

A white humanoid robot with a rounded head, two small eyes, and a friendly appearance. It has two articulated arms with white and grey segments, and a torso with a small screen. It is mounted on a circular base with orange and white segments. The robot is positioned in the center of the slide, behind the title text.

# **TIAGo Training Sessions**

## **Upper Body Motions**

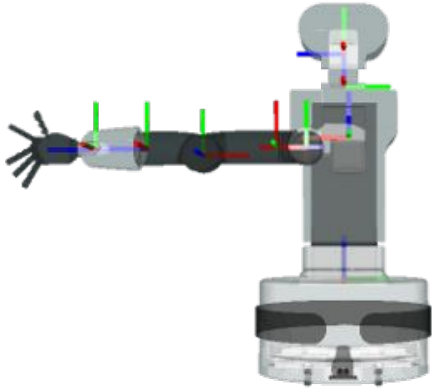
# Upper body motions



# Introduction



# Upper body joints



Joint	Type	Lower limit	Upper limit
torso	prismatic	0 mm	350 mm
head_1	revolute	-75°	75°
head_2	revolute	-60°	45°
arm_1	revolute	0	157.5°
arm_2	revolute	-90°	62.5°
arm_3	revolute	-202.5°	90°
arm_4	revolute	-22.5°	135°
arm_5	revolute	-120°	120°
arm_6	revolute	-90° (*)	90° (*)
arm_7	revolute	-120°	120°

(\*) Wrist version with force/torque sensor arm\_6 range is [-81°, 81°]

# Motion types

Type of motion	Interfaces	Joints supported
Playback of <b>prerecorded motions</b>	<ul style="list-style-type: none"><li>• Action Server</li></ul>	<ul style="list-style-type: none"><li>• torso</li><li>• arm</li><li>• hand / gripper</li><li>• head</li></ul>
<b>Joint trajectory</b>	<ul style="list-style-type: none"><li>• Topic</li><li>• Action Server</li></ul>	<ul style="list-style-type: none"><li>• torso</li><li>• arm</li><li>• hand / gripper</li><li>• head</li></ul>
<b>Position increments</b>	<ul style="list-style-type: none"><li>• Action Server</li></ul>	<ul style="list-style-type: none"><li>• torso</li><li>• head</li></ul>

# Motions executed by joystick







# Motion execution with joystick (I)



Buttons	Description	Motion type	Self-collisions check
LB	Torso upwards	position increment	✓
LT	Torso downwards	position increment	✓

# Motion execution with joystick (II)



Buttons	Description	Motion type	Self-collisions check
	Move head leftwards	Position increment	✓
	Move head rightwards	Position increment	✓
	Move head downwards	Position increment	✓
	Move head upwards	Position increment	✓



# Motion execution with joystick (III)



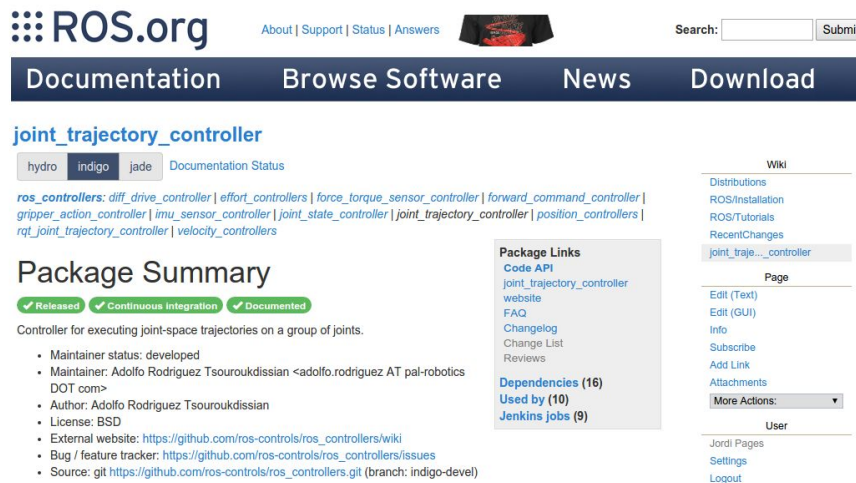
Buttons	Description
RB	Open hand / gripper
RT	Close hand / gripper

# Joint trajectory motions



# Joint trajectory motions

- Executing joint-space trajectories on a group of joints
- Underlying sytem: [joint\\_trajectory\\_controller](#) ROS package:



The screenshot shows the ROS.org website for the `joint_trajectory_controller` package. The page includes a navigation bar with links for Documentation, Browse Software, News, and Download. The package name is highlighted in blue. Below the name, there are tabs for different ROS versions: hydro, indigo, and jade. The Documentation Status is shown as a green checkmark. The Package Summary section describes the controller for executing joint-space trajectories on a group of joints. It lists the maintainer as Adolfo Rodriguez Tsouroukdissian, the author as Adolfo Rodriguez Tsouroukdissian, and the license as BSD. It also provides links to the external website, bug/feature tracker, and source code. The Package Links section includes links for Code, API, website, FAQ, Changelog, Change List, and Reviews. The Dependencies section lists 16 dependencies, and the Used by section lists 10 users. The Wiki section includes links for Distributions, ROS/Installation, ROS/Tutorials, Recent Changes, and joint\_trajectory\_controller. The Page section includes links for Edit (Text), Edit (GUI), Info, Subscribe, Add Link, Attachments, and More Actions. The User section includes links for Jordi Pages, Settings, and Logout.

ROS.org About | Support | Status | Answers

Search:  Submit

Documentation Browse Software News Download

[joint\\_trajectory\\_controller](#)

hydro indigo jade Documentation Status

*ros\_controllers: diff\_drive\_controller | effort\_controllers | force\_torque\_sensor\_controller | forward\_command\_controller | gripper\_action\_controller | imu\_sensor\_controller | joint\_state\_controller | joint\_trajectory\_controller | position\_controllers | rqt\_joint\_trajectory\_controller | velocity\_controllers*

### Package Summary

Released Continuous integration Documented

Controller for executing joint-space trajectories on a group of joints.

- Maintainer status: developed
- Maintainer: Adolfo Rodriguez Tsouroukdissian <adolfo.rodriguez AT pal-robotics DOT com>
- Author: Adolfo Rodriguez Tsouroukdissian
- License: BSD
- External website: [https://github.com/ros-controls/ros\\_controllers/wiki](https://github.com/ros-controls/ros_controllers/wiki)
- Bug / feature tracker: [https://github.com/ros-controls/ros\\_controllers/issues](https://github.com/ros-controls/ros_controllers/issues)
- Source: git [https://github.com/ros-controls/ros\\_controllers.git](https://github.com/ros-controls/ros_controllers.git) (branch: indigo-devel)

#### Package Links

Code API  
joint\_trajectory\_controller  
website  
FAQ  
Changelog  
Change List  
Reviews

#### Dependencies (16)

Used by (10)  
Jenkins jobs (9)

#### Wiki

Distributions  
ROS/Installation  
ROS/Tutorials  
Recent Changes  
joint\_trajectory\_controller

#### Page

Edit (Text)  
Edit (GUI)  
Info  
Subscribe  
Add Link  
Attachments  
More Actions:

#### User

Jordi Pages  
Settings  
Logout

# Joint trajectory motions specification (I)

- Trajectories are specified as a set of waypoints to be reached at specific time instants

trajectory\_msgs/JointTrajectory:

Header header  
string[] joint\_names  
JointTrajectoryPoint[] points

trajectory\_msgs/JointTrajectoryPoint:

# Each trajectory point specifies either positions[, velocities[, accelerations]]  
# or positions[, effort] for the trajectory to be executed.  
# All specified values are in the same order as the joint names in JointTrajectory.msg

float64[] positions  
float64[] velocities  
float64[] accelerations  
float64[] effort  
duration time\_from\_start

# Joint trajectory motions specification (II)

- Example of [trajectory\\_msgs/JointTrajectory](#) msg:

```
header:
  seq: 83
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
joint_names: ['head_1_joint', 'head_2_joint']
points:
  -
    positions: [-0.10506145631511613, -0.04555309342499999]
    velocities: []
    accelerations: []
    effort: []
    time_from_start:
      secs: 0
      nsecs: 1000000000
```

# Joint trajectory motions interfaces

- **Topic interfaces** ([trajectory\\_msgs/JointTrajectory](#))

/torso\_controller/command

/arm\_controller/command

/head\_controller/command

/hand\_controller/command

/gripper\_controller/command

/torso\_controller/safe\_command  
/arm\_controller/safe\_command

} executes motion only if it does not lead to a self-collision

- **Action interfaces** ([control\\_msgs::FollowJointTrajectoryAction](#))

/torso\_controller/follow\_joint\_trajectory

/head\_controller/follow\_joint\_trajectory

/arm\_controller/follow\_joint\_trajectory

/hand\_controller/follow\_joint\_trajectory

/gripper\_controller/follow\_joint\_trajectory

/safe\_arm\_controller/follow\_joint\_trajectory  
/safe\_torso\_controller/follow\_joint\_trajectory

} executes motion only if it does not lead to a self-collision

**trajectory\_msgs/JointTrajectory trajectory**

JointTolerance[] path\_tolerance

JointTolerance[] goal\_tolerance

duration goal\_time\_tolerance

Goal

---

int32 error\_code

string error\_string

Result

---

Header header

string[] joint\_names

trajectory\_msgs/JointTrajectoryPoint desired

trajectory\_msgs/JointTrajectoryPoint actual

trajectory\_msgs/JointTrajectoryPoint error

Feedback

# Joint trajectory motions execution (I)

- **Command line execution** (topic interface):

```
rostopic pub -1 /head_controller/command trajectory_msgs/JointTrajectory
"header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: "
joint_names: ['head_1_joint','head_2_joint']
points:
- positions: [-0.2, 0.5]
  velocities: []
  accelerations: []
  effort: []
  time_from_start: {secs: 2, nsecs: 0}"
```

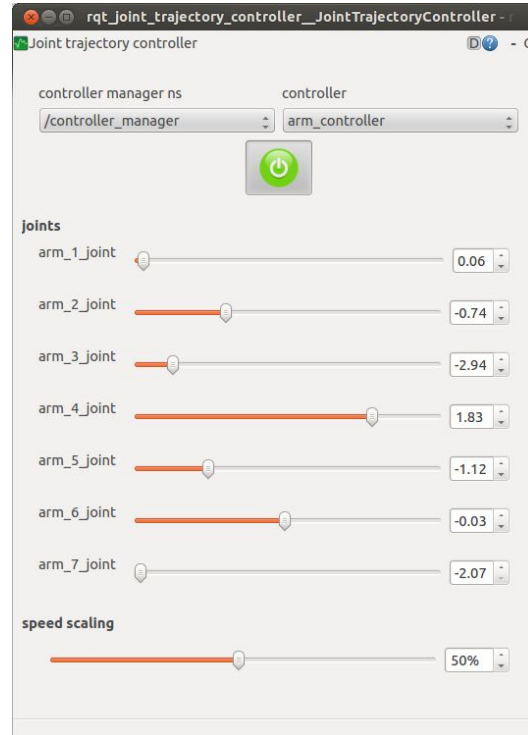
# Joint trajectory motions execution (II)

- **rqt GUI interface** (topic interface):

```
export ROS_IP=10.68.0.128
```

specify the IP of your computer

```
roslaunch rqt_joint_trajectory_controller rqt_joint_trajectory_controller
```



→ /arm\_controller/command

→ /torso\_controller/command

→ /head\_controller/command

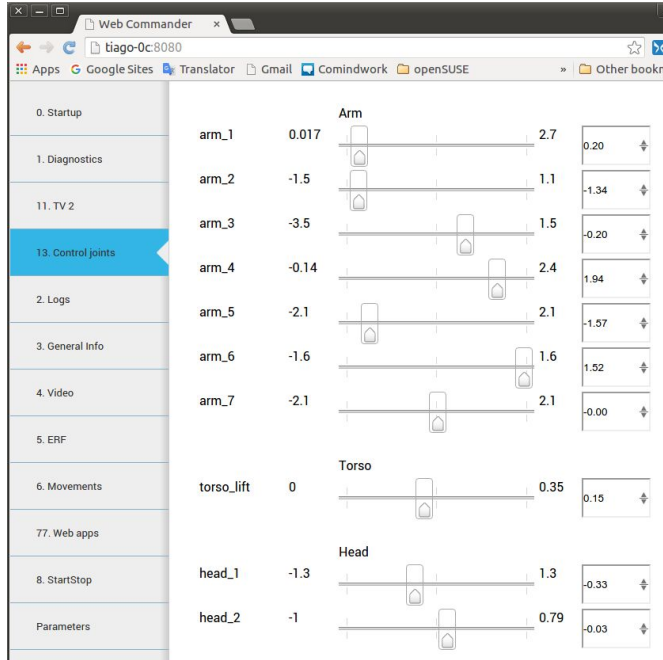
→ /gripper\_controller/command



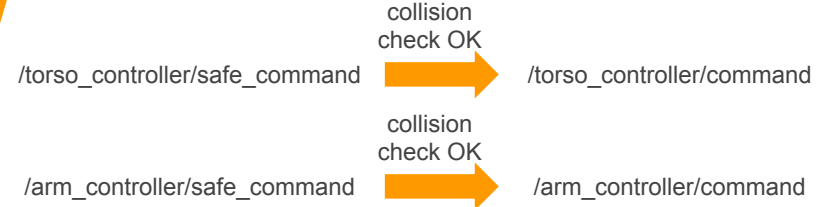


# Joint trajectory motions execution (III)

- Web server interface (topic interface) in the **Control joints** tab



For the **torso** and the **arm** the safe joint trajectory topic interfaces are used:



# Joint trajectory motions execution (IV)

- **move\_joint** (action interface)
- The node **move\_joint** in `play_motion` package can be used to move any individual joint of the upper body

- Examples:

```
roslaunch play_motion move_joint torso_lift_joint 0.18 2
```



**move\_joint** sends the required motion to the corresponding Action Server:  
*/torso\_controller/follow\_joint\_trajectory*

```
roslaunch play_motion move_joint head_2_joint 0.2 3
```



**move\_joint** sends the required motion to the corresponding Action Server:  
*/head\_controller/follow\_joint\_trajectory*

**WARNING:** this node does not check if the new position is in collision

# Predefined upper body motions



# Predefined motions

- Predefined motions involving a subset of these DoF can be played back at any time
- Underlying system for upper body predefined motions: **play\_motion** ROS package:



The screenshot shows the ROS.org website interface for the `play_motion` package. At the top, the ROS.org logo is followed by navigation links: About, Support, Status, and answers.ros.org. Below this is a dark blue navigation bar with buttons for Documentation, Browse Software, and News. The `play_motion` package page is displayed, with tabs for hydro and indigo (selected), and a link to Documentation Status. The Package Summary section includes status badges: Released, Continuous Integration, and Documented. It describes the package as playing a pre-recorded motion on a robot and lists maintainer, author, license, and source information. A sidebar on the right contains Package Links (Code API, FAQ, Changelog, Change List, Reviews) and Dependencies (8), with a link to Jenkins jobs (9). At the bottom, a link to the package's wiki page is provided.

ROS.org [About](#) | [Support](#) | [Status](#) | [answers.ros.org](#)

Documentation Browse Software News

`play_motion`

hydro indigo Documentation Status

### Package Summary

✓ Released ✓ Continuous Integration ✓ Documented

Plays a pre-recorded motion on a robot

- Maintainer status: developed
- Maintainer: Bence Magyar <bence.magyar AT pal-robotics DOT com>
- Author: Paul Mathieu <paul.mathieu AT pal-robotics DOT com>
- License: BSD
- Source: git [https://github.com/pal-robotics/play\\_motion.git](https://github.com/pal-robotics/play_motion.git) (branch: indigo-devel)

[http://wiki.ros.org/play\\_motion](http://wiki.ros.org/play_motion)

**Package Links**  
[Code API](#)  
[FAQ](#)  
[Changelog](#)  
[Change List](#)  
[Reviews](#)

**Dependencies (8)**  
[Jenkins jobs \(9\)](#)

# Predefined motions specification (I)

- **Motion specification** format

- *joints*: list of joints used in the motion
- *points*: list of robot states
  - *positions*: list of joint positions w.r.t. joints list
  - *time\_from\_start*: time given to reach the position
- *meta*: meta information that can be used by other applications

- Predefined motions:

tiago\_bringup/config/tiago\_motions.yaml

- home
- unfold\_arm
- reach\_floor
- reach\_max
- head\_tour
- wave
- pregrasp\_weight
- do\_weights
- pick\_from\_floor
- shake\_hands
- open\_hand
- close\_hand
- pointing\_hand
- gun\_hand
- thumb\_up\_hand
- pinch\_hand

# Predefined motions specification (II)

- Motion specification example:

```
rosparam get /play_motion/motions/wave -p
```

Web commander

<http://tiago-0c:8080>

6. Movements

Wave

```
joints: [arm_1_joint, arm_2_joint, arm_3_joint, arm_4_joint, arm_5_joint, arm_6_joint, arm_7_joint]
```

```
meta:
```

```
  description: wave
```

```
  name: Wave
```

```
  usage: demo
```

```
points: - positions: [0.06337464909724033, -0.679638896132783, -3.1087325315620733, 2.0882339360702575,  
                    -1.1201172410014792, -0.031008601325809293, -2.0744261217334135]
```

```
  time_from_start: 0.0
```

```
- positions: [0.06335930908588873, -0.7354151774072313, -2.939624246421942, 1.8341256735249563,  
            -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
```

```
  time_from_start: 1.0
```

```
- positions: [0.06335930908588873, -0.7231278283145929, -2.9385504456273295, 2.2121050027803877,  
            -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
```

```
  time_from_start: 2.0
```

```
- positions: [0.06335930908588873, -0.7354151774072313, -2.939624246421942, 1.8341256735249563,  
            -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
```

```
  time_from_start: 3.0
```



# Predefined motions: Action Server

- ROS topics: `rostopic list | grep -i play_motion`

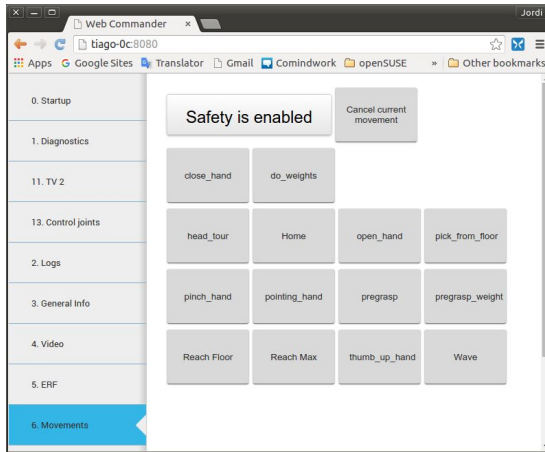
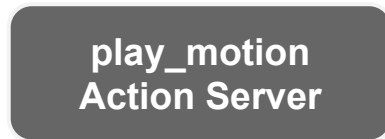
```
/play_motion/cancel  
/play_motion/feedback  
/play_motion/goal  
/play_motion/result  
/play_motion/status
```

- **play\_motion** action goal API:
  - **motion\_name**  
name from the motion definition file
  - **skip\_planning**  
when true, skip motion planning in execution
  - **priority**  
unimplemented, will serve to guide preemption policy
- **Preemption policy**
  - While a goal is active, reject all incoming goals
  - Unlike typical action servers (new goal preempts active one)

# Predefined motions: Action Clients



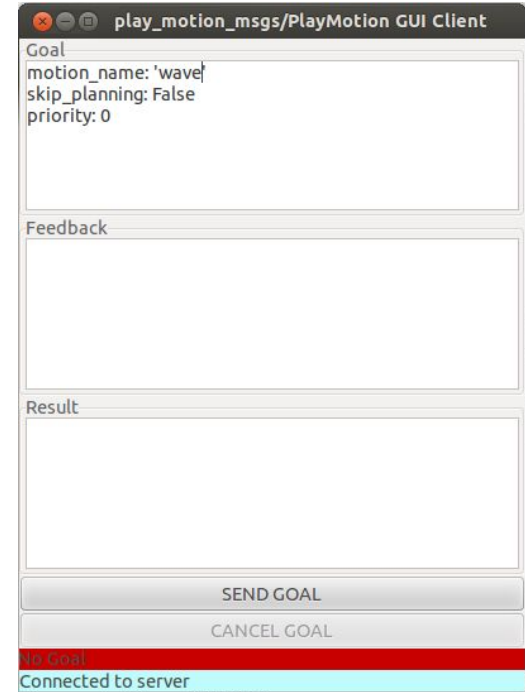
joystick



Web commander



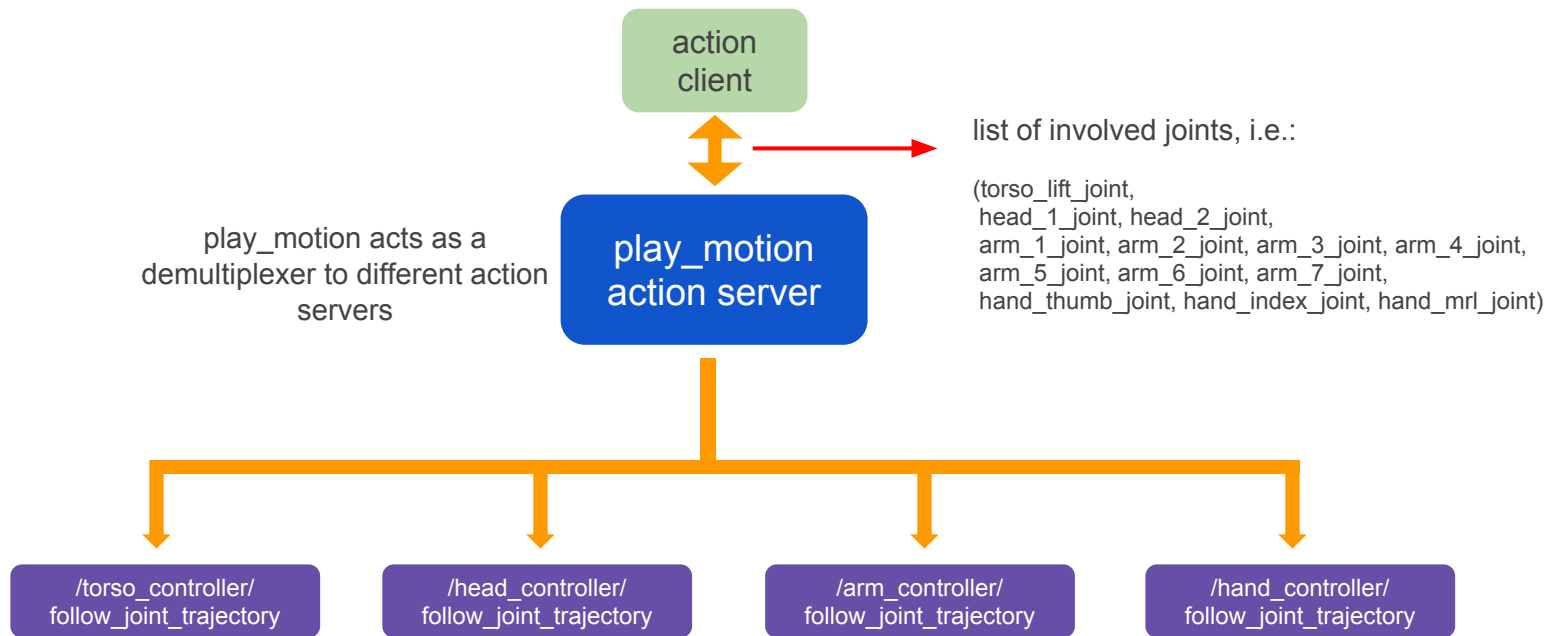
tiago\_tutorials



roslaunch actionlib axclient.py /play\_motion



# How does play\_motion work?



# Predefined motions execution (I)

- GUI interface: your computer IP

ROS\_IP=10.68.0.128

roslaunch actionlib axclient.py /play\_motion

- 1) Make sure that the client gets connected to the server

play\_motion\_msgs/PlayMotion GUI Client

Goal  
motion\_name: 'wave'  
skip\_planning: False  
priority: 0

Feedback

Result

SEND GOAL  
CANCEL GOAL

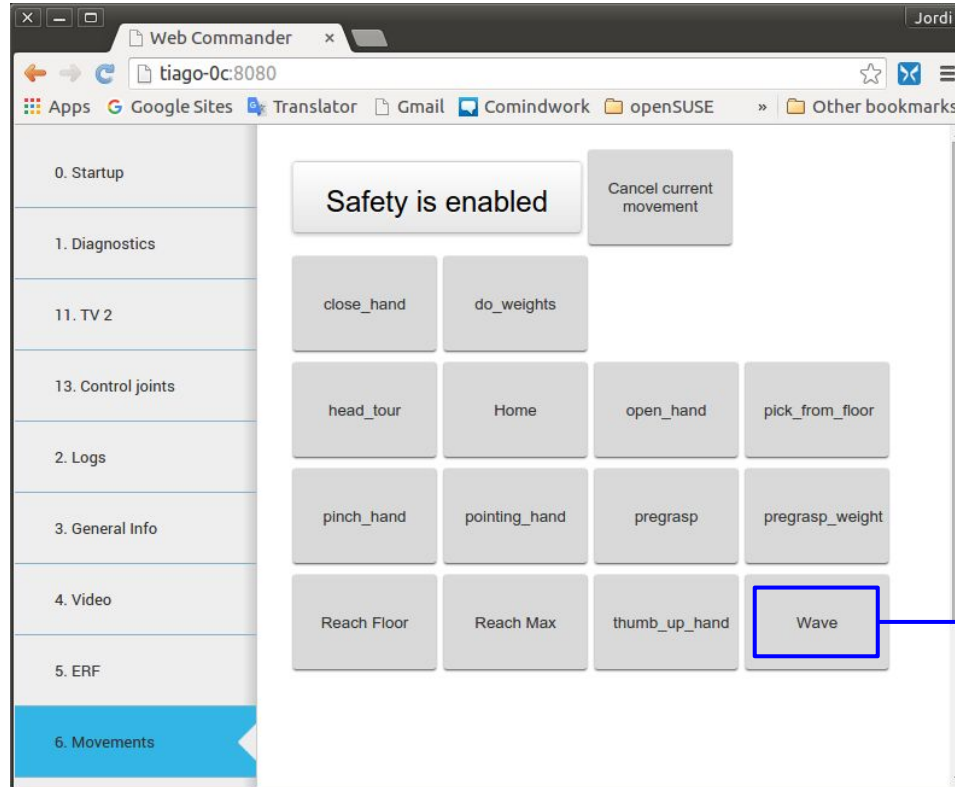
Connected to server

- 2) Fill in the name of the motion

- 3) Press button to send the goal

# Predefined motions execution (II)

- Web interface



# Assigning motions to the Web interface

```
gedit `rospack find tiago_bringup`/config/tiago_motions.yaml
```

```
play_motion:  
  controllers: [arm_controller, head_controller, torso_controller, hand_controller]  
  motions:  
    nod:
```

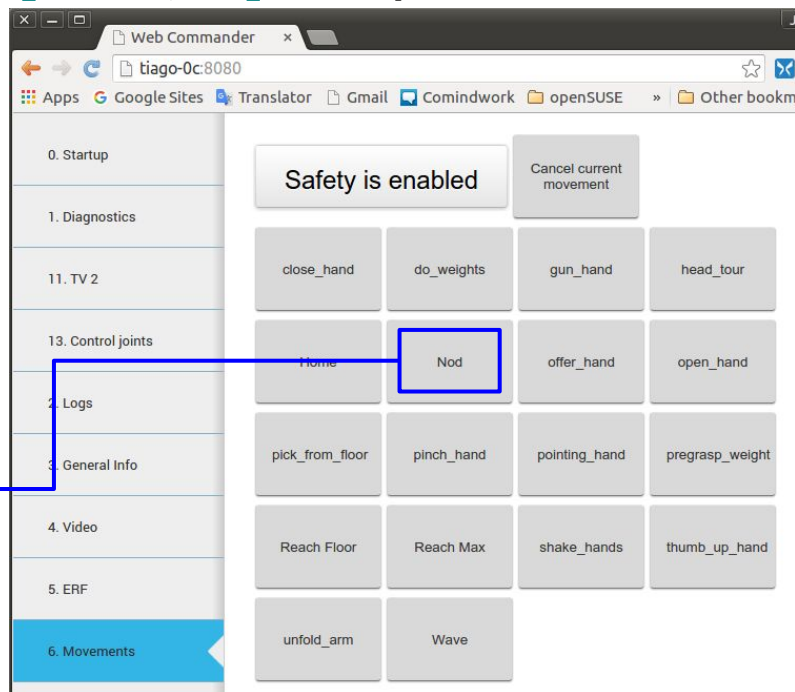
```
      joints: [head_1_joint, head_2_joint]
```

```
      points:
```

- positions: [0.0, 0.0]  
time\_from\_start: 2.0
- positions: [0.0, 0.5]  
time\_from\_start: 3.5
- positions: [0.0, -0.10]  
time\_from\_start: 5.5
- positions: [0.0, 0.0]  
time\_from\_start: 7.0

```
meta:
```

```
  name: Nod  
  usage: demo  
  description: 'nod'
```



# Joystick motion triggers



# Joystick upper body motion triggers

- Motion trigger example:

```
rosparam get /teleop/head_up -p
```

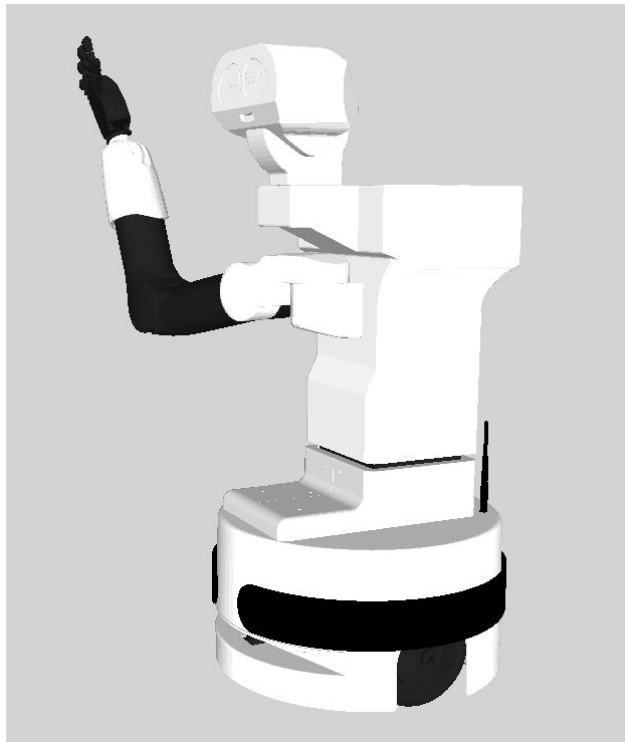
```
action_goal:  
  increment_by: [0.2, 0]  
action_name: /head_controller/increment  
buttons: [0]  
type: action
```

```
rosparam get /teleop/open_hand -p
```

```
action_goal:  
  motion_name: open_hand  
  skip_planning: true  
action_name: /play_motion  
buttons: [5]  
type: action
```

Default joystick triggers location:

tiago\_bringup/config/joy\_teleop.yaml



# Questions?

