TIAGo Training Sessions

Upper Body Motions

PAL ROBOTICS

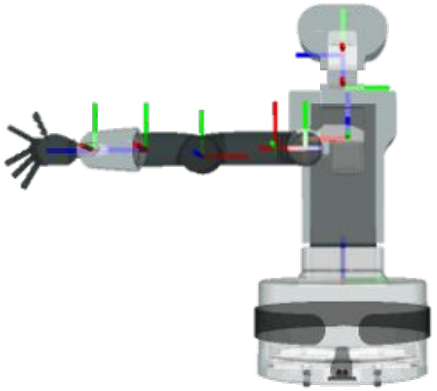
Upper body motions

TIAGo T.S.

PAL ROBOTICS

Introduction

Upper body joints



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

(*) Wrist version with force/torque sensor arm_6 range is [-81º, 81º]

Motion types

Type of motion	Interfaces	Joints supported
Playback of **prerecorded motions**	• Action Server	• torso • arm • hand / gripper • head
Joint trajectory	• Topic • Action Server	• torso • arm • hand / gripper • head
Position increments	• Action Server	• torso • head

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:

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
- Bug / feature tracker: https://github.com/ros-controls/ros_controllers/issues
- Source: 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
RecentChanges
joint_traje..._controller

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

User
Jordi Pages
Settings
Logout

PAL ROBOTICS

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: 100000000
```

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

Goal

```
trajectory_msgs/JointTrajectory trajectory
JointTolerance[] path_tolerance
JointTolerance[] goal_tolerance
duration goal_time_tolerance
---
```

Result

```
int32 error_code
string error_string
---
```

Feedback

```
Header header
string[] joint_names
trajectory_msgs/JointTrajectoryPoint desired
trajectory_msgs/JointTrajectoryPoint actual
trajectory_msgs/JointTrajectoryPoint error
```

Joint trajectory motions execution (I)

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

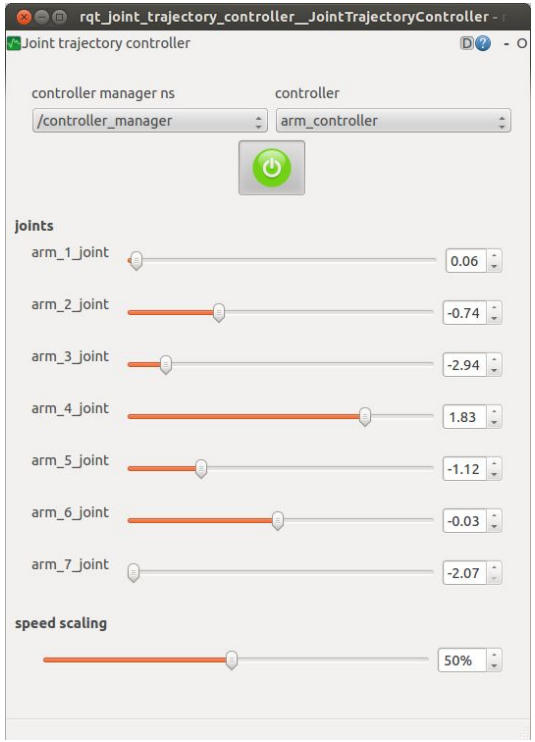
```
rostopic pub -1 /head_controller/command trajectory_msgs/JointTrajectory
"header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
joint_names: ['head_1_joint','head_2_joint']
points:
- positions: [-0.2, 0.5]
  velocities: []
  accelerations: []
  effort: []
  time_from_start: {secs: 2,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

Joint trajectory motions execution (II)

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

```
export ROS_IP=10.68.0.128
rosrun rqt_joint_trajectory_controller rqt_joint_trajectory_controller
```

specify the IP of your computer



/arm_controller/command

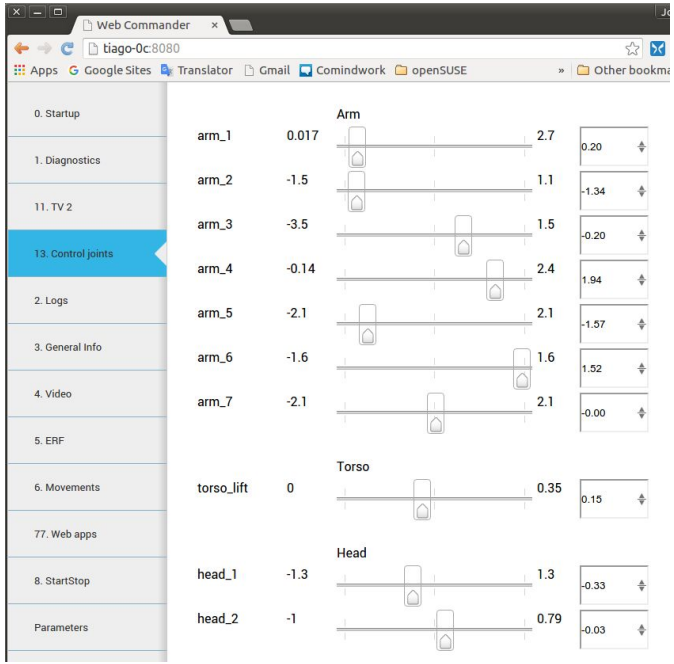
/torso_controller/command

/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:

/torso_controller/safe_command → collision check OK → /torso_controller/command

/arm_controller/safe_command → collision check OK → /arm_controller/command

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:

```
rosrun 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*

```
rosrun 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:

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 (branch: indigo-devel)

Package Links
- Code API
- FAQ
- Changelog
- Change List
- Reviews

Dependencies (8)
Jenkins jobs (9)

http://wiki.ros.org/play_motion

PAL ROBOTICS

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

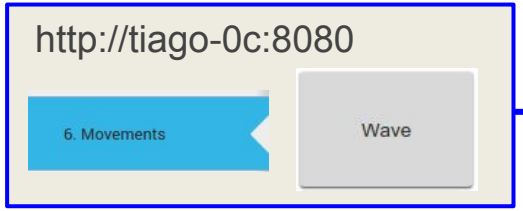
  - home
  - unfold_arm
  - reach_floor
  - reach_max
  - head_tour
  - wave
  - pregrasp_weight
  - do_weights
  - pick_from_floor
  - shake_hands
  - open_hand
  - close_hand
  - pointing_hand
  - gun_hand
  - thumb_up_hand
  - pinch_hand

Predefined motions specification (II)

- Motion specification example:

rosparam get /play_motion/motions/wave -p

Web commander




```
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
```

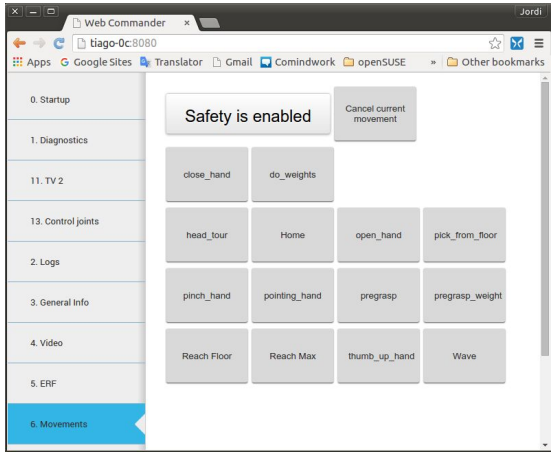
- **play_motion** action goal API:
  - **motion_name**
    name from the motion definition file
  - **skip_planning**
    when true, skip motion planning in execution
  - **priority**
    unimplemented, will serve to guide preemption policy

- **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

play_motion
Action Server

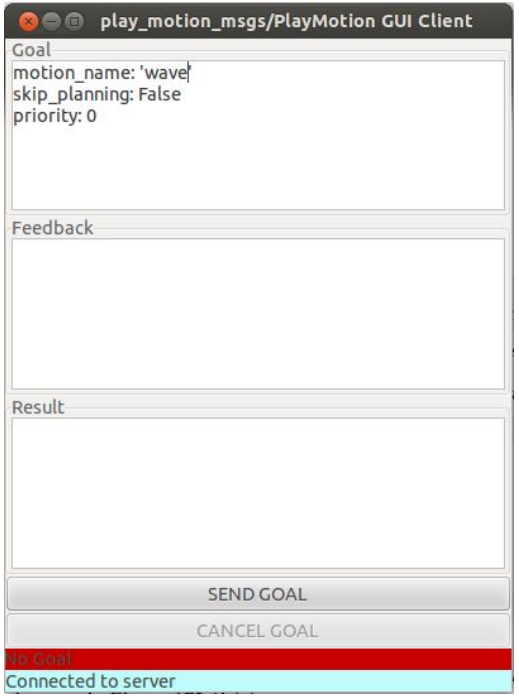


Web commander

Python
Action
Client

C++ Action
Client

tiago_tutorials

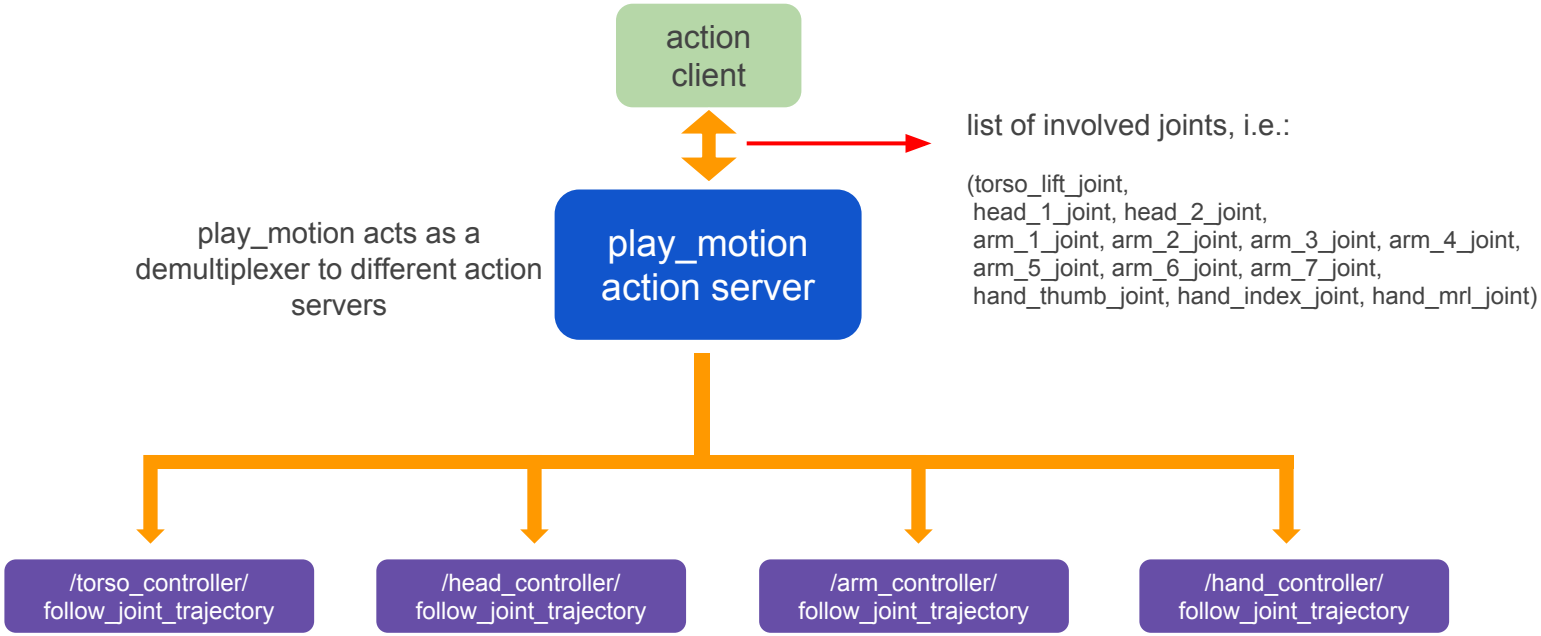


rosrun actionlib axclient.py /play_motion

TIAGo T.S.

PAL ROBOTICS

How does play_motion work?



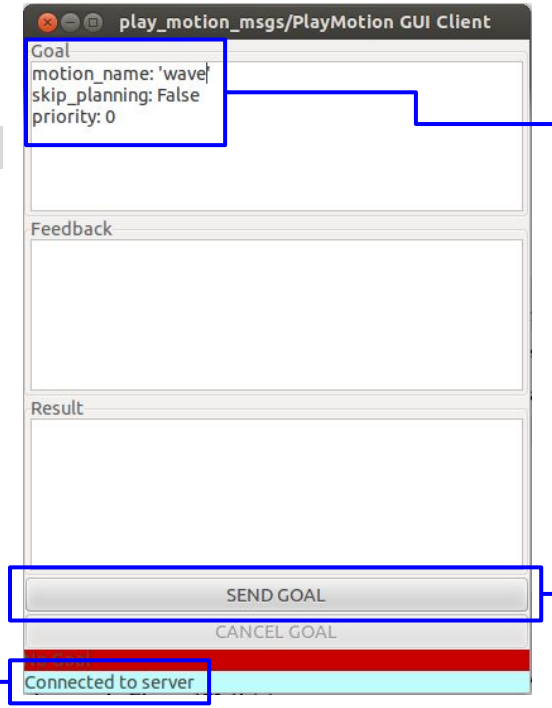
Predefined motions execution (I)

- GUI interface:

your computer IP

ROS_IP=10.68.0.128

rosrun actionlib axclient.py /play_motion



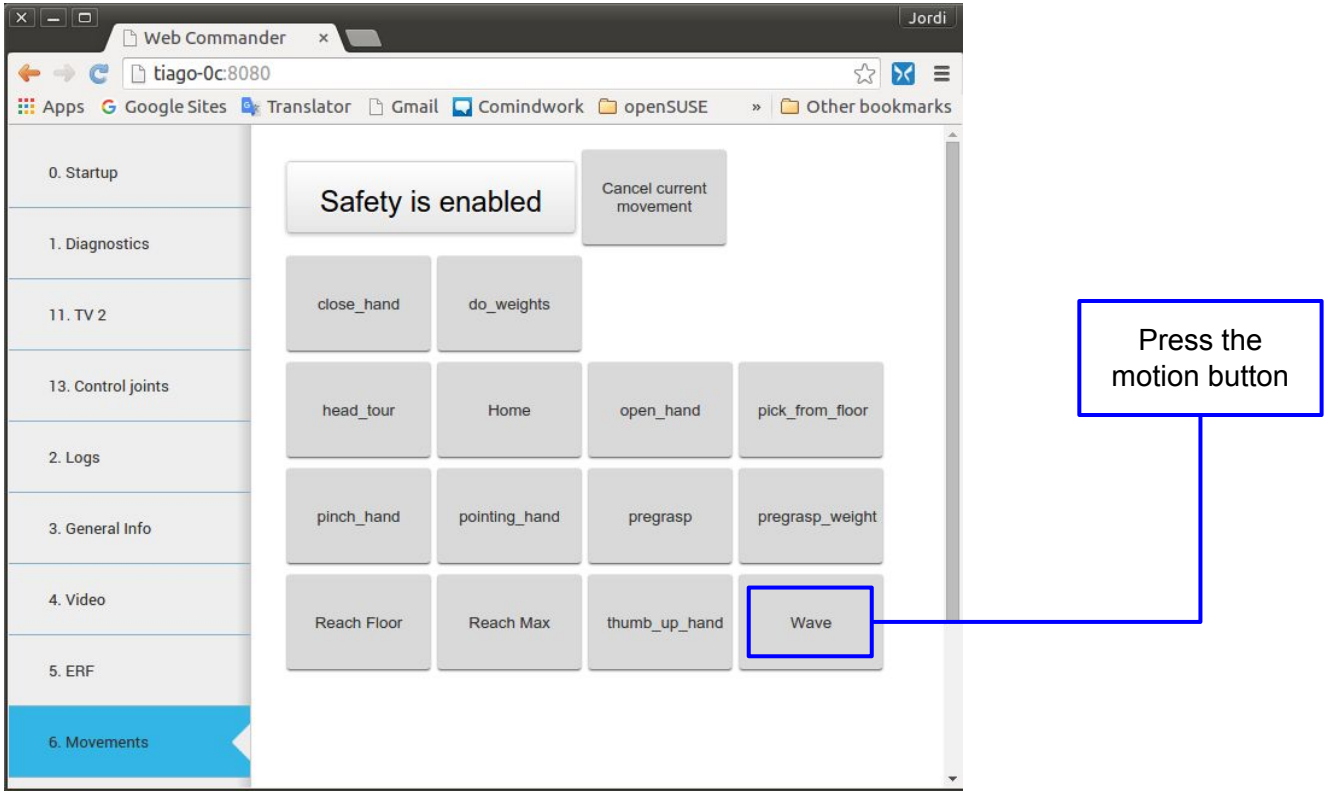
1) Make sure that the client gets connected to the server

2) Fill in the name of the motion

3) Press button to send the goal

Predefined motions execution (II)

- Web interface

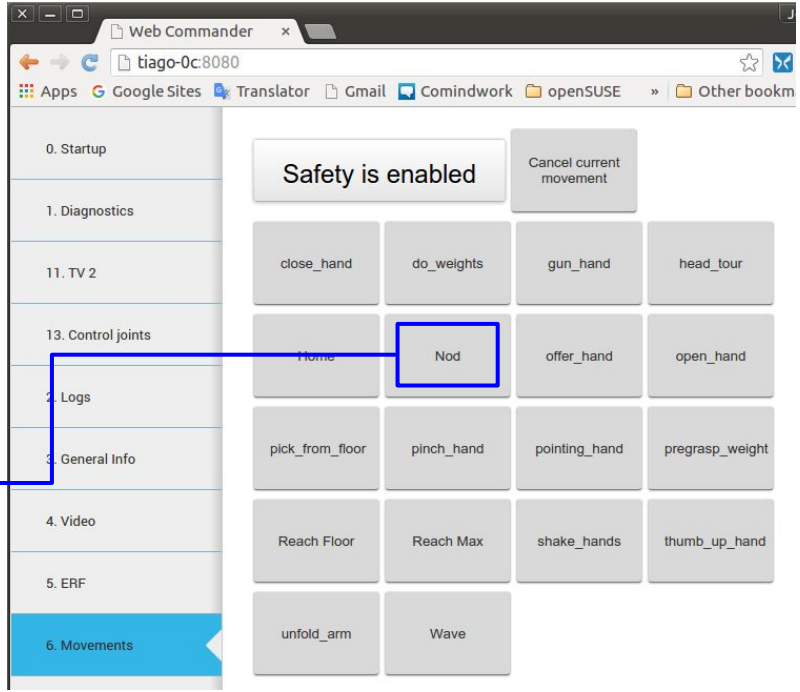


PAL ROBOTICS

Assigning motions to the Web interface

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

```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'
```



PAL ROBOTICS

Joystick motion triggers

Joystick upper body motion triggers

- Motion trigger example:

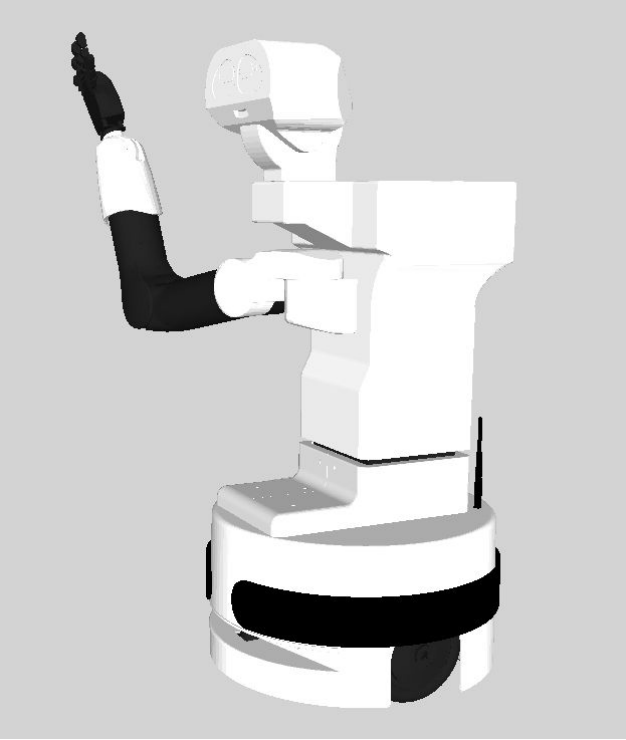
rosparam get /teleop/head_up -p

```
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?