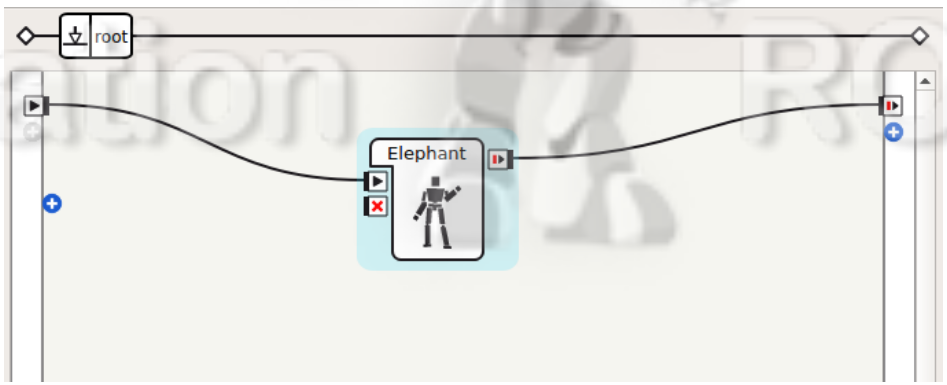
Create your first behavior

- The behavior activates the  input when it starts
- It stops when the  output is activated
- The box has inputs on the left and outputs on the right
  - The  (onStart) input starts the box
  - The  (onStop) input stops the box
  - The  (onStopped) output is activated when the box stops
- The + buttons allow to add inputs and outputs

Hello world

Create your first behavior

- Link the 'Elephant' box to the input and the output of the behavior
  - Click on the behavior input and drag to the box input
  - Same for outputs



SoftBank Robotics

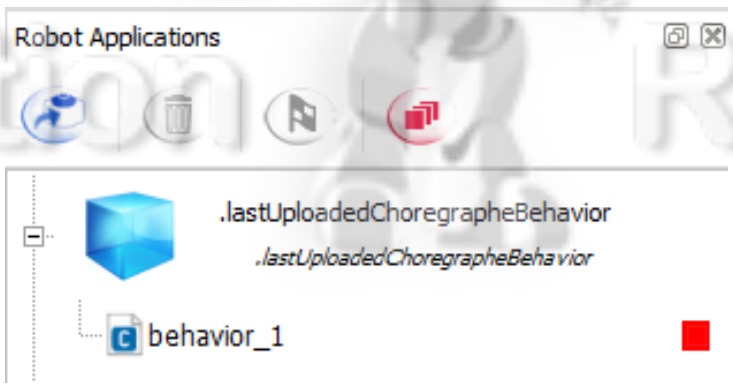
Hello world
Create your first behavior

- A behavior can be run without a robot (Choregraphe should be connected to a local NaoQi)
- Run the behavior by clicking the  button
  - You can see the robot view move
  - The links become green when activated
- You can stop the behavior by clicking the  button

Hello world
Create your first behavior

When launching a behavior with Choregraphe:

- Choregraphe installs your behavior as an application
  - Stored on the robot
  - Named .lastUploadedChoregrapheBehavior



Hello world
Create your first behavior

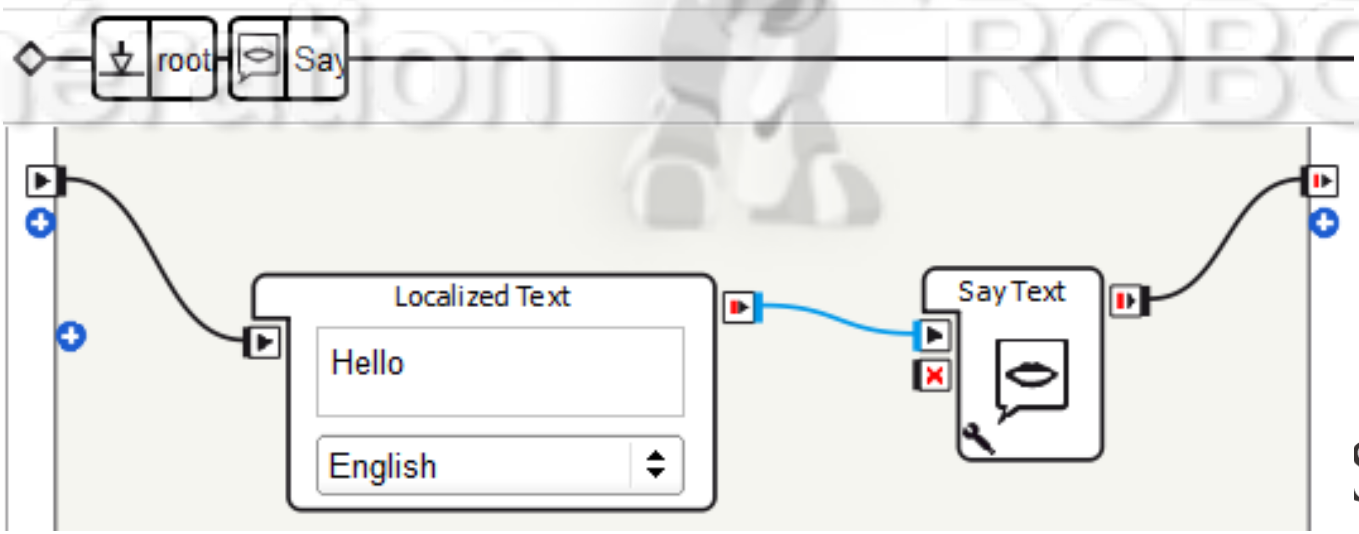
When launching a behavior with Choregraphe:

- Then Choregraphe starts the application
- The robot sends information back to Choregraphe
  - To get links green
  - Timing on Choregraphe may not be accurate

Hello world
Parallel behaviors

We want the robot to speak while moving

- Use the 'Say' box (Audio/Voice)
- Double-click on it: you enter the box
- You can see where you are in the hierarchy with the 'Box Path' line
  - Click on 'Root' to come back to first level



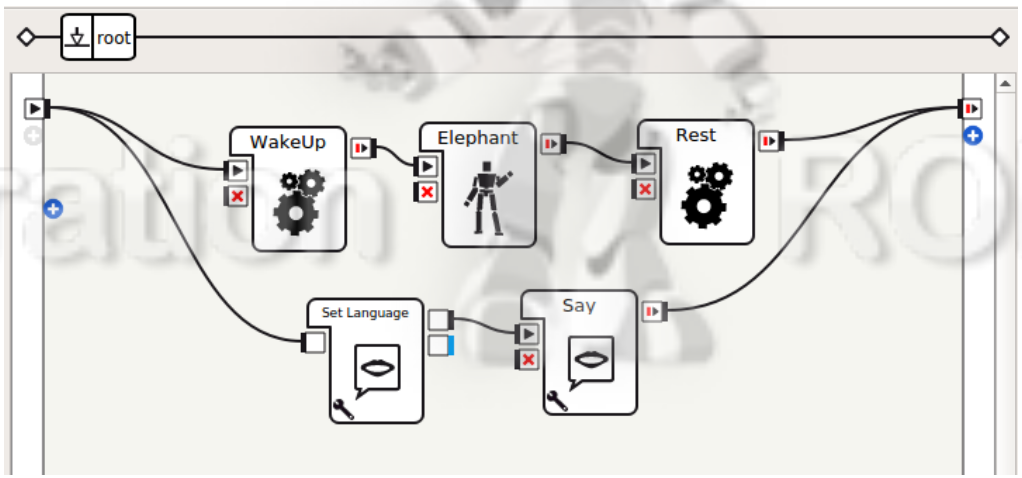
SoftBank Robotics

Hello world
Parallel behaviors

- Select your language and enter your text
  - Warning: the 'Say' box does not change the robot's language
  - You can specify a sentence for each language (multi-language behavior)
- Click on the wrench to see parameters and modify the robot's voice

Hello world
Parallel behaviors

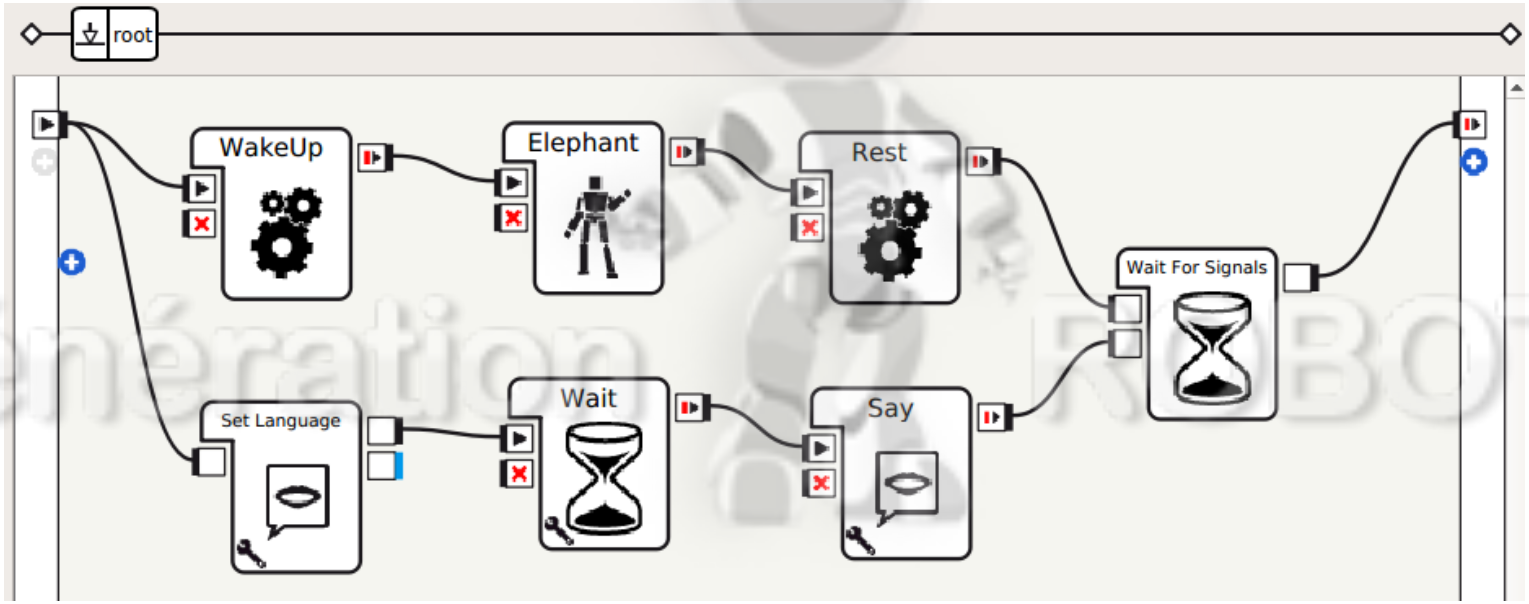
- To change the language of the robot, use a 'set Language' box



Hello world
Parallel behaviors

- The behavior stops before the end of the move
- Add a 'Wait for signals' box (Flow Control)
  - The output fires when the two inputs have been stimulated
- You can also add a 'Wait' box (Flow Control/Time) before speaking
  - Set the time in parameter
  - Warning: don't use the 'Timer' box that fires every n seconds

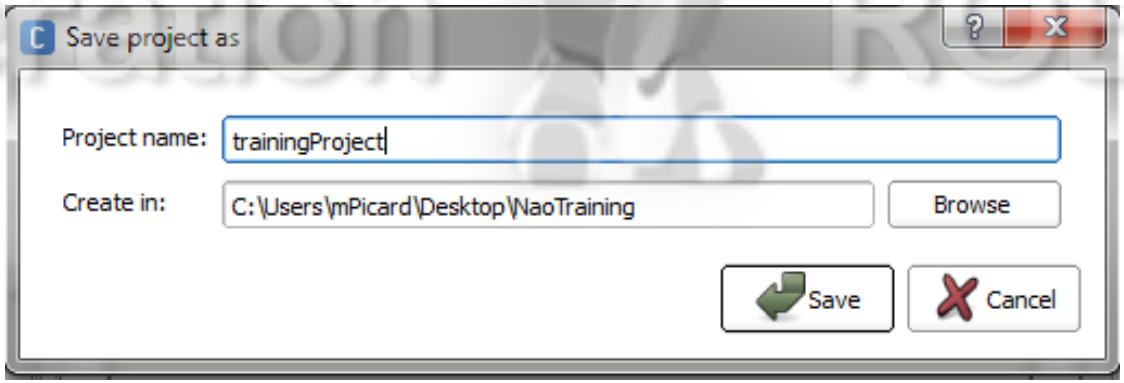
Hello world
Parallel behaviors



SoftBank Robotics

Hello world
Parallel behaviors

- Save your project by clicking on the  button
- Select a path and a name
  - A folder with the name will be created
  - Warning, when you save again, the folder contents will be erased !



SoftBank Robotics

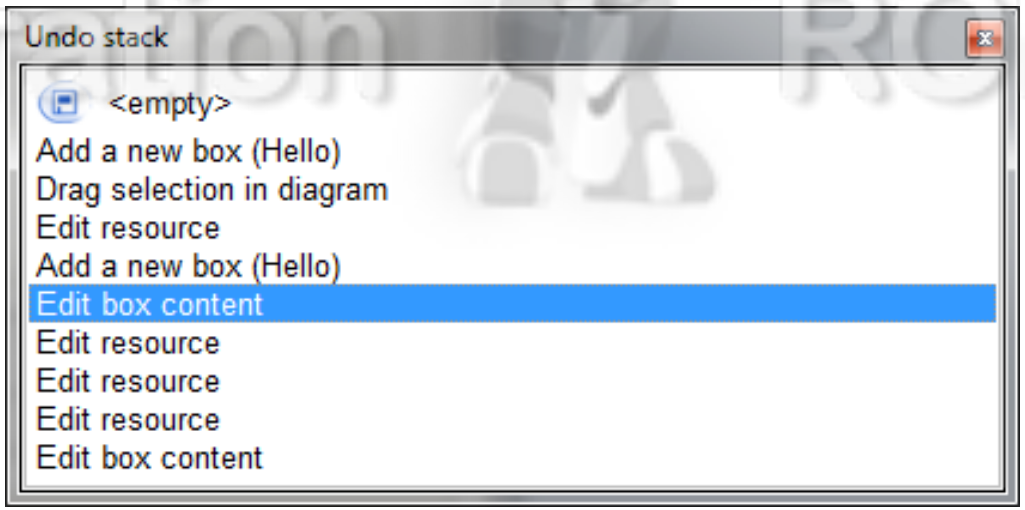
Hello world
Parallel behaviors

- In your project folder, you will find:
  - A PML file describing the contents
  - Behavior folders: containing a .xar file
  - Dialog folders : containing a .dlg file

Hello world
Useful features

Undo stack:

- In the 'view' tab, select 'undo Stack'
- This panel shows the successive states of your project
- Click to go back to some state
- If you modify the project when in a 'previous' state, the 'future' states are lost



SoftBank Robotics

Hello world
Useful features

Simulate a robot:

- Sometimes, it is useful to work without a robot
  - To test programs
  - Because there are not enough robots
- To do that, you have to run a NaoQi

Hello world
Useful features

1. Choregraphe creates a local naoqi at launch
   - Destroyed if you connect a robot, can be re-created through menu 'connection > connect to virtual robot'
   - Cannot be accessed by any program outside Choregraphe (e.g. Monitor)
2. Manually launch naoqi
   - Go to `path-to-Choregraphe/bin` and launch naoqi-bin.exe
   - Connect with Choregraphe (the Nao icon is dotted or use IP 127.0.0.1 and port 9559)
   - Can be accessed by Monitor

Hello world
Useful features

For the two last methods, your simulated robot will:

- Have always stiffened joints
- Have no cameras
- Have no audition nor audio player functions
- Have no leds
- Have no tablet (for Pepper)

Hello world

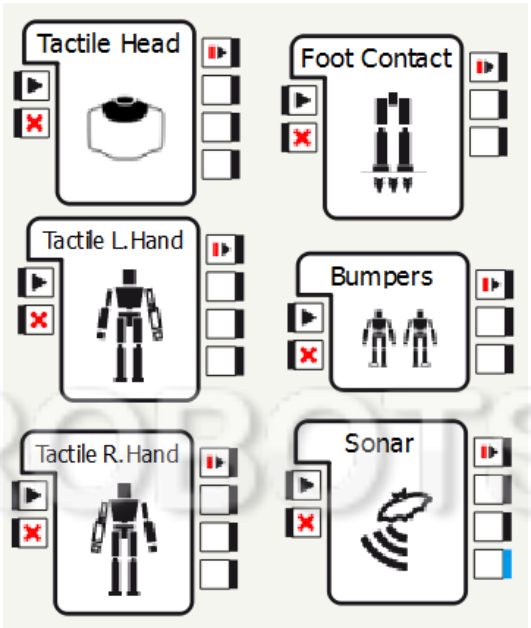
Notions summary / Key concepts

- Choregraphe GUI
- Diagram (root level) onStart, onStopped input/outputs
- Boxes inputs/outputs
- Structure of a project
- Application location on the robot (Play button F5 or Packaging and installation)
- Notion of parallel activities
- Some boxes (Elephant, Say, Wait, Wait for Signal, . . .)
- Robot simulation

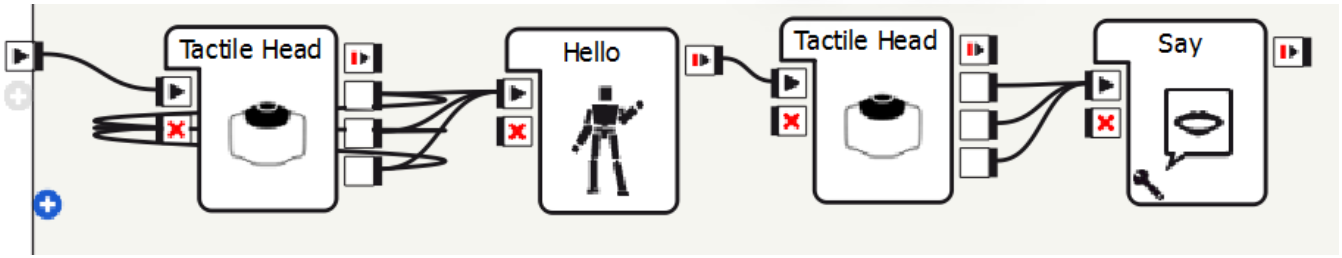
Outline

Sense the world and people emotions
Sensing boxes

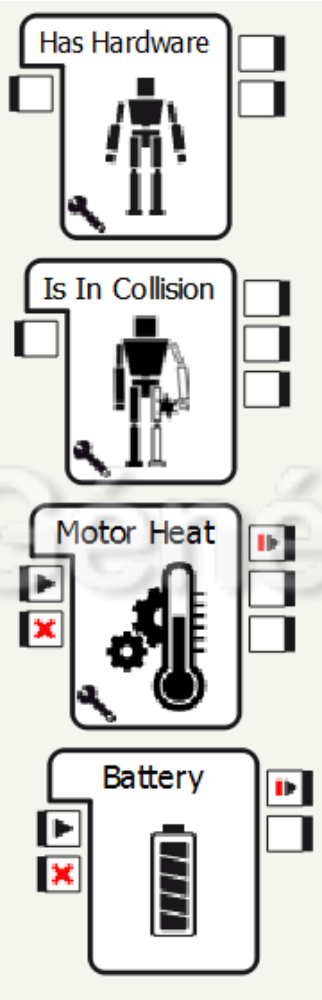
- Sensing boxes allow you to get events from sensors
- Will trigger events until stopped



- Careful if you have several times the same box



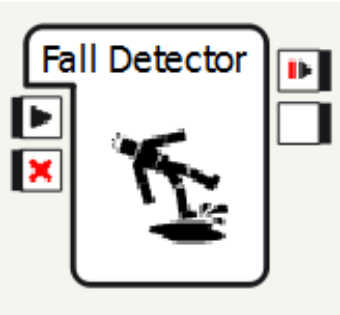
Sense the world and people emotions
Sensing boxes



Other sensing boxes:

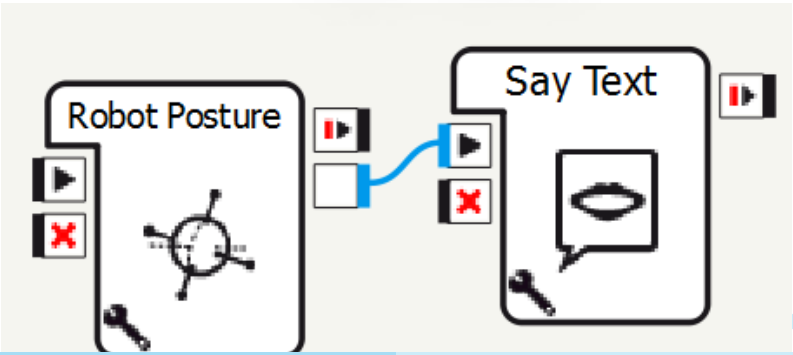
- Does the robot has legs, laser head... ?
- The robot knows if it's colliding or nearly colliding with itself.
- Get warned if the robot overheat. It won't hurt itself but can stop moving to cool down
- Is there enough battery left to do what you want to do ?

Sense the world and people emotions
Sensing boxes



- When the robot feels it's falling, it has reflexes: protect with arms, remove stiffness, say 'ouch'

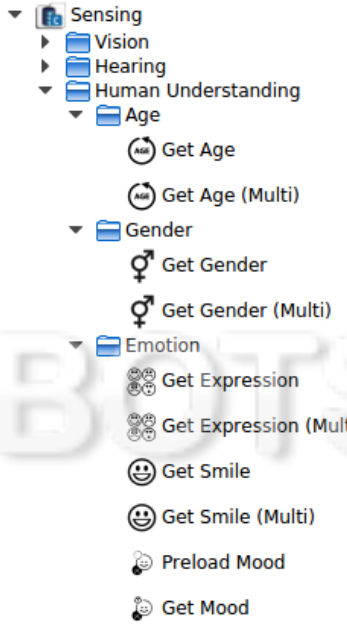
- The 'Robot posture' box outputs the position of the robot
- $ qicli call ALRobotPosture.getPostureList
  - For pepper : *Stand, StandInit, StandZero, Crouch*
  - For nao: *Standing, Sitting, LyingBack, LyingBelly, LyingLeft, LyingRignt, Back, Belly, Left, Right, UpSideDown, Lifted, Unknown*



Sense the world and people emotions
Sensing boxes

Emotion interaction is a great feature in Pepper

- Emotions boxes allow you to get human face information extracted on age, gender, smile, mood or voice emotion with a confidence level
- Mood : positive, neutral, negative, unknown
- Smile : smile degree from 0 to 1
- Expression : happy, surprised, neutral, sad, angry



Sense the world and people emotions
Object recognition

Object recognition is based on the extraction and matching of interest points

- This method is robust to size and orientation changes
- Interest points are corners, strong color changes...
- Objects have to present enough interest points (avoid white goblet, pen...)

Sense the world and people emotions
Object recognition

A database of objects has to be built beforehand

- The database is built in the computer
- It can be saved and completed afterward
- Don't forget to send the database to the robot

see ∼ /.local/share/naoqi/vision/visionrecognition/current

Sense the world and people emotions
Object recognition

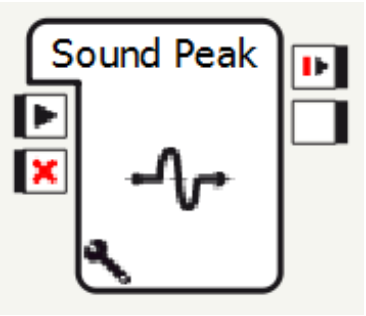
1. In the video monitor, click the  button
   - You have 4 seconds to position the object you want in front of the camera
2. On the monitor, click to outline the object

SoftBank Robotics

Sense the world and people emotions
Object recognition

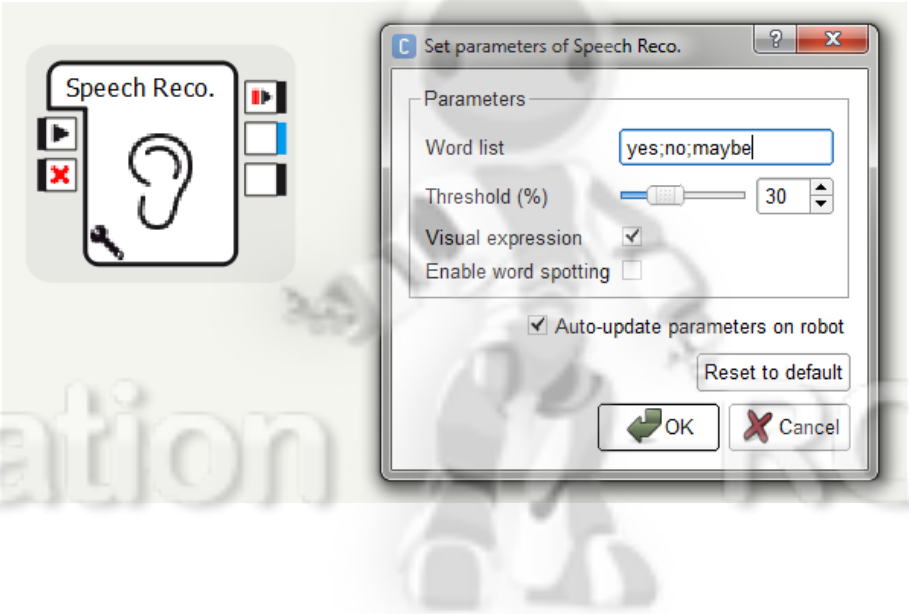
3. Name the object and send the recognition database to the robot
4. To detect an object, use the *vision reco.* box
   - The box returns the name and the orientation of the detected object

Sense the world and people emotions
Audio



- The 'Sound Peak' box detects loud noises
- The 'Sound Loc.' box is used in the 'Sound Tracker' box (in the tracker folder)

Sense the world and people emotions
Audio

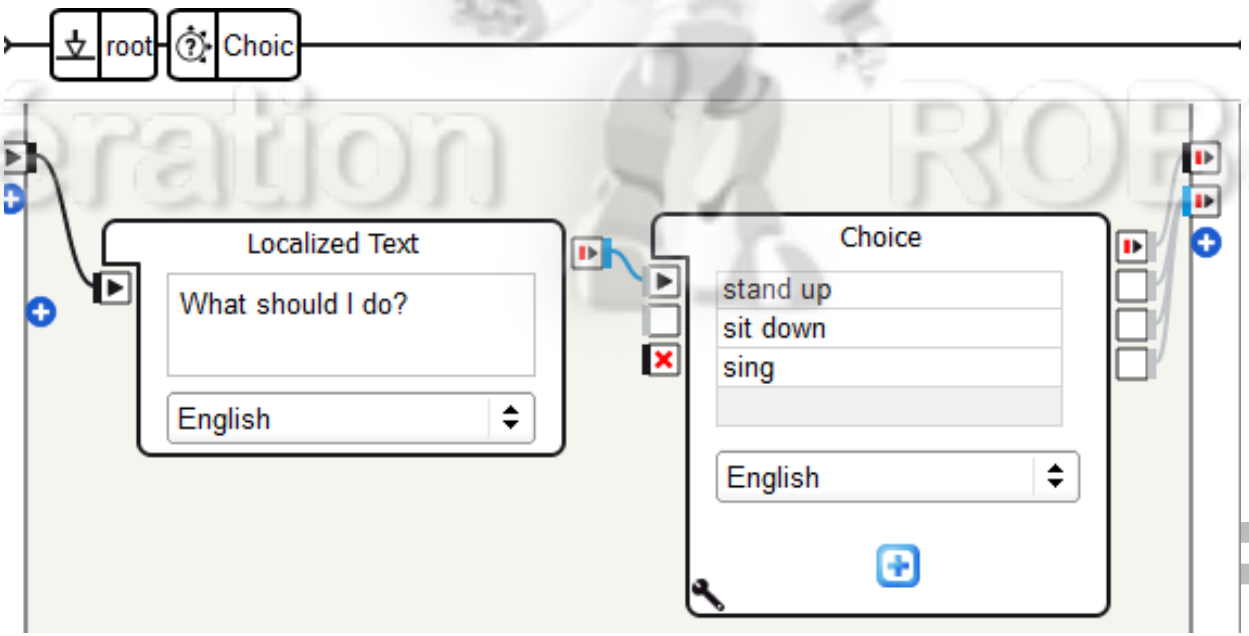


- The 'Speech Reco.' box recognizes words from a given list
- Speech recognition uses Nuance algorithms
  - Two language licenses by robot

Sense the world and people emotions
Audio

- Speech recognition is also used in the 'Choice' box
- The robot asks a questions and waits until it gets an answer
  - You can select the answer from the head sensors

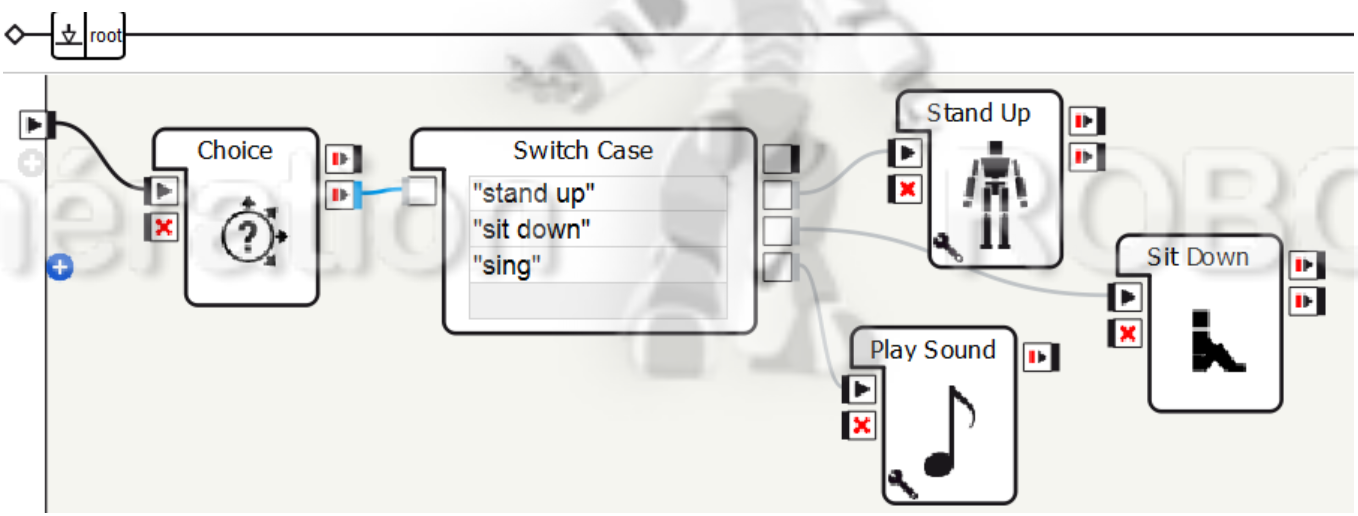
Example: an obedient robot



SoftBank Robotics

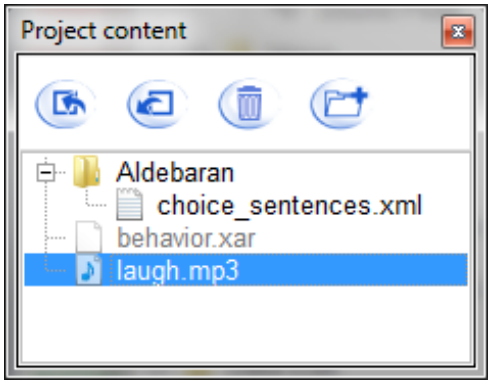
Sense the world and people emotions
Audio

- The 'Switch case' box fires an output if it corresponds to the input

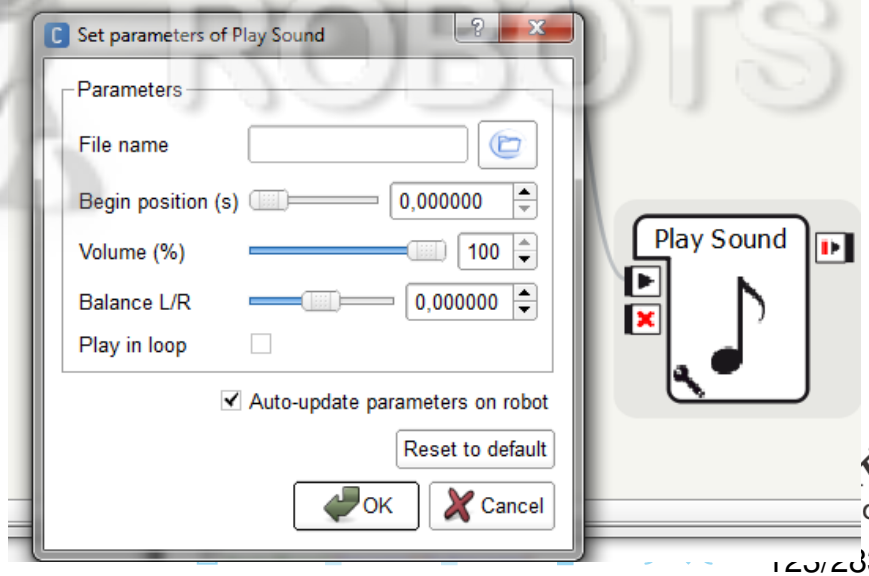


Sense the world and people emotions
Audio

- The 'Play sound' box plays wav, mp3 and ogg files
  - Add the file with the project content panel
  - The files are copied, re-upload them to modify them



- Use the parameters of the box to specify the file



Outline

Create and organize boxes
Learn and recognize face

- Before recognizing a face, you need to learn it using the 'Learn Face' box

Create and organize boxes
Learn and recognize face

"Type" of boxes inputs and outputs

- blue : string data
- yellow : numbers data
- black : stimulation input i.e. Bang input
- grey : dynamic data types i.e. Bang + data

Create and organize boxes
Learn and recognize face

- Run a behavior with only a 'Learn Face' box
  - Double-click the input to start the box and enter your name
  - Wait for the eyes to become blue, then green
- A face needs to be learned only once
- A learned face is remembered even after the robot's shutdown

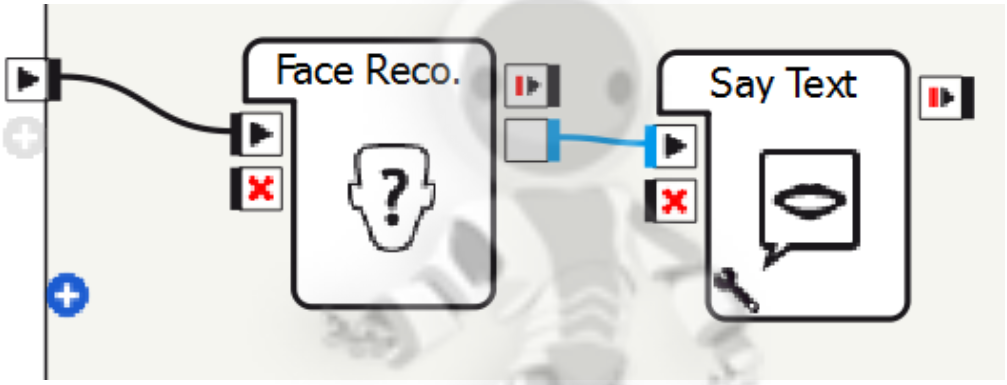
Faces db is located in ~ /.local/share/vision/facerecognition

Create and organize boxes
Learn and recognize face

- In a new project, use the 'Face Reco' box
  - Once started, it analyzes all camera images and outputs the name of the recognized person
  - Use the onStop () input to stop the box
- Take a 'Say' box and open it
- Copy the 'Say text' box from inside the 'Say' box
  - This box makes the robot pronounce a text
  - Link it to the 'Face Reco' box so that the robot tells your name when it sees you

SoftBank Robotics

Create and organize boxes
Learn and recognize face



- Face recognition is sensitive to head orientation
- Careful of backlight
- Use the Video Monitor to check you are in the camera field of view
- or use 'Select camera' box

SoftBank Robotics

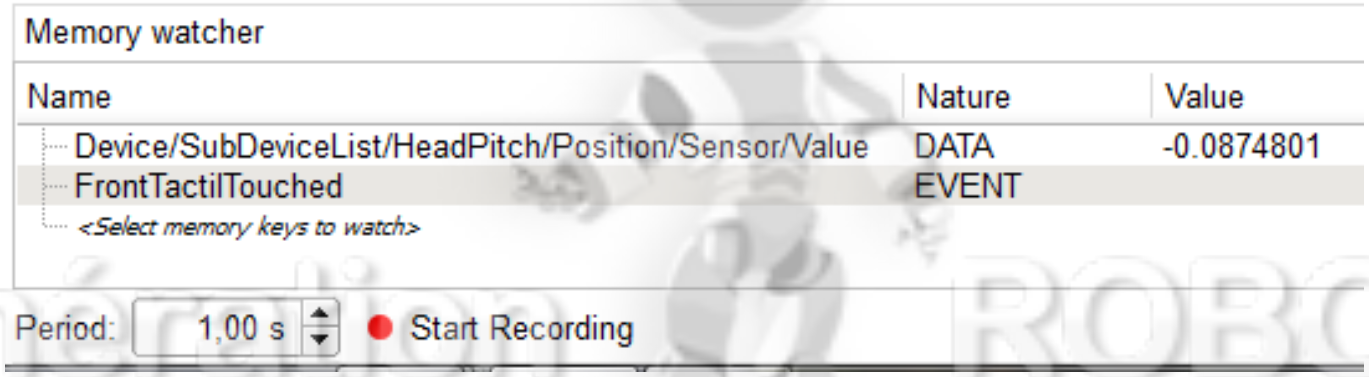
Create and organize boxes
Memory boxes

ALMemory

- Naoqi has a shared memory space
  - Accessible by any behavior, any script
- Stores data under a key
  - Stored data can be boolean, integer, float, list, string or object
- Can manage events

Create and organize boxes
Memory boxes

You can look at the content of ALMemory with the Memory Watcher



You can also look memory content with qicli

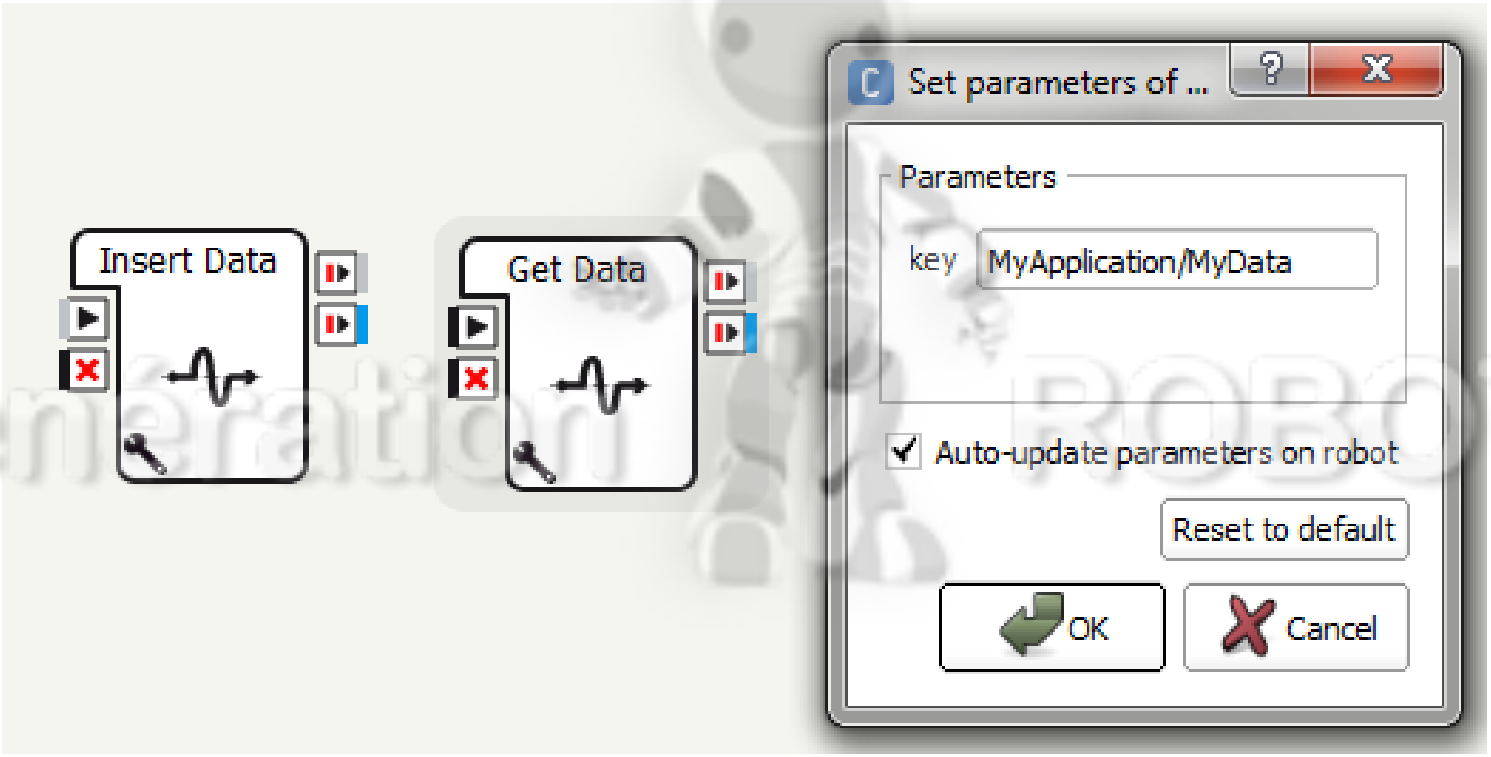
- $ qicli watch --almemory FrontTactilTouched
- $ qicli watch --almemory EngagementZones/PersonEnteredZone1

Create and organize boxes
Memory boxes

In the advanced box library, you have the memory boxes

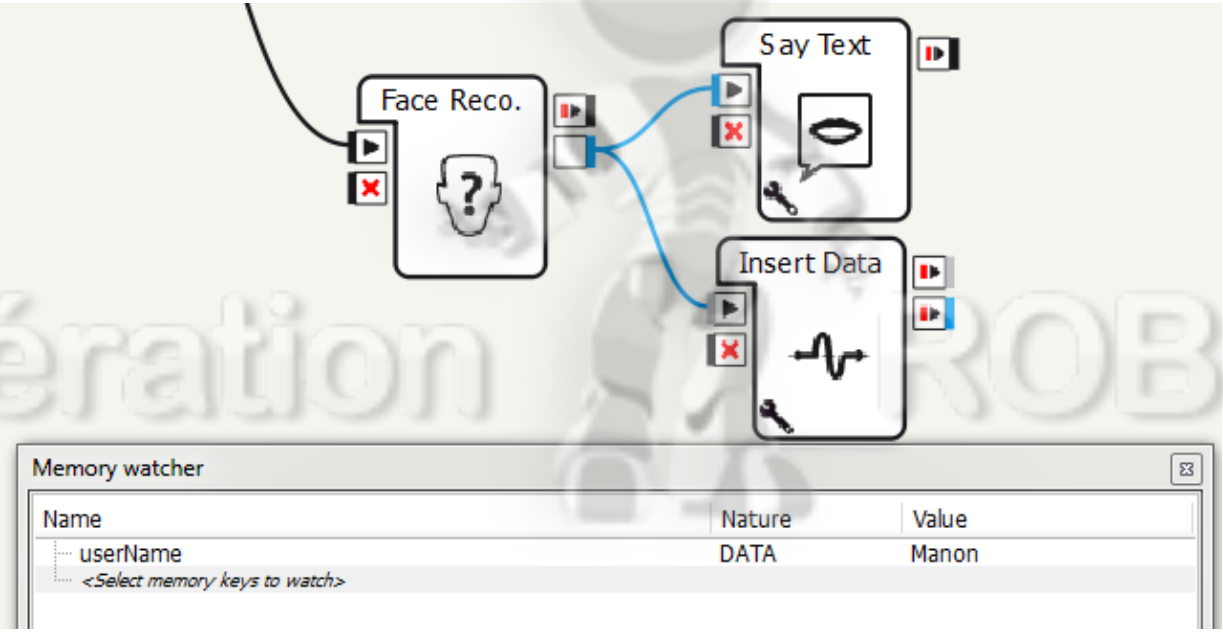
- insertData to put something in ALMemory
  - If the key don't exist, the entry is created in ALMemory
  - If the key exist, the entry is replaced
- getData to read something from the memory

Create and organize boxes
Memory boxes



Create and organize boxes
Memory boxes

You can save the name of the people seen in ALMemory

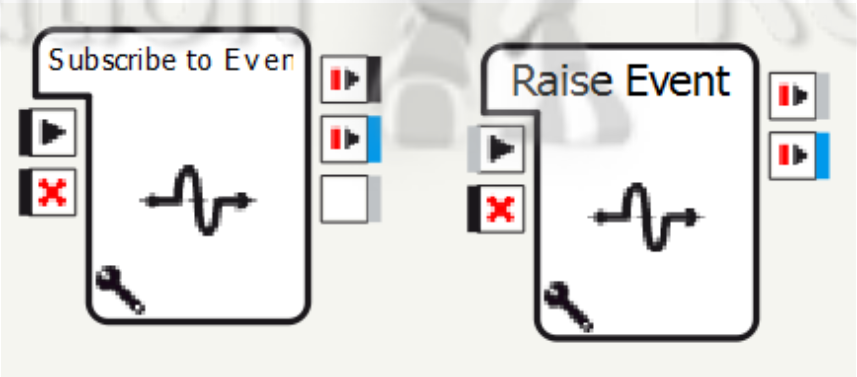


The value stays in the memory until Naoqi is stopped

Create and organize boxes
Memory boxes

Memory events

- You could replace the insertData box by a raiseEvent
  - This will insert the value in ALMemory
  - But also create an event
- The event will be catched by any 'subscribe to event' box listening to this event



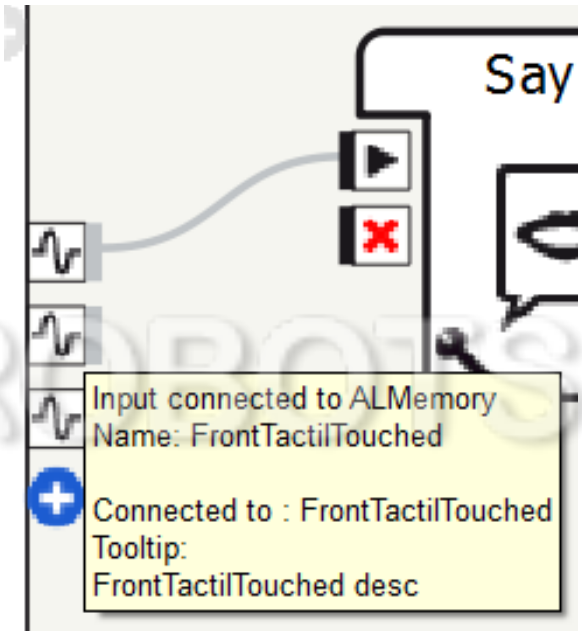
Create and organize boxes
Memory boxes

- A 'subscribe to event' box is equivalent to a ALMemory input  :
  - Linked to a key stored in the robot's memory
  - An event is triggered each time the value changes

Create and organize boxes
Memory boxes

ALMemory input:

- Click on the + button on the left of the flow diagram
- Select some key
  - For example 'FrontTactilTouched', 'MiddleTactilTouched' and 'RearTactilTouched'
- Run the behavior and touch the head of the robot to see triggered events



Create and organize boxes

Summary

- Memory
- Box parameters
- Sensors
- Object recognition and object database
- Face learning and recognition
- More boxes (Tactile Head, Bumpers, Vision Recognition, Sound boxes, Speech reco, Choice...)

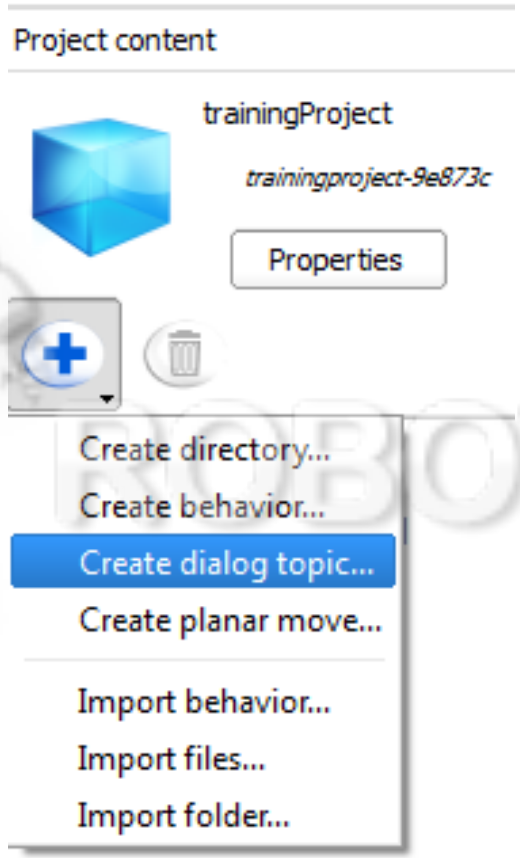
Outline

Bases of Dialog
Introduction to dialog

- ALDialog is a module that allows to create complex dialogs
- Dialogs can be easily integrated inside an application or Autonomous Life
- Or can be set as internal to an application

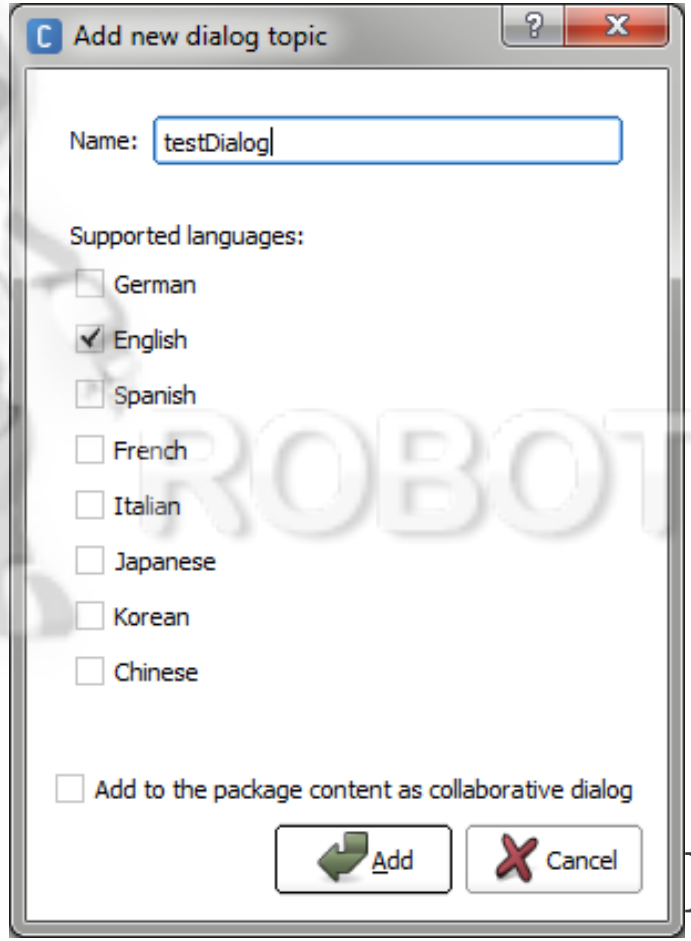
Bases of Dialog
Introduction to dialog

- Create a new dialog topic by clicking the + button in the project content panel



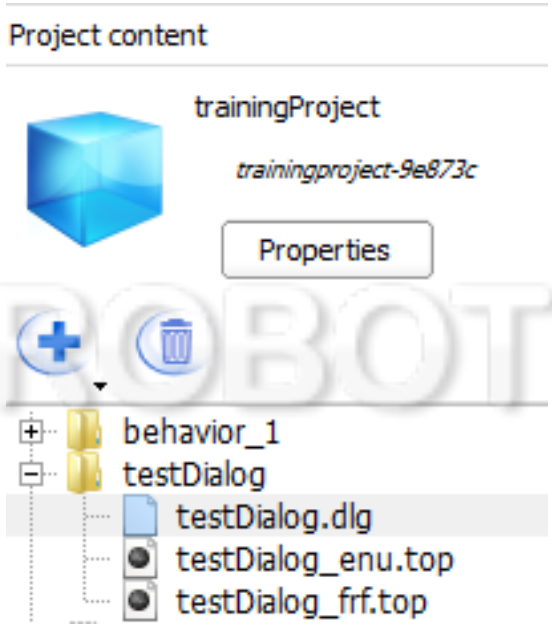
Bases of Dialog
Introduction to dialog

- Choose a name for your dialog
- Select languages
- Collaborative dialogs are used by Autonomous Life



Bases of Dialog
Introduction to dialog

- The corresponding files have been created in a new folder in your project
  - the .top files contains the dialog in a given language
  - the .dlg file manages what languages are supported

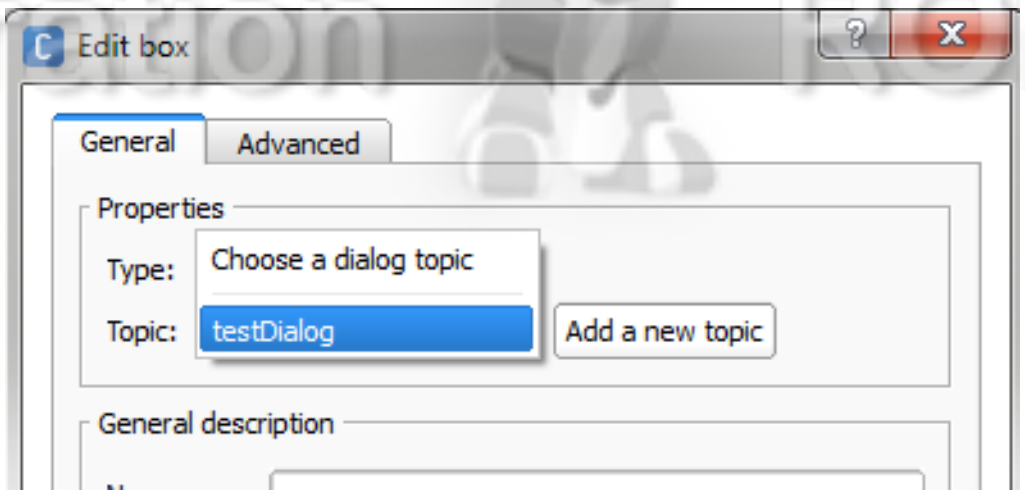


Bases of Dialog
Introduction to dialog

- A dialog topic is an entity living at project level
- It consists in a .dlg file referring to a bunch of .top files.
  - Adding a new language in .dlg will create the corresponding .top
  - Careful, as deleting a supported language will also delete the .top

Bases of Dialog
Introduction to dialog

- Create a new dialog box
- Link it to your dialog
  - Several boxes can refer to the same dialog
  - If you enter a new name, the corresponding files will automatically be created.
- Several topics can be active in parallel



SoftBank Robotics

Bases of Dialog
Introduction to dialog

- Open the testDialog_enu.top file
- It should look like this:

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

- For the robot to say something, add:

```
u:(hello) hello, human friend.
```

Bases of Dialog
Introduction to dialog

- Run the box
  - The robot listens to 'hello'
  - And answers 'hello, human friend. '
- Prefer long trigger sentences as they are easier to recognize
- Avoid long answers for the robot, as they are boring
- Use the dialog menu to see what the robot understood

Bases of Dialog
Introduction to dialog

```
u:(hello)
  hello, human friend.
u:(what is your name)
  My name is Pepper
```

- Indentation doesn't matter
  - But is a good practice
- The robot continuously listens to all triggers
  - All **u** rules are in parallel

Bases of Dialog
Introduction to dialog

- Forbidden characters in 'Human speech'
  - ; , : $
  - Apostrophe (‘) is allowed but not the typographic apostrophe (’)
  - Punctuation: ?,!...

Bases of Dialog
Introduction to dialog

Special characters:

- # for commenting
- Several options : [and]

```
u:(Can you [help advise] me) [yes absolutely]
```

  - In human speech, both possibilities activate the rule
  - In robot answer, different choices are said sequentially if you run the rule several time
  - There can be an infinite number of options

Bases of Dialog
Introduction to dialog

Special characters:

- Quotes for word sequence definition

```
u:(What should I visit in [Paris "this city"]) ["I
    will" "let me"] tell you
```

- Optional parts : { and }

```
u:(Are you a {tourist} guide) No, only a robot with
    a tourist database
```

Bases of Dialog
Introduction to dialog

Answer choice:

- By default, in a choice, the robot will say the first choice, then the second if the rule is played again...

```
u:(Where should I go next) ["the Eiffel tower" "
    Montmartre" "the Parthenon"]
```

- Human: Where should I go next -> Robot : the Eiffel tower
- Human: Where should I go next -> Robot : Montmartre
- Human: Where should I go next -> Robot : the Parthenon
- Human: Where should I go next -> Robot : the Eiffel tower
- ...

Bases of Dialog
Introduction to dialog

Answer choice:

- ^rand [answer1 answer2]: The robot will select a random valid sentence

```
u:(Where should I go next) ^rand["the Eiffel tower" "
    Montmartre" "the Parthenon"]
```

- Human: Where should I go next -> Robot : Montmartre
- Human: Where should I go next -> Robot : the Parthenon
- Human: Where should I go next -> Robot : the Eiffel tower
- Human: Where should I go next -> Robot : the Parthenon
- ...

SoftBank Robotics

Bases of Dialog
Introduction to dialog

Word combination:

- ^repeat: human can tell whatever word from the list, any number of times

```
u:(^repeat[word1 word2 word3 wordn]) answer
```

```
u:(This train is ^repeat[really very] fast.) You are
    right.
```

- Human: This train is fast -> Robot : You are right.
- Human: This train is really really very really fast -> Robot : You are right.

Bases of Dialog
Introduction to dialog

Special characters:

- Any choice: *

```
u:(I need to go to * )
Let's go.
```

  - Use wildcards sparingly !

- Forbidden word : !

```
u:(I want to go to Montparnasse !tower)
It's not far.
```

Bases of Dialog
Introduction to dialog

Subrules and scope :

- Subrules are activated when the upper rule has been played

```
u:(input1) answer                    Scope of the rule u:
   u1:(input2) answer
   u1:(input3) answer      Scope of the rule u1:
     u2:(input4) answer
        u3:(input5) answer
     u2:(input6) answer

proposal: sentence
    u1:(input7) answer
    u1:(input8) answer
```

Bases of Dialog
Introduction to dialog

Subrules:

- If one rule of one level (u1) is played, others rules from the same level are desactivated
- If a rule not belonging to the subrules is played, the subrules are desactivated
- You can use the ^stayInScope function to keep a subrule loaded

```
u:(input) answer
   u1:(input1) answer ^stayInScope
   u1:(input2) answer
```


SoftBank Robotics

Bases of Dialog
Variables and events

Local variables:

- Catch some part of human input and use it in robot answer
- _ and $ characters

```
u:(i will take a _[taxi train] to the _[airport Louvres
    ])
Let's find a $1 so you can go to the $2
```

Bases of Dialog
Variables and events

Global variables:

- Can be re-used in the dialog
- Assign with =
- Reuse by putting name
- Erase with ^clear

```
u:(I want to go to the _[airport Louvres]) I will find
    you a way to $1 $destination=$1
  u1:(Where should I go) Follow my instructions to go to
      $destination
  u1:(I changed my mind) no problem ^clear(destination)
```

Bases of Dialog
Variables and events

Conditions:

- Possible conditions: ==, >, <, <>
- If human input, the rule is triggered when input is heard if condition is true

```
u1:(How long will it take $destination==airport) More
    than one hour
```

Bases of Dialog
Variables and events

Conditions:

- Condition based on an empty variable will be ignored
- If the condition is in the robot answer, the sentence will be said only if the condition is true
- ^first makes the robot say the first valid sentence

```
u1:(How much does it cost)
^first[
"$destination==airport approximatively 50 euros"
"$destination==Louvres approximatively 15 euros"
"i don't know"
]
```

Bases of Dialog
Variables and events

Events:

- Use events to react to box inputs
- And variable for box outputs of type string

```
u:(e:onStart) Hello, where do you want to go ?
u:(I want to got to the airport)
Ok.
$computeDestinationOutput=$destination
```

Bases of Dialog
Variables and events

Events:

- e:eventName can also contain a value $eventName

```
u:(e:foundMetroLine)
You need to use the metro number $foundMetroLine
```

- Events are triggered when a variable is changed
- Or outside the dialog, through ALMemory

All dialog variables and events are ALMemory values!

Bases of Dialog
Variables and events

Events:

```
u:("e:foundMetroLine tell me how to go")
You need to use the metro number $foundMetroLine
```

- You have to tell the sentence after the event has been trigered

```
u:(e:FrontTactilTouched $FrontTactilTouched==1) Stop
    touching me!
```

- The robot only reacts to presses on the tactile sensor (value to 1) and not releases (value to 0)
  - Otherwise the output will be triggered twice each time we tap on the font tactile sensor

Bases of Dialog
Variables and events

Events:

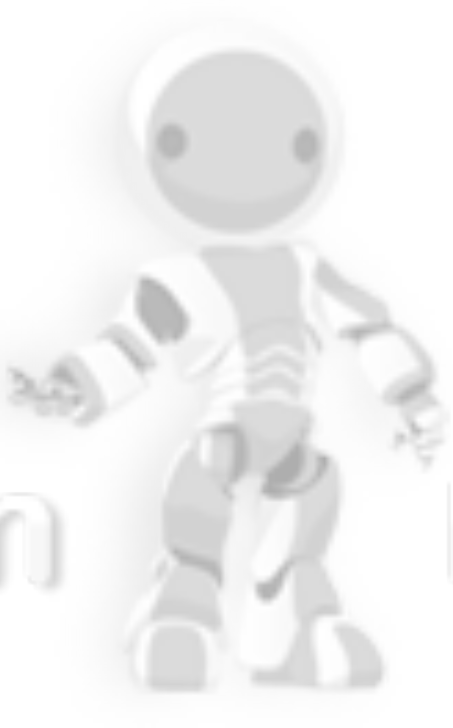
- Some events are triggered by the dialog engine
  - e:Dialog/NotSpeaking5 if the robot heard nothing for 5 seconds
  - e:Dialog/NotUnderstood if the robot could not match what it heard to a rule
  - e:Dialog/Fallback if the matched rule is a fallback rule

Bases of Dialog

Summary

- Dialog Topic & Boxes
- Language choice
- Basic dialog syntaxes
- Rule levels & Scope (^stayInScope)
- Local and global variables (_ and $ characters)
- Conditions (== ,^first)
- Events (e:onEvent)
- Dialog engine events (NotSpeaking5...)

Outline



Advanced Dialog
Advanced Dialog

A **concept** is a list of words and/or phrases that refer to one idea.
Syntax: concept:(concept-name) [words and phrases here]

```
concept:(confirmation)
    [ yes
    "why not"
    sure  ]
```

Advanced Dialog
Advanced Dialog

You can then use your concept:

```
concept:(confirmation) [ yes "why not" sure ]
u:(hello Pepper)
    hello human do you want a hug?
    u1:(~confirmation)
        cool lets do that !
    u1:(no)
        no problem, may be later
```

Note : it is not a python list, no commas

Advanced Dialog
Advanced Dialog

You get the concept matched within the rule in a local variable

```
concept:(confirmation) [ yes "why not" sure ]

u:(_~confirmation)
  You said $1
```

Advanced Dialog
Advanced Dialog

A concept is useful to :

- get many words linked to the same references
- write clean dialog
- provide more flexibilty when you want to update

Advanced Dialog
Advanced Dialog

You can use your concept as any "list" :

```
concept:(love) ["i love you" "you are beautiful" "you
    are so smart"]
concept:(activities) [dance sing play talk]
u:(hello)
hello human ^rand[~love] .
I can ^enumerate[~activities]
```

Advanced Dialog
Advanced Dialog

You can define your concept in a .top file then "include" it to your dialog:

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

include: file.top
```

Advanced Dialog
Advanced Dialog

A concept file is useful to :

- reduce the size of your dialog
- re-use your concept in all your dialog

Advanced Dialog
Advanced Dialog

Dynamic concept

In the topic file add this line :

```
dynamic: animals
```

In your python code create the concept :

```
dial=session.service("ALDialog")
dial.setConcept("animals", "enu", ["dog","cat","duck"])
```

Language examples: "enu", "English", "jpj", "Japanese"

Advanced Dialog
Advanced Dialog

A dynamic concept is useful to :

- update dynamically your concept
- use a database or a webservice to fill up your concept
- separate content and logic of your dialog
- translate your dialogs

Advanced Dialog
Advanced Dialog

Define a **proposal** and use of ^**goto** or ^**gotoReactivate** :

```
proposal: %name_of_tag
Hello you !
u1:(how are you)
Fine

u:(hello Pepper)
    ^goto(name_of_tag)

u:(hello again)
    ^gotoReactivate(name_of_tag)
```

Advanced Dialog
Advanced Dialog

A proposal is useful to :

- reduce the size of your dialog
- avoid too much similar subrules
- clean architecture
- some enumeration

Several topic progression functions exist :^nextProposal,^ previousProposal,^sameProposal,^goto,^gotoRandom,^ gotoReactivate,^topicTag

Advanced Dialog
Advanced Dialog

Tags for voice tuning can be used within dialogs to adapt prosody, speed, ...

```
    u:(hello)
        hello \pau=1000\ you are \rspd=150\ speedy today
```

Marks a 1 second pause and increase 50% quickest than normal

Advanced Dialog
Advanced Dialog

- \vct=150\ Say the sentence with a pitch of +50%
- \rspd=50\ Say the sentence 50% slower than normal speed
- \pau=1000\ Insert a pause of 1s
- \style=value\ change style of voice : neutral (default value) / joyful / didactic
- \rst\ Resetting control sequences to the default

Advanced Dialog
Advanced Dialog

- \bound=[WSN]\ Modify phrase boundary : (W: Weak, Strong„ No boundary)
- \emph=value\ Word prominence level (0: Reduced ; 1: Stressed ; 2: Accented)
- \eos=value\ End-of-sentence detection (0 or 1)
- \readmode=value\ Change read mode (sent, char, word)
- \tn=spell\\spell=2000\ Spelling and spelling duration pause

Advanced Dialog
Advanced Dialog

Animations can be run from the dialog directly

- ^start
- ^wait
- ^stop
- ^run

Example :

```
u:(e:peopleDetected)
    ^start(animations/Stand/Gestures/Hey_2)
        Hello my friends ! \pau=1000\ come here I
            have a secret to tell you
    ^wait(animations/Stand/Gestures/Hey_2)
```

Advanced Dialog

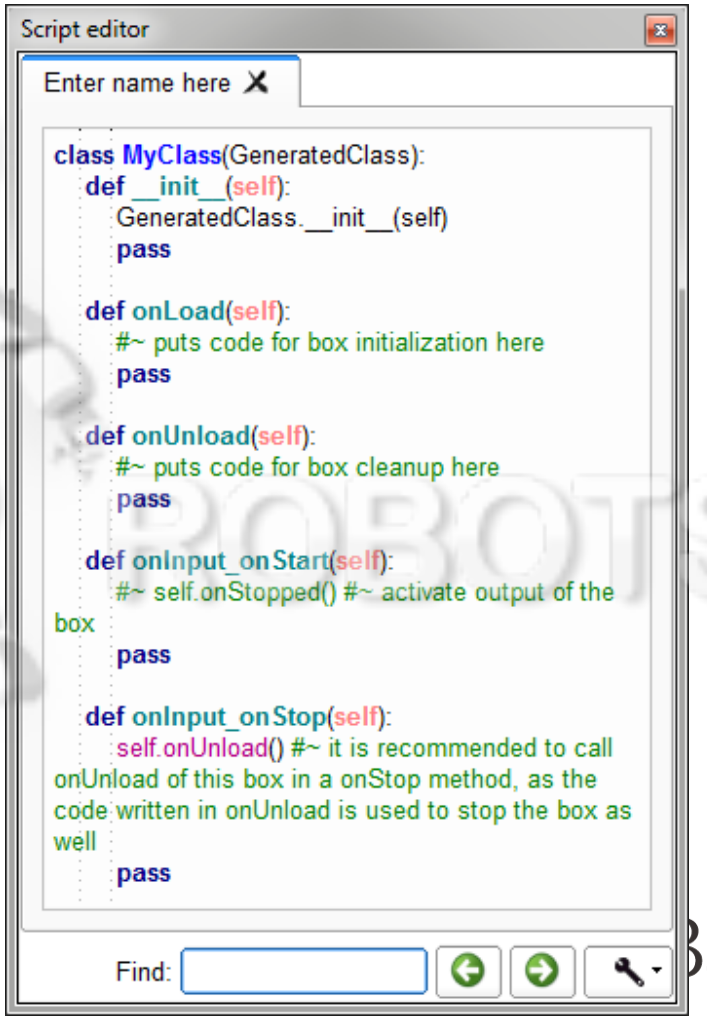
Summary

- Concepts
- Include
- Dynamic concepts
- Proposals
- Tags for voice tuning
- Run animations

Outline

Python in Choregraphe
Hello World in Python

- Create a new box of script type
  - Or use the 'Script' box from the templates
- Double-clicking the box opens the 'Script editor' window
- A Choregraphe box is in fact a Python class
  - The 3 first lines initialize the class, you should never modify them !



Python in Choregraphe
Hello World in Python

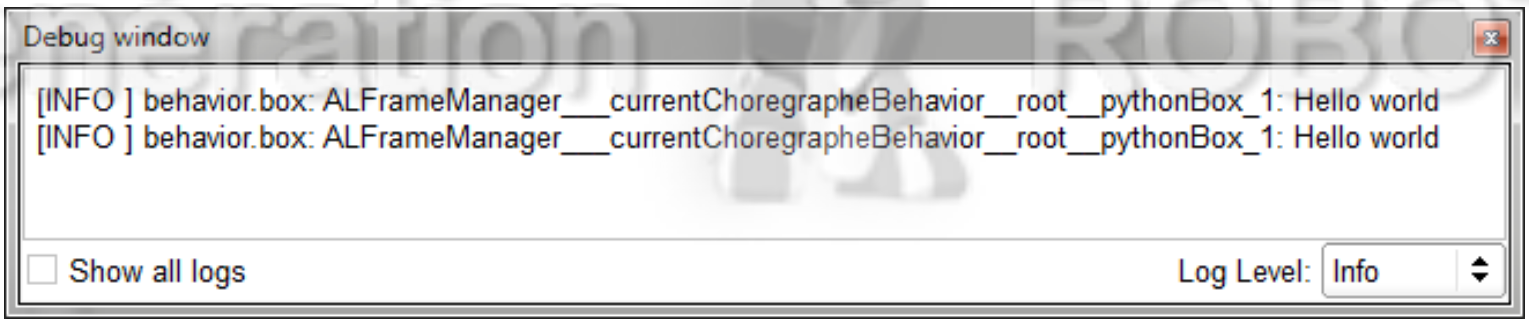
We want the box to produce a 'Hello world' log

- The *self.logger.info* method writes some text in the Debug Window
- Put your code in the *onInput_onStart* method
  - It will be executed when the onStart input is triggered
- As you now have a statement in the *onInput_onStart* method, you can remove *pass*

Python in Choregraphe
Hello World in Python

```
def onInput_onStart(self):
    self.logger.info("Hello world")
    #~ self.onStopped() #~ activate output of the box
```

When you trigger the input:



Python in Choregraphe
Hello World in Python

- The output is never activated
- Uncomment the second line to activate it
  - Use *self.outputName()* to trigger an output

```
def onInput_onStart(self):
    self.logger.info("Hello world")
    self.onStopped() #~ activate output of the box
```

Python in Choregraphe
Hello World in Python

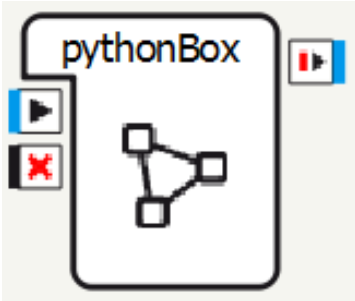
We now want to build some sentence using a name

- Modify the *onStart* input to receive a string data
- Add an argument in the method
- Use this parameter in the log


```
def onInput_onStart(self, param):
    self.logger.info("Hello " + param)
    self.onStopped() #~ activate output of the box
```

Python in Choregraphe
Hello World in Python

- Modify the *onStopped* input to send a string data
- Create a variable containing the sentence
- Put the sentence as argument of the output




```
def onInput_onStart(self, param):
    sentence = "Hello " + param
    self.logger.info(sentence)
    self.onStopped(sentence) #~ activate output of the
        box
```

Python in Choregraphe
Hello World in Python

We want to change the beginning of the sentence

- Add a new field in the box
  - In the *onLoad* method
  - This variable will be accessible from all methods in the box
- The *onLoad* method is run when the box is loaded
  - Put there your initialization code
  - Add some log to see when it is run

```
def onLoad(self):
    self.logger.info("initialization")
    self.sentenceStart = "Hello"
```

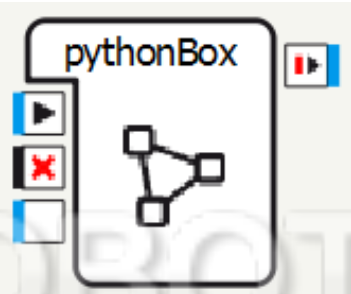
Python in Choregraphe
Hello World in Python

- Use the new field in your *onInput_onStart* method

```
def onInput_onStart(self, param):
    sentence = self.sentenceStart + param
    self.logger.info(sentence)
    self.onStopped(sentence) #~ activate output of the
        box
```

Python in Choregraphe
Hello World in Python

We will change the sentence start by using another input



- Create a new *changeSentence* input with a string type
- Create the *onInput_changeSentence* method

```
def onInput_changeSentence(self, param):
    self.sentenceStart = param
```

Outline

Advanced Python
Using the API

- APIs are methods provided by Softbank Robotics to control the robot
- They are organized in modules
  - Most used modules are ALMemory, ALMotion, ALTextToSpeech, ALDialog, ...
  - You can access the modules documentation locally from Choregraphe in "Help > API Reference" menu
  - You should be able to access the same online documentation from the robot's webpage
  - Advanced programmers can create their own modules in Python or C++

SoftBank Robotics

Advanced Python
Using the API

- Some modules inherit from ALModule
  - It allows you to ping the module (`ping()`), get its version (`version()`) or exit the module (`exit()`)
  - You can retrieve the module methods list and their description (`getMethodList()`, `getMethodHelp(const string& methodName)`)
  - You can use *qicli info* to get information on a module

Advanced Python
Using the API

- Documentation is available online
  - `http://doc.aldebaran.com/2-5/`
- Your are invited to explore the API as we are only going to explore some most important modules and functionnalities.

Advanced Python
Using the API

In order to use API methods, you first need to create a **connection** or a **proxy** on its module

Service = recommended way / Proxy = older way

- proxy/service connection is some kind of a pointer toward the module
- Once you have a proxy/service, you don't need to wonder if the module is local or remote

```
def onLoad(self):
    self.memory = self.session().service("ALMemory")
```

```
def onLoad(self):
    self.memory = ALProxy("ALMemory")
```

Advanced Python
Using the API

- Then use this proxy/service to call the methods of the module
- We will use the *insertData* method from ALMemory to create a new memory input

```
def onInput_onStart(self, param):
    sentence = self.sentenceStart + param
    self.memory.insertData("lastSeen", param)
    self.logger.info(sentence)
    self.onStopped(sentence) #~ activate output of the
        box
```

Advanced Python
Using the API

- You can check what is in the variable with the 'Memory viewer' window
- It will stay in the memory until NaoQi is stopped
- This value can be retrieved and changed from any box
  - Create a new box to retrieve the data

```
#Retrieve a data from memory
try:
    var = self.memory.getData("lastSeen")
except:
    self.logger.info("this value does not exist")
```

- You can create an event while setting a value

```
self.memory.raiseMicroEvent("lastSeen", param)
```

Advanced Python
Using the API

Box libraries vs API functions

- Use boxes because:
  - Boxes are ready to use
  - Boxes integrate safety parts (stop walking if you fall, don't execute this function if it's already executing...)
  - Moves are easier to record
  - Resource manager
- Do not reinvent the wheel
- Look inside existing boxes for good coding practice

Advanced Python
Using the API

Box libraries vs API functions

- Use Python API because:
  - You can store variables and use self-defined inputs and outputs
  - Much easier to do conditions, loops etc...
  - Do math, sleep...
  - Boxes can't do everything !

Advanced Python
Using the API

- Some functions are blocking (mainly motion functions)
  - The script stops until the function is finished

```
motion.angleInterpolation("HeadPitch", 1.0, 2.0, True)
self.logger.info("Print this after moving")
```

- To prevent a function from being blocking, you can use 'post'

```
motion.post.angleInterpolation("HeadPitch", 0.0, 2.0,
    True)
self.logger.info("Print this while moving")
```

Advanced Python
Using the API

Async calls with _async=True or qi.async

```
# naoqi way
fut = tts.say(self.sentence, _async=True)

# qi framework way
fut = qi.async(tts.say, self.sentence)

# add a callback to handle end of exploration
fut.addCallback(self.mycallback)

def mycallback(fut):
print "Fut value:" + str(fut.value())
```

Advanced Python
Using the API

ALBehaviorManager:

- ALBehaviorManager allows you to run other Choregraphe behaviors from a Choregraphe box, to set them as default behavior...
- From the behavior menu, you can add import behaviors

Advanced Python
Using the API

- You can get installed behaviors and running behaviors:

```
manager = session.service("ALBehaviorManager")

names = manager.getInstalledBehaviors()
print "Behaviors on the robot:"
print names

names = manager.getRunningBehaviors()
print "Running behaviors:"
print names
```

- You can start and stop behaviors:

```
manager.post.runBehavior(behaviorName)
time.sleep(5)
if (manager.isBehaviorRunning(behaviorName)):
    manager.stopBehavior(behaviorName)
```

Advanced Python
Python outside of Choregraphe

We want a standalone python script to access the NAOqi and say "Hello World" using the text to speech proxy of the NAOqi. Create a new file with the following content:

```
#!/usr/bin/env python
import qi
session = qi.Session()
session.connect("tcp://pepper.local:9559")
tts = session.service("ALTextToSpeech")
tts.say("First script")
```

Advanced Python
Python outside of Choregraphe

Let's break down the code.

```
import qi
session = qi.Session()
session.connect("tcp://pepper.local:9559")
```

Loads the qi framework. Creates a session. As we are not connected to a specific robot, like we are in Chorgperaph. We have to connect the session to the robot by it's IP and port through a tcp connection.

```
tts = session.service("ALTextToSpeech")
```

This gives us access to the text to speech module of the NAOqi.

```
tts.say("Hello World")
```

Finally we can call the say method with our desired text.

Advanced Python
Python outside of Choregraphe

The old way you may find in examples and exercices

```
#!/usr/bin/env python
from naoqi import ALProxy
tts = ALProxy("ALTextToSpeech","pepper.local",9559)
tts.say("Hello World")
```

Advanced Python
Python outside of Choregraphe

Let's take a look how to listen to events. Below you find the simplest example how you can listen to an event, raised in ALMemory.

```
import qi, time
def mycallback(value):
    print "callback",value

app = qi.Application(["TouchMyHead", "--qi-url=tcp://
    localhost:9559"])
app.start()
memory = app.session.service("ALMemory")
subscriber = memory.subscriber("FrontTactilTouched")
subscriber.signal.connect(mycallback)
time.sleep(10)
```

Advanced Python
Python outside of Choregraphe

```
def mycallback(value):
    print "callback",value
```

A simple callback function to see when an event is fired.

```
app = qi.Application(["TouchMyHead", "--qi-url=tcp://
    localhost:9559"])
```

Application initializes the qi framework, it extract –qi-* commandline arguments and initialise various options accordingly. It also provides facilities to connect the main session. This ease the creation of console applications that connects to a session or want to be standalone.

Advanced Python
Python outside of Choregraphe

```
app.start()
memory = app.session.service("ALMemory")
```

Starts the app that is linked to NAOqi and gets access to the ALMemory module.

```
subscriber = memory.subscriber("FrontTactilTouched")
subscriber.signal.connect(mycallback)
```

Subscribes to "FrontTactilTouched" and connects fired events to the function "callback"

Outline

Using the Tablet
Tablet

Tablet Specifications

- Connected to the robot through Ethernet over USB
- The robot IP address seen from the tablet is 198.18.0.1
- Has its own wifi connection (can be disabled)
- Button behind the tablet on the right
- Developped specifically for the robot
- Android 5.1

Using the Tablet
Tablet

In order to use the tablet we need to subscribe to the tablet services. See boxes StartApp, ShowImage, ...

```
tabletService = session.service("ALTabletService")
```

Using the Tablet
Tablet

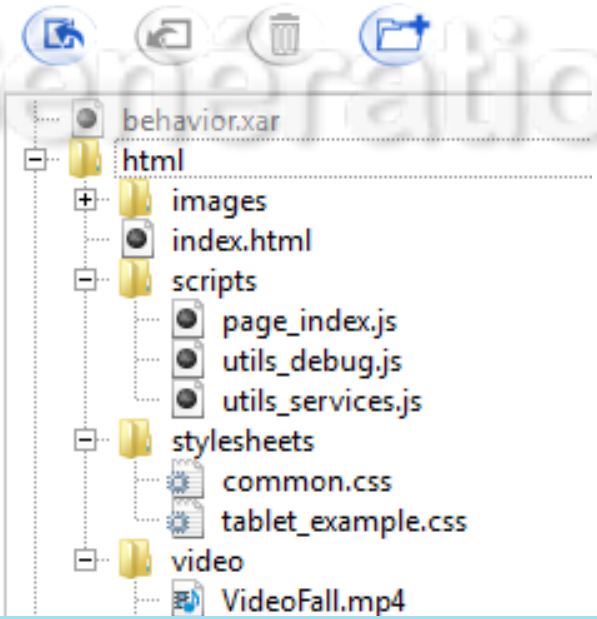
Then you can display a web site : (wifi has to be activated on the tablet in this example)

```
tabletService.showWebview("http://www.google.com")
```

Using the Tablet
Tablet

If you want to host your own web site you need to:

- Add an "html" folder to you application (downcase)
- Add a default index.html file to your behavior (mandatory)
- Add any page, ressource, libraries, framework you need (hello.html, sound.mp3, video.mp4, angular.js, jquery, ...)
- Upload your behavior on the robot



Using the Tablet
Tablet

- Once your web site is hosted onto the robot you can access it.
- The url rewriting is handled by Naoqi.
- The url is composed of the ip, "apps" and the name of the package
- ex: http://198.18.0.1/apps/package_name

```
url = "http://198.18.0.1/apps/test_tab/hello.html"
tabletService.showWebview(url)
```

Using the Tablet
Tablet

You can turn on and off the tablet by calling the specific method:

```
tabletService.turnScreenOn(True) # turn screen on
tabletService.turnScreenOn(False) # turn screen off
```

Using the Tablet
Tablet

You can display some videos on the tablet :

- add the video to your html folder
- create a video launcher (python)
- upload your behavior to the robot

Using the Tablet
Tablet

You can display some video on the tablet, stop, pause and resume it:

```
    url = "http://198.18.0.1/apps/test_tab/x8.mp4"
    tabletService.playVideo(url)
    tabletService.pauseVideo() # pause
    tabletService.resumeVideo() # resume
    tabletService.stopVideo()
```

Using the Tablet
Tablet

You can display some image onto the tablet :

- add the picture to your html folder
- create a picture launcher (python)
- upload your behavior to the robot

Using the Tablet
Tablet

You can display, hide, preload :

```
url = "http://198.18.0.1/apps/test_tab/img.png"
tabletService.showImage(url)
tabletService.showImageNoCache(url)
tabletService.hideImage()
tabletService.preLoadImage(url)
```

Using the Tablet
Tablet

Create an onTouch subscriber:

```
    tabletService.onTouchDown.connect(self.mycallback) #
        link the onTouch event to a call back function

    def mycallback(self, x, y): #callback function that
        just displays position
        self.logger.info("X :"+str(x))
        self.logger.info("Y :"+str(y))
```

Using the Tablet
Tablet

The tablet has a slow CPU, some events and animations may not be as responsive as on a browser. Be careful with "heavy" JavaScript framework.

Using the Tablet
Tablet

There is a screen size difference between the "old tablet" (for Pepper 1.7, 1.6 and early models) and the "new tablet" (for Pepper 1.8a, 1.8). This change can affect existing tablet content of Pepper Apps. The "old tabletś webview" resolution is 1707 x 1067. The "new tabletś webview" resolution is 962 x 601. To make your app work correctly on the new tablet, you need to add a fix to your JavaScript code:

```
See file 13-005-tablet-display-fix.js
```

Using the Tablet

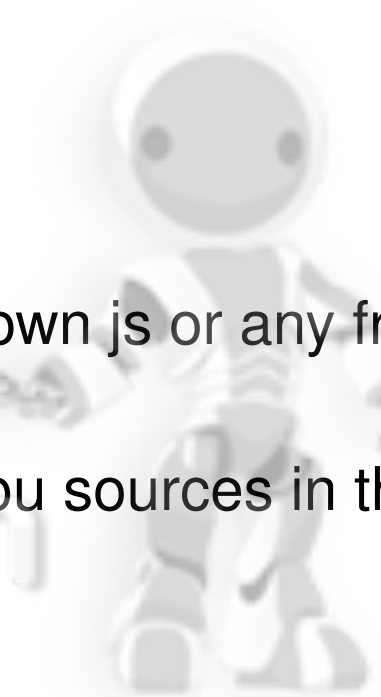
Summary

- Tablet main specifications
- Connect to the tablet through a service
- Robot IP 198.18.0.1
- Display an external website on the tablet
- Display your own website from the project
- Turn on/off the tablet
- Display videos and images
- Interact touching the tablet

Outline

Using JavaScript on the Tablet

Javascript



- You can embed you own js or any framework (test size and compatibility)
- You should embed you sources in the application (avoid remote libs)

Using JavaScript on the Tablet
Javascript

- QiMessaging library provides JavaScript bindings to use QiMessaging services (modules) in a web browser
- This library can be used to build HTML5 applications for your robot.

Import the sdk present from the robot to the tablet:

```
2.5: <script src="/libs/qi/2/qi.js"></script>
2.4: <script src="/libs/qimessaging/2/qimessaging.js"></
    script>
```

Using JavaScript on the Tablet
Javascript

Then you can create a session
QiSession 2.0 uses Promises and Futures
Syntax :

```
p.then(function(value) {
    // Promise resolved
}, function(error) {
    // Promise rejected
});
```

Using JavaScript on the Tablet

Javascript

Example to connect a session :

```
QiSession(connected, disconnected, location.host);
session = null;

function connected(s) {
    session = s;
    console.log("QiSession connected");
}

function disconnected(s) {
    console.log("QiSession disconnected");
}
```

Using JavaScript on the Tablet

Javascript

You can use this session to subscribe to a ALMemory event:

```
session.service("ALMemory").then(
function (memory) {
    memory.subscriber("completeName").then(
        function (subscriber) {
            subscriber.signal.connect(function(state){
                console.log(state);
            });
        },
        function (error) {
            console.log("An error occurred:", error);
        });
    });
```

Using JavaScript on the Tablet

Javascript

You can use this session in order to link a callback function to Naoqi:

```
session.service("ALMemory").then(
    function (memory) {
        memory.insertData("name_of_var", "hello world");
        memory.raiseEvent("training/name", 1);
    },
    function(error){
        console.log("An error occurred:", error);
    });
```

Using JavaScript on the Tablet

Javascript

Debugging

Access the website of your robot throught it ip and use you navigator console :

ex : http://123.123.123.123/apps/mypackagename/

ADB is available when connecting to the robot through SSH

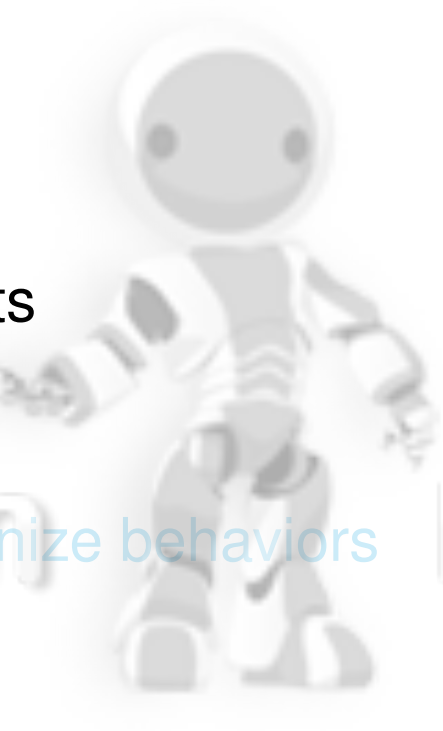
```
adb logs:
/home/nao/.local/share/PackageManager/apps/j-tablet-
    browser/adb logcat

adb shell:
/home/nao/.local/share/PackageManager/apps/j-tablet-
    browser/adb shell
```

Using JavaScript on the Tablet

Summary

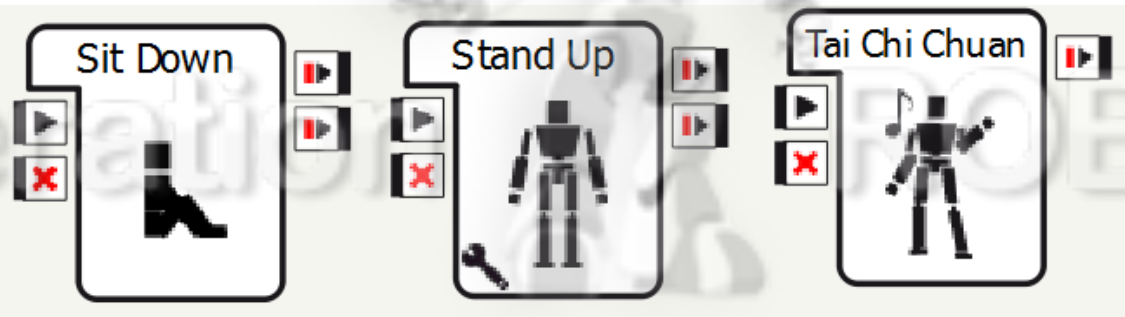
- QiMessaging js library
- Debugging with navigator
- ADB logcat and ADB shell

Outline

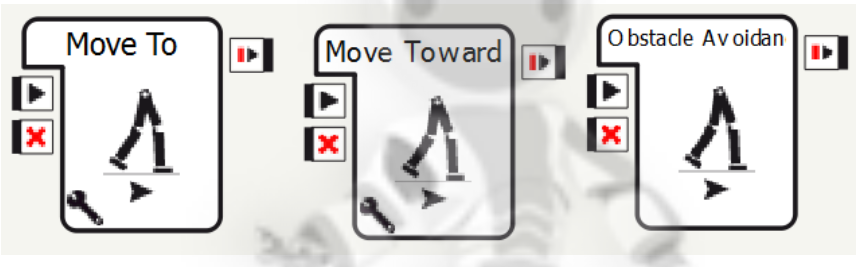


Move
Movement boxes

- Other motion boxes include stiffness and posture detection, safeties, retries...



Move
Movement boxes

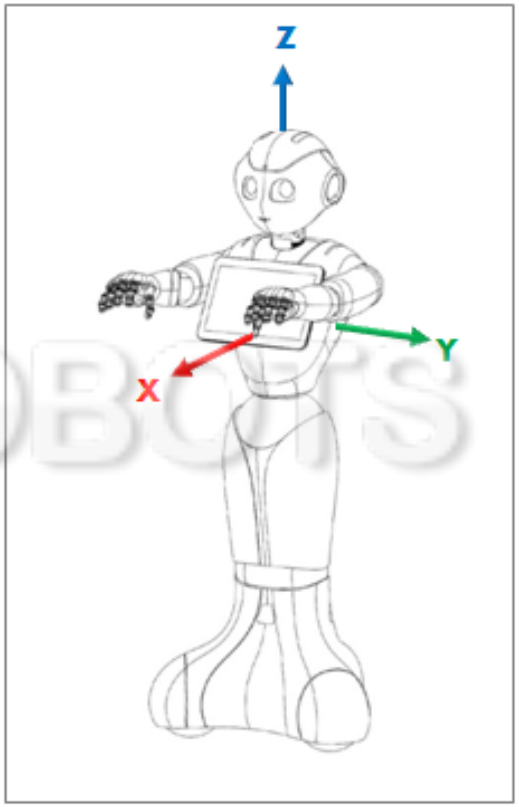


- Several walk boxes are available
  - Determine a stop point for 'Move To'
  - Set a direction for 'Walk Toward'
  - The robot uses its sonar in 'Obstacle avoidance'
  - Parameters are in meters and radians

Move
Movement boxes

Robot frame:

- Origin is on the floor between the two feet
- X axis is toward the robot's front
- Y axis is from right to left
- Z axis is up
- Theta angle is positive if the robot turns left

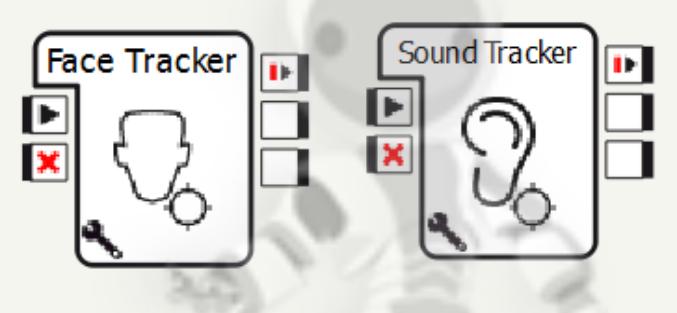


SoftBank Robotics

Move
Movement boxes

- The robot's motion is omnidirectional: it can go forward, backward or smoothly turn
  - Distance may not be precise if it slips.
  - Be careful with cables and obstacles

Move
Movement boxes



- Trackers boxes are complex behaviors that give life to the robot
- Whole body tracker allows an increased tracking space
  - Works for faces or red balls
  - The robot must be standing at start

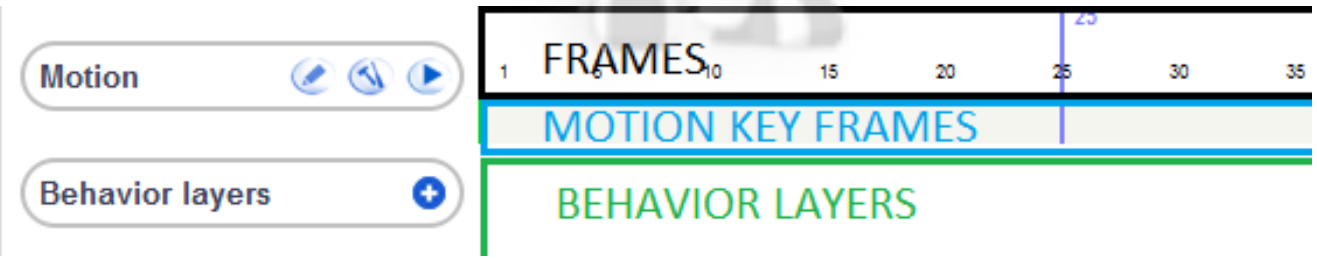
Move
Creating movements

- Movements are stored in timeline boxes
- Create a new box with type 'timeline'
  - Or use the 'Timeline' box from template

Move
Creating movements

The timeline panel is divided in 3 parts

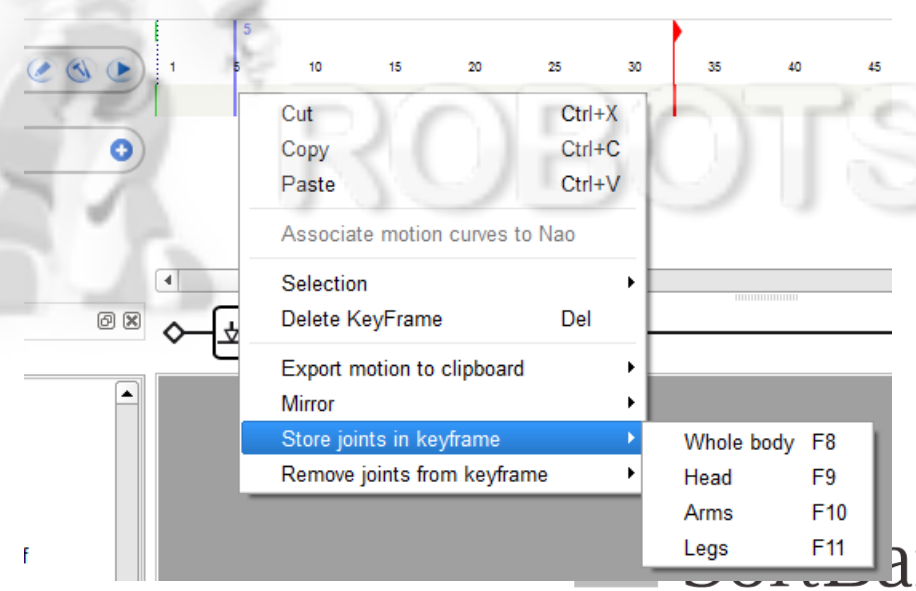
- The frames parts
  - By default 25 frames per second
- The motion key frames part
  - Motion key frames contain joint positions
- The Behavior layers part
  - To synchronize boxes and movements



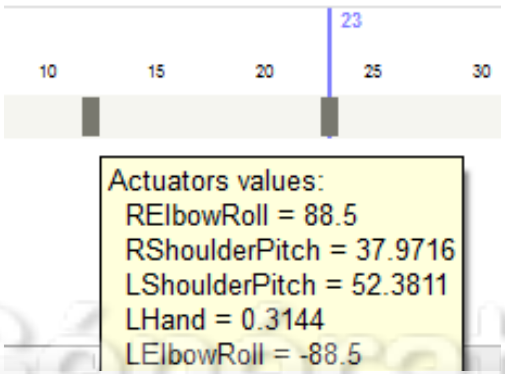
Move
Creating movements

- A movement is a set of positions a joint has to reach at a certain time
- To record a position

1. Click on the frames part to select a frame
2. Right-click on the motion keyframes part and select 'store joints in keyframe'



Move
Creating movements



A new keyframe (grey rectangle) appears

- You can drag and drop it, copy and paste...
- If you let your mouse on the keyframe, the content is displayed (in degrees)
- Don't put a keyframe on the first frame: the robot needs some time to move

- Click on the blue triangle to start the execution of the movement
  - No need to launch behavior (shortcut)
  - Don't forget stiffness

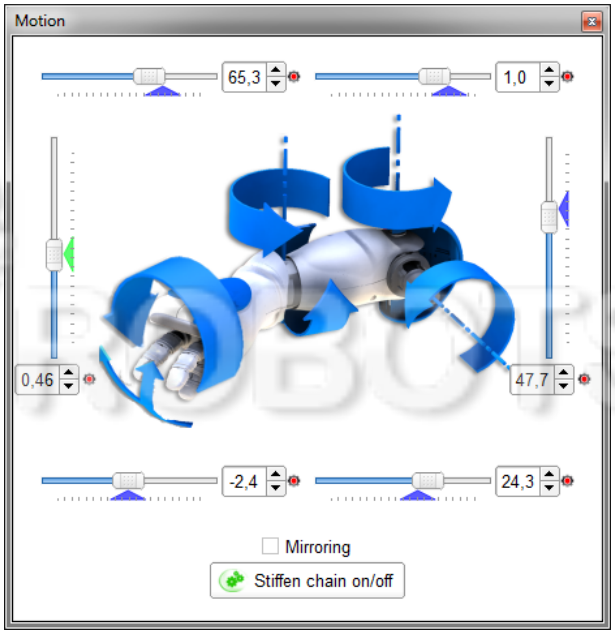
SoftBank Robotics

Move
Creating movements

- Motion will interpolate between the keyframes
  - Using Bézier interpolation
- If you click on the keyframes part, the robot reaches the intermediate positions
- You can select some frames and shift or compress keyframes

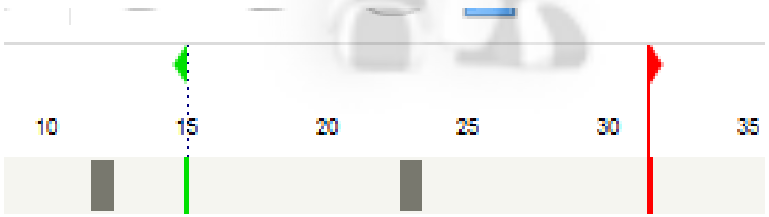
Move
Creating movements

- If you click on a part of the robot on the 'Robot view', you can modify the joint positions one by one in the 'Inspector'
- The little red dots indicate if a joint has some position to reach ih this keyframe
  - Click on the dot to remove a joint from the keyframe



Move
Creating movements

- By right-clicking a keyframe, you can mirror it:
  - Positions for right arm and leg will be transfered to left arm and leg and vice versa
- The green and red flags indicate start and stop of the timeline
  - You can decide to skip the beginning or the end of the movement

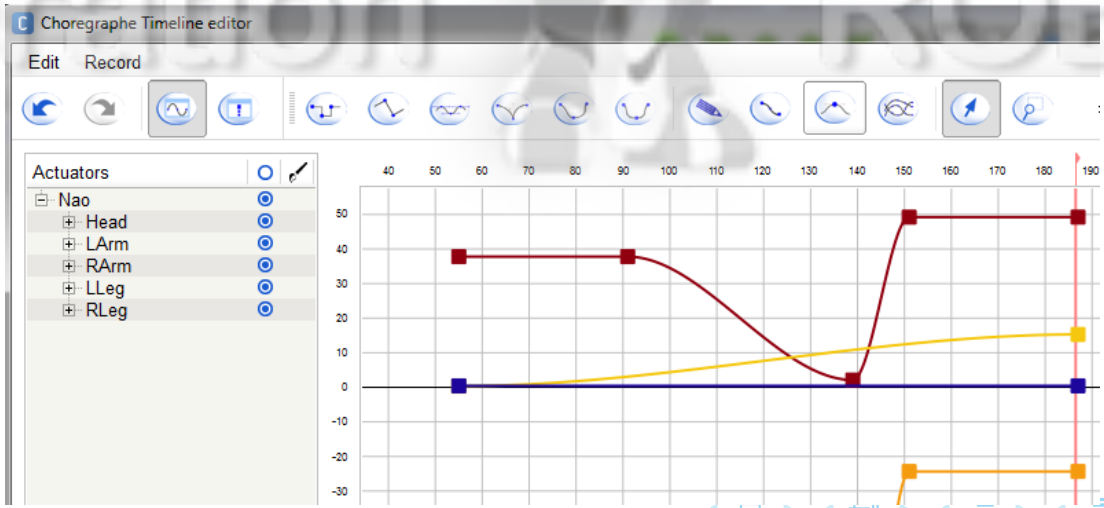


SoftBank Robotics

Move
Creating movements

Timeline editor

- To have a finer view on you movement, open the timeline editor window by clicking the 
  - Visualize the curves corresponding to joints movements
  - Modify keyframes joint by joint
  - Change interpolation type
  - Remove unnecessary keyframes ('simplify' button)



SoftBank Robotics

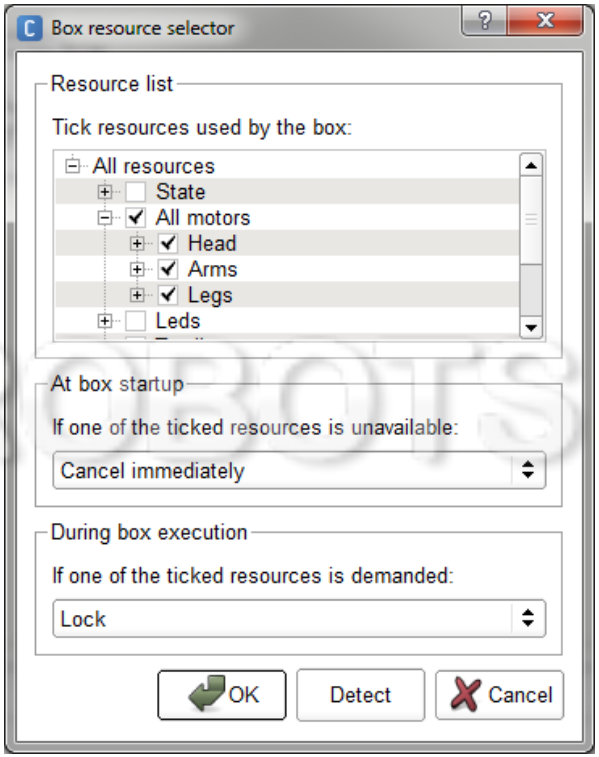
Move
Creating movements

Recording mode

- The  button put the robot in 'Animation mode'
- Manage stiffness with:
  - Middle head sensor for the head
  - Hand sensors for the arms
  - Bumpers for the legs (Nao only!)
- For recording a position just tap the head tactile sensors

Move
Resource manager

- A resource is an element needed for executing a task or a program (motor, microphone...)
- Resource management is crucial when several behaviors are run in parallel
- Right-click your box and choose 'Edit Resources'
  - The box can detect used resources for timeline boxes



Move
Resource manager

At box startup	If resources are not available
Cancel immediately	The box is not executed
Wait	The box is executed if resources are freed within n seconds
Wait infinitely	The box is executed as soon as resources are freed (may be never)

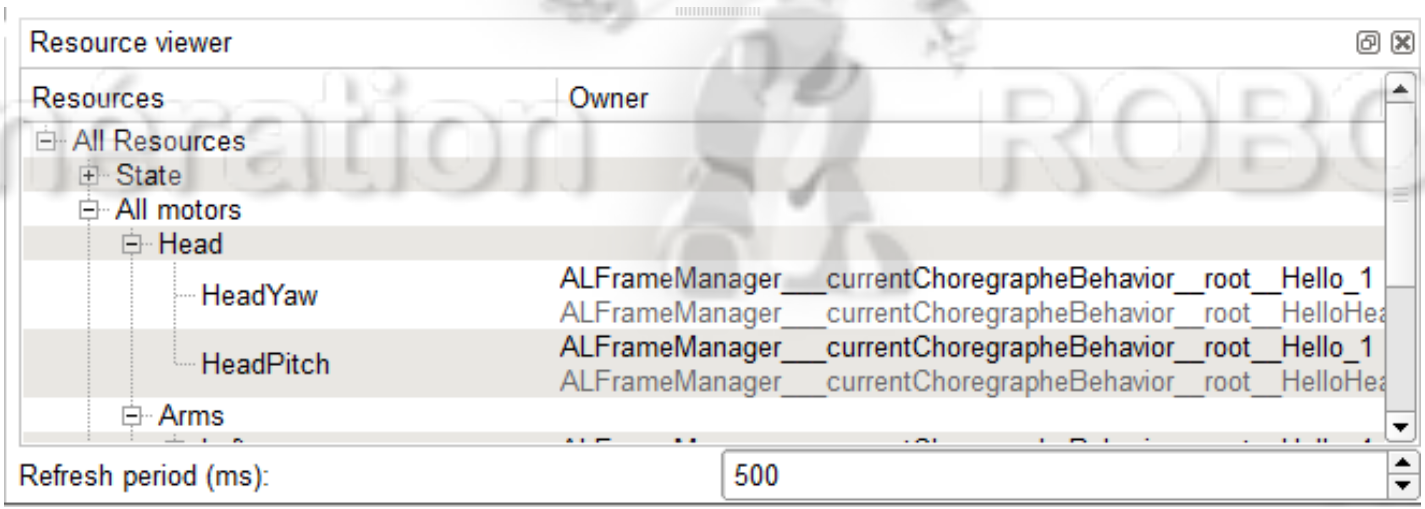
A box can't block a part of the resources so resource managing can't create deadlocks

Move
Resource manager

During box execution	if resources are asked by another box
Lock	The box keeps resources
Stop the box	The box is stopped and the resources released
Trigger box callback	onResource method in python code is executed

Move
Resource manager

- The 'resource viewer' panel in Choregraphe shows:
  - The used resources in black
  - The asked resources in grey
  - You can modify the refresh period



SoftBank Robotics

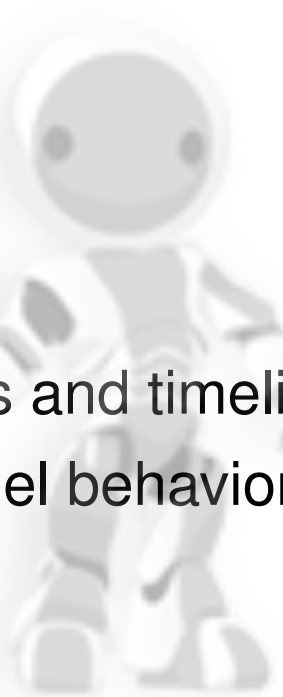
Move

Summary

- Boxes (moves)
- Timeline box type
- Frames, Motion keyframes
- Record positions methods
- Change joints positions from robot view
- Timeline editor fine tuning method
- Animation mode method
- Set frame Start and End
- Ressources management (box Edit ressource, Ressource viewer)

Outline

Integrate and synchronize behaviors
Add boxes to your movement

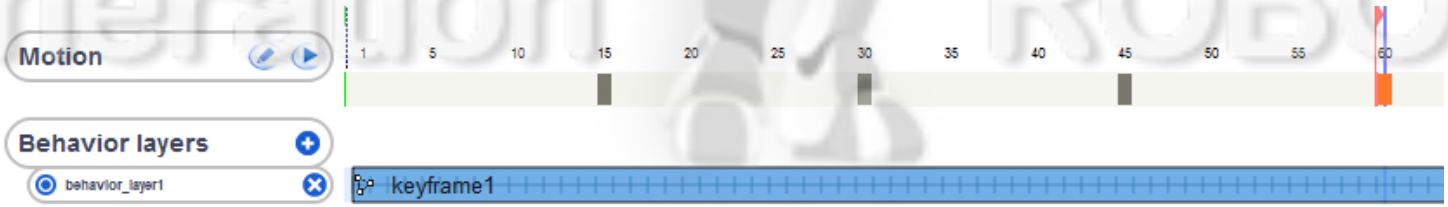


Behavior layers allow you to:

- Easily synchronize boxes and timelines
- Easily synchronize parallel behaviors
- Do state machines

Integrate and synchronize behaviors
Add boxes to your movement

- Create a timelime box
- Add some keyframes
  - For example a YMCA dance
- Add a behavior layer to your timeline

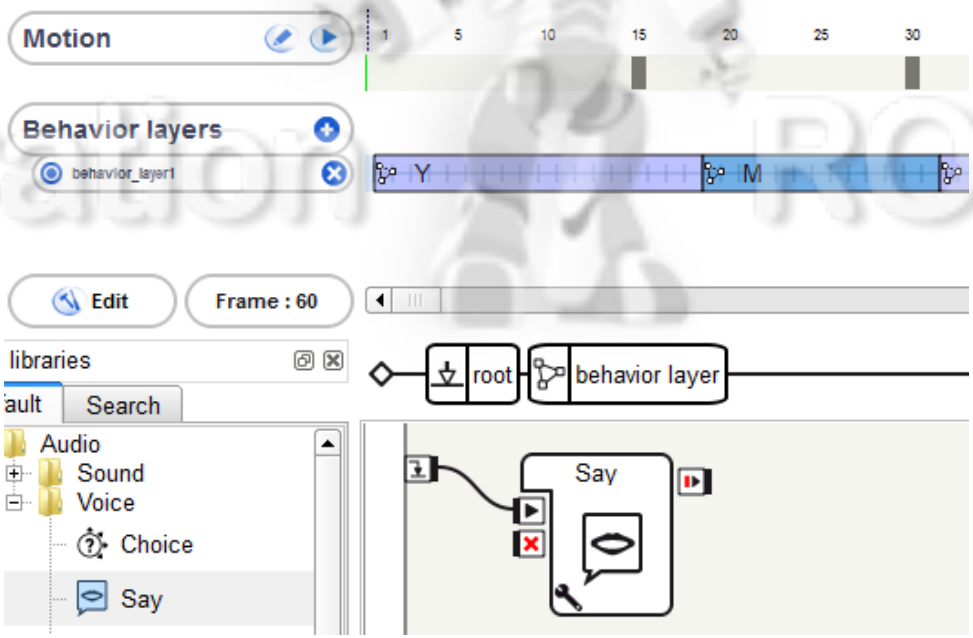


SoftBank Robotics

Integrate and synchronize behaviors

Add boxes to your movement

- Create several keyframes in your layer
  - Right-click and 'insert keyframe'
  - Rename them: right-click and 'edit keyframe'
- Put boxes in each key frame
  - For example, the robot says Y, M, C and A



SoftBank Robotics

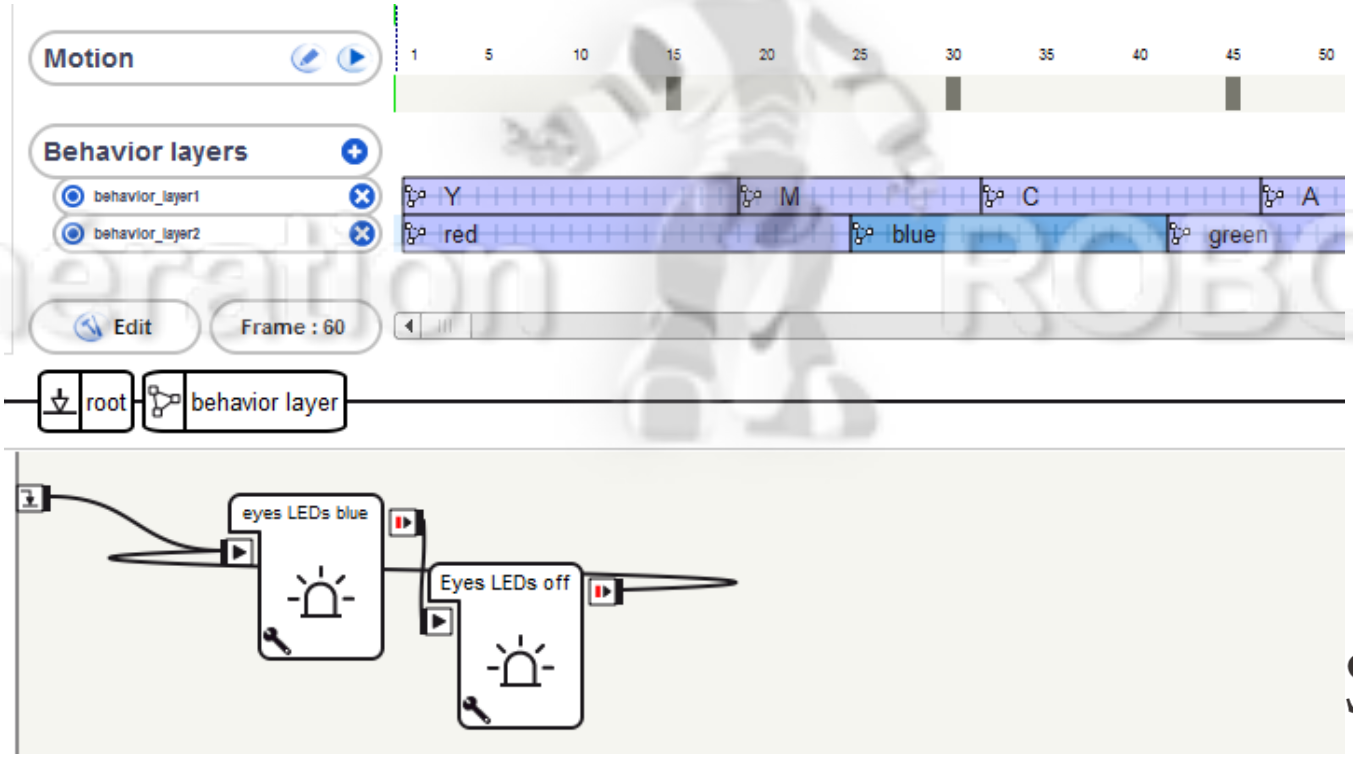
Integrate and synchronize behaviors
Add boxes to your movement

- A behavior layer can contain several keyframes
- A keyframe is a flow diagram linked to a certain number of frames
- When one of these frames is played
  - If the flow diagram is not loaded, it loads and starts
  - If the flow diagram is loaded, it continues playing
- If the current frame is not linked to the keyframe
  - If the flow diagram is loaded, it unloads

Integrate and synchronize behaviors

Add boxes to your movement

- You can add another behavior layer to have the eyes leds twinkle
- The two layers refers to the same frames



SoftBank Robotics

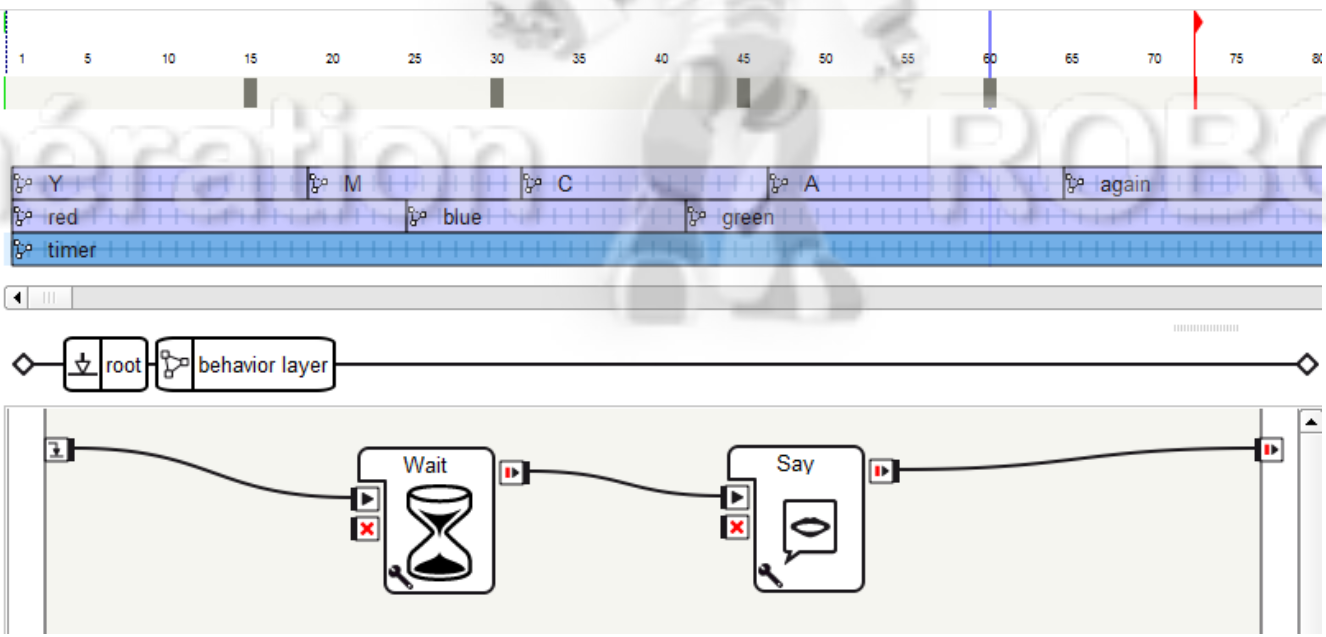
Integrate and synchronize behaviors
Add boxes to your movement

- You can drag and drop a keyframe, copy and paste it
- The timeline box stops
  - At the end of the movement
  - If the  output is triggered
- Stopping the timeline box stops the movements and unloads all boxes

Integrate and synchronize behaviors

Add boxes to your movement

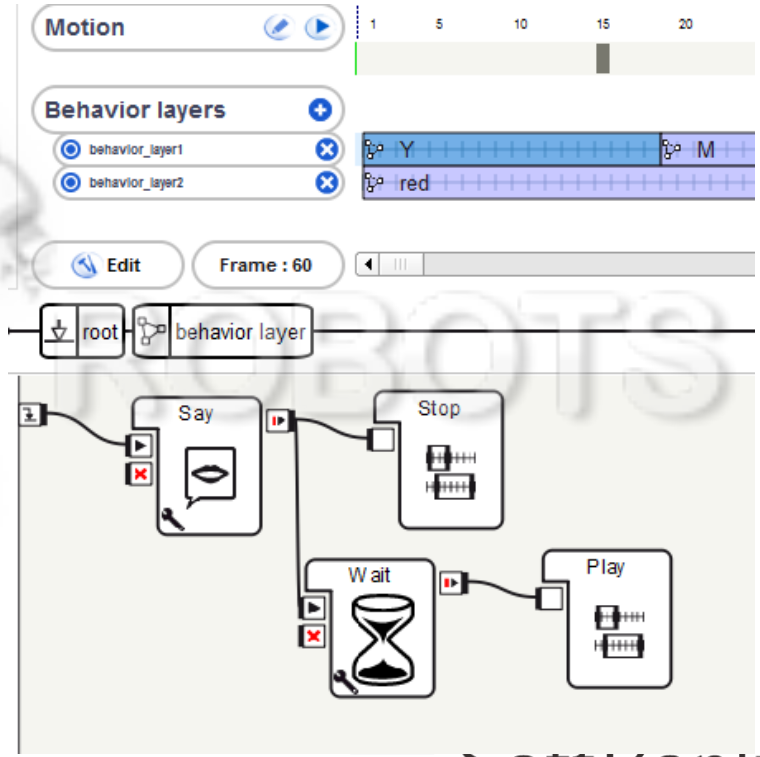
- Add a layer with a wait box to have a timeout
  - As there is only one keyframe in the layer, it will never be unloaded
  - Therefore the timer will never be restarted
- You can mute a layer by clicking the button



SoftBank Robotics

Integrate and synchronize behaviors
Control time with boxes

- In the previous example, the timeline decided when the boxes were launched
- But some boxes can affect the timeline
  - The 'Stop' box pauses the timeline at the current frame
  - The 'Play' box makes it go again from the current frame

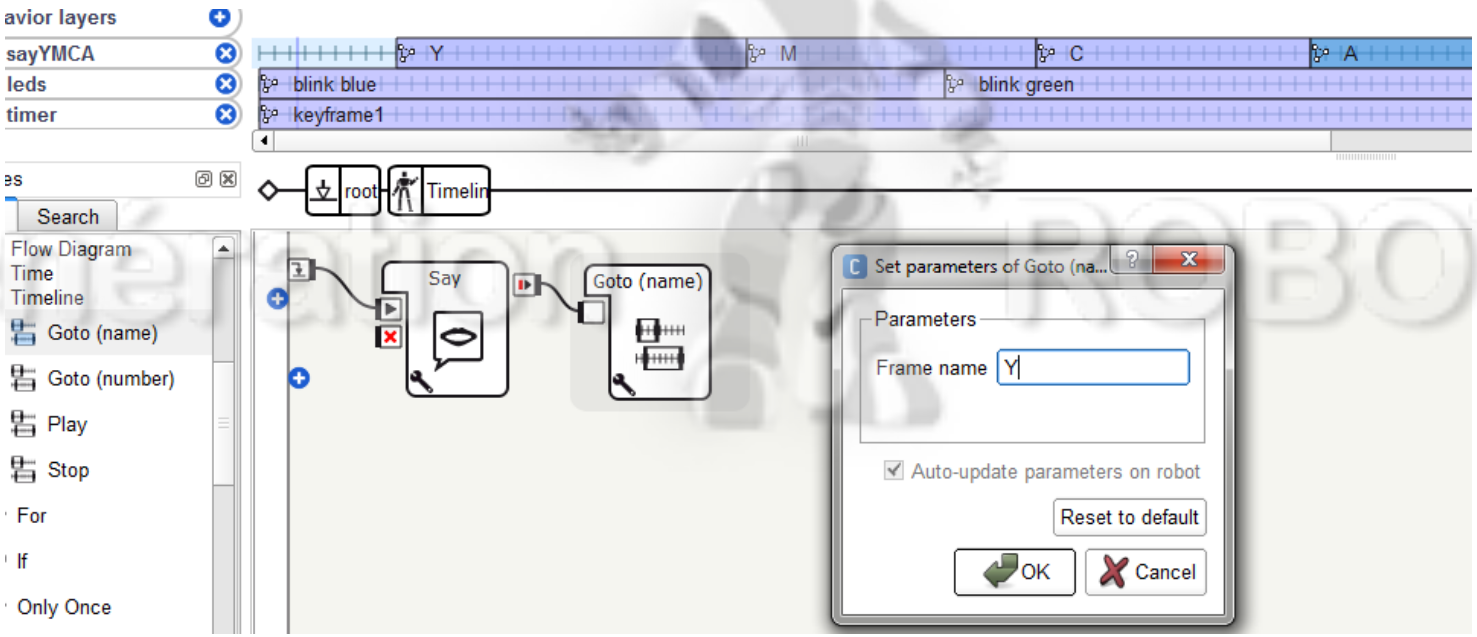


Integrate and synchronize behaviors
Control time with boxes

- You can also jump from one frame to another with 'goTo' boxes
  - Enter the keyframe name for the 'goTo (name)' box
  - Enter the frame number for the 'goTo (number)' box
- You can easily create a state machine
  - When you are at one frame, you know which keyframes are loaded and which are not
  - You don't even need anything in the motion part

Integrate and synchronize behaviors
Control time with boxes

- Put a Goto box at the end of the movement to loop
  - Careful not to ask for a too quick movement



Integrate and synchronize behaviors

Summary

- Behavior Layers
- Use Timeline as a State Machine
- Goto Boxes
- Ending a timeline
- Loading/Unloading keyframe
- onStopped effect of a behavior layer keyframe
- frozen FPS

Outline

Advanced notions
Manipulating Autonomous Life

How does Autonomous Life decides which application to launch ?

- Autonomous launchpad checks which activities can be launched
  - Based on people detection, trigger sentences...
- Interactive activities have priority over solitary ones
- Priority increases when the activity has not been played for long

Advanced notions
Manipulating Autonomous Life

How does Autonomous Life decides which application to launch ?

- In Choregraphe, use the Memory watcher
  - AutonomousLife/CompletedActivity : the last finished activity
  - AutonomousLife/FocusedActivity : the current activity
  - AutonomousLife/LaunchSuggestions : activities with launch requirements filled

Advanced notions
Manipulating Autonomous Life

Activity focus:

- When Autonomous Life launches an activity, this activity has the focus
- This activity can launch another activity
  - The first activity is stopped and may be 'stacked'
  - This second activity gets the focus
  - If the first activity was stacked, it restarts at the end of the second activity

Advanced notions
Manipulating Autonomous Life

```
int switchFocus(string activity_name, int flags)
```

- Allows to launch another activity
- If flag is STOP_CURRENT, the current activity is stopped (default, flags = 0)
- If flag is STOP_AND_STACK_CURRENT, the current activity is stopped, stacked and will be restarted when the other activity is finished (flags = 1)
- Max 64 stacked activities !

Advanced notions
Advanced Usage

Breathing (part of ALMotion module)

- Automatically restarted each time a switchFocus occurs
- Breathing can be stopped or modified
  - Can be applied to “Body”, “Legs”, “Arms”, “LArm”, “RArm” and “Head”
  - Default configuration is [[’Bpm’, 12], [’Amplitude’, 0.5]]
  - The highest allowed amplitude lowers, when the frequency is increased to avoid unstable movements

```
motionProxy.setBreathEnabled('Body', True)
motionProxy.setBreathConfig([['Bpm', 30], ['Amplitude',
    0.0]])
```

Advanced notions
Advanced Usage

Basic Awareness (part of ALBasicAwareness module)

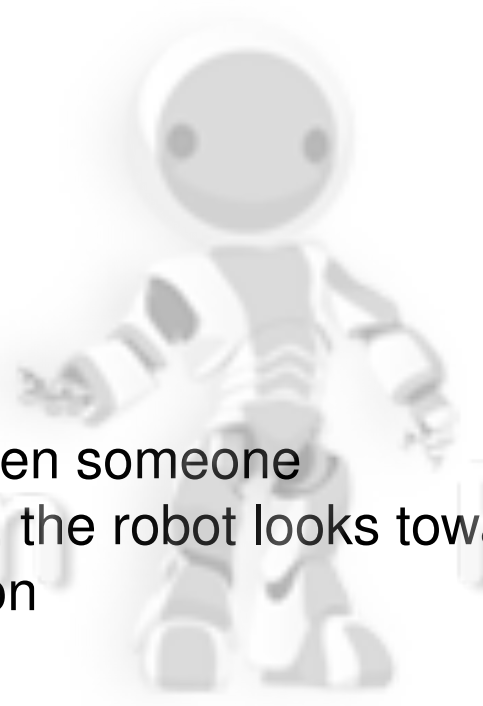
```
ba = session.service("ALBasicAwareness")
ba.setEnabled(True)  #start awareness
ba.setEnabled(False) #stop  awareness
bool running = ba.isEnabled()
```

- 4 types of stimuli
  - Sound, Movement, People, Touch

```
ba.setStimulusDetectionEnabled(stimName, enable)
bool enabled = ba.isStimulusDetectionEnabled(stimName)
```


SoftBank Robotics

Advanced notions
Advanced Usage



Basic Awareness states

- Unengaged
  - Looks for stimuli
- SemiEngaged
  - The robot has seen someone
  - If another stimuli, the robot looks towards the stimuli then back to the person
- FullyEngaged
  - The robot interacts with someone
  - It does not listen to stimuli

Advanced notions
Advanced Usage

Basic Awareness states

```
setEngagementMode(string modeName)
string modeName = getEngagementMode()
```

Events:

```
ALBasicAwareness/StimulusDetected
ALBasicAwareness/HumanTracked
ALBasicAwareness/HumanLost
```

Advanced notions
Advanced Usage

Expressive Listening (part of ALAutonomousMoves module)

```
setExpressiveListeningEnabled(bool enabled)
bool enabled = getExpressiveListeningEnabled()
```

Small displacements (part of ALAutonomousMoves module)

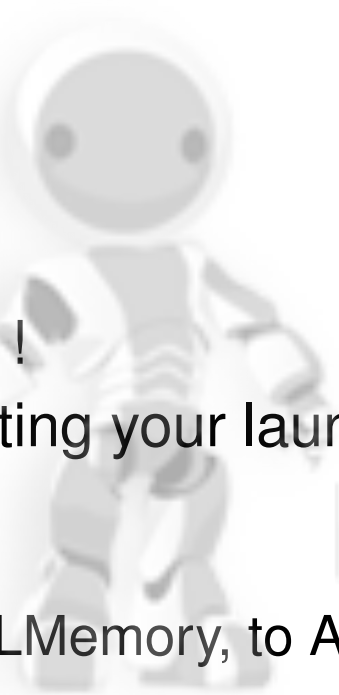
```
see Autonomous Abilities management
http://doc.aldebaran.com/2-4/ref/life/
    autonomous_abilities_management.html#autonomous-
    abilities-management
```

Advanced notions
Advanced Usage

Launchpad plugin creation

- You can create your own launch pad plugin
- It is a Choregraphe behavior that runs in background
- And create or update ALMemory values
- With the goal to lead Autonomous Life to launch some activities

Advanced notions
Advanced Usage



Launchpad plugin creation

- Only for advanced users !
- Be minimalist when creating your launchpad plugin !
  - No moving the robot
  - Be CPU friendly
  - Just a pipeline from ALMemory, to ALMemory

Advanced notions
Advanced Usage

Launchpad plugin creation

- Create a behavior with nature 'no nature'
- Listen to ALMemory event, process (not too much), insert ALMemory values
- Create an activity with launch conditions corresponding to your launchpad conditions

Advanced notions
Advanced Usage

Launchpad plugin creation

- Add autonomousLaunchpadPlugin.xml to your launchpad plugin
- Launch the application containing the plugin

"autonomousLaunchpadPlugin.xml"

```
<?xml version='1.0' encoding='UTF-8'?>
<autonomousLaunchpadPlugin>
  <groups>
    <group>solitary</group>
  </groups>
</autonomousLaunchpadPlugin>
```

Thank you for your attention !!
Questions, feedback, remarks...

SoftBank Robotics