jsk_robot Documentation

Release latest

Jan 09, 2022

CONTENTS

jsk_robot is a stack for the packages which are used in JSK lab.

The code is open source, and available on github.

This repository contains following ros packages:

jsk_robot

[!\[Build Status](https://travis-ci.com/jsk-ros-pkg/jsk_robot.svg){]}(https://travis-ci.com/jsk-ros-pkg/jsk_robot)

Deb Build Status

[//]: # (!!DO NOT EDIT !!)

[//]: # (THIS SECTION IS AUTOMATICALLY GENERATED BY)

[//]: # (rosrun jsk_tools generate_deb_status_table.py jsk_robot)

Package | Kinetic (Xenial) | Melodic (Bionic) |

|:------------------|:——— | | jsk_robot (arm64) | [!\[Build Status](http://build.ros.org/job/Kbin_uxv8_uXv8__jsk_robot__ubuntu_xenial_arm64__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uxv8_uXv8__jsk_robot__ubuntu_xenial_arm64__binary) | [!\[Build Status](http://build.ros.org/job/Mbin_ubv8_uBv8__jsk_robot__ubuntu_bionic_arm64__binary/badge/icon){]}(http://build.ros.org/job/Mbin_ubv8_uBv8__jsk_robot__ubuntu_bionic_arm64__binary) | | jsk_robot (armhf) | [!\[Build Status](http://build.ros.org/job/Kbin_uxhf_uXhf__jsk_robot__ubuntu_xenial_armhf__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uxhf_uXhf__jsk_robot__ubuntu_xenial_armhf__binary) | [!\[Build Status](http://build.ros.org/job/Mbin_ubhf_uBhf__jsk_robot__ubuntu_bionic_armhf__binary/badge/icon){]}(http://build.ros.org/job/Mbin_ubhf_uBhf__jsk_robot__ubuntu_bionic_armhf__binary) | | jsk_robot (i386) | [!\[Build Status](http://build.ros.org/job/Kbin_uX32__jsk_robot__ubuntu_xenial_i386__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uX32__jsk_robot__ubuntu_xenial_i386__binary) | — | | jsk_robot (amd64) | [!\[Build Status](http://build.ros.org/job/Kbin_uX64__jsk_robot__ubuntu_xenial_amd64__binary/badge/icon){]}(http://build.ros.org/job/Kbin_uX64__jsk_robot__ubuntu_xenial_amd64__binary) | [!\[Build Status](http://build.ros.org/job/Mbin_uB64__jsk_robot__ubuntu_bionic_amd64__binary/badge/icon){]}(http://build.ros.org/job/Mbin_uB64__jsk_robot__ubuntu_bionic_amd64__binary) |

[//]: #

jsk_robot_startup ===

lifelog

see lifelog/README.md

scripts/email_topic.py

This node sends email based on received rostopic (jsk_robot_startup/Email). Default values can be set by using *~email_info* There is [a client library](./euslisp/email-topic-client.l) and [sample program](./euslisp/sample-email-topic-client.l). If you want to see a demo. Please configure a smtp server and setup your email_info yaml at /var/lib/robot/email_info.yaml and run.

`bash roslaunch jsk_robot_startup sample_email_topic.launch receiver_address:=<a mail address to send a mail to> `

Parameters

- *~email_info* (type: *String*, default: */var/lib/robot/email_info.yaml*)

Default values of email configuration. Example of a yaml file is below.

```yaml subject: hello body: world sender_address: hoge@test.com receiver_address: fuga@test.com smtp_server: test.com smtp_port: 25 attached_files:

- /home/user/Pictures/test.png

```

Subscriber

- *email* (type: *jsk_robot_startup/Email*)

Subscriber of email command.

scripts/ConstantHeightFramePublisher.py ![pointcloud_to_scan_base_tf_squat.png](images/pointcloud_to_scan_base_tf_squat.png) ![pointcloud_to_scan_base_tf_stand.png](images/pointcloud_to_scan_base_tf_stand.png)

This script provides a constant height frame from the ground to get a imagenary laser scan for pointcloud_to_laserscan package. Biped robots need to use this constant frame to get constant laser scan for 2D SLAM package for wheeled ones like gmapping, because the pose of biped robots including height of the base link changes during a task in contrast to wheeled ones. In this frame, x, y and yaw is same as base frame of the robot body, z is constant and roll and pitch is same as the ground.

Parameters

- *~parent_frame* (String, default: "BODY")

  This parameter indicates the parent frame of the constant height frame, which is expected to be a base frame of the robot body.

- *~odom_frame* (String, default: "odom")

  This parameter indicates the odometry frame on the ground.

- *~frame_name* (String, default: "pointcloud_to_scan_base")

  This parameter indicates the name of the constant frame.

- *~rate* (Double, default: 10.0)

  This parameter indicates publish rate [Hz] of the constant frame.

- *~height* (Double, default: 1.0)

  This parameter indicates initial height [m] of the constant frame.

Subscribing Topics

- *~height* (*std_msgs/Float64*)

  This topic modifies height [m] of the constant frame.

util/initialpose_publisher.l

This script sets initial pose with relative pose from specified TF frame by publishing */initialpose*.

Parameters

- *~transform_base* (String, default: "map")

  TF frame of publishing topic */initialpose*.

- *~transform_frame* (String, default: "eng2/7f/73B2")

  Base TF frame to calcurate relative initial pose

- *~initial_pose_x* (Double, default: 0.0)

  Relative pose x

- *~initial_pose_y* (Double, default: 0.0)

  Relative pose y

- *~initial_pose_yaw* (Double, default: 0.0)

  Relative pose yaw

Subscribing Topics

- */amcl_pose* (*geometry_msgs/PoseWithcovariancestamped*)

util/mux_selector.py

This node check and select mux input topic on condition of the specified topic. This node takes three arguments for one topic. The first one is the topic to be monitored. When a message from this topic is received, it is assigned as a variable *m*. If a condition specified as the second argument, this node calls a service to select the topic specified as the third argument.

Usage

```
` rosrun jsk_robot_startup mux_selector.py /joy1 'm.buttons[9]==1' /cmd_vel1 /joy2 'm.
buttons[9]==1' /cmd_vel2 `
```

Parameters

- *~patient* (Double, default: 0.5)

  Indicates the allowable range of the difference between the received topic time and the current time.

- *~frequency* (Double, default: 20.0)

  Frequency of processing loop.

- *~default_select* (String, default: *None*)

  Default topic name.

- *~wait* (Bool, default: *False*)

  If wait is *True*, this node waits for the topic to be received.

Subscribing Topics

The topic specified in the argument is subscribed.

scripts/check_room_light.py

This node publish the luminance calculated from input image and room light status.

Subscribing Topics

- *~input* (*sensor_msgs/Image* or *sensor_msgs/CompressedImage*)

  Input topic image

Publishing Topics

- *~output* (*jsk_robot_startup/RoomLight*)

  Room light status and room luminance

Parameters

- *~luminance_threshold* (Float, default: 50)

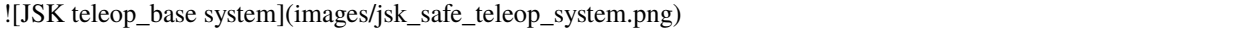
  Luminance threshold to deteremine whether room light is on or off

- *~image_transport* (String, default: raw)

  Image transport hint.

launch/safe_teleop.launch

This launch file provides a set of nodes for safe teleoperation common to mobile robots. Robot-specific nodes such as */joy*, */teleop* or */cable_warning* must be included in the teleop launch file for each robot, such as [safe_teleop.xml for PR2](https://github.com/jsk-ros-pkg/jsk_robot/blob/master/jsk_pr2_robot/jsk_pr2_startup/jsk_pr2_move_base/safe_teleop.xml) or [safe_teleop.xml for fetch](https://github.com/jsk-ros-pkg/jsk_robot/blob/master/jsk_fetch_robot/jsk_fetch_startup/launch/fetch_teleop.xml).



CHAPTER

ONE

JSK_PR2_ROBOT

Teleoperation

For the JSK safe teleop system, please see [data flow diagram of safe_teleop.launch](https://github.com/jsk-ros-pkg/jsk_robot/tree/master/jsk_robot_common/jsk_robot_startup#launchsafe_teleoplaunch)

![teleop_command](images/pr2_teleop_command.png)

Setup for Development Users ` mkdir -p catkin_ws/my_first_demo cd catkin_ws/my_first_demo wstool init src wstool set jsk_demos https://github.com/jsk-ros-pkg/jsk_demos -t src --git wstool update -t src source ~applications/ros/hydro/devel/setup.bash catkin b `

Setup for Application Users (for administrator only)

use [jsk_pr2.rosinstall](https://github.com/jsk-ros-pkg/jsk_robot/blob/master/jsk_pr2_robot/jsk_pr2_startup/jsk_pr2.rosinstall) to install software ` mkdir -p ros/hydro/src cd ros/hydro wstool init src git clone https://github.com/jsk-ros-pkg/jsk_robot.git src/jsk-ros-pkg/jsk_robot wget -O src/.rosinstall https://raw.githubusercontent.com/jsk-ros-pkg/jsk_robot/master/jsk_pr2_robot/jsk_pr2_startup/jsk_pr2.rosinstall wstool update -t src rosdep install --from-paths src --ignore-src -r -y catkin b `

jsk_pr2_startup

setup

###. rewrite */etc/ros/robot.launch*

Please rewrite */etc/ros/robot.launch* like following: ```xml <launch>

<!– Robot Description –> <param name=”robot_description” textfile=”/etc/ros/groovy/urdf/robot.xml” />

<!– Robot Analyzer –> <rosparam command=”load” file=”$(find pr2_bringup)/config/pr2_analyzers.yaml” ns=”diag_agg” />

<!– Robot bringup –> <include file=”$(find jsk_pr2_startup)/pr2_bringup.launch” /> <!– <group> –> <!– <remap from=”/joy” to=”/joy_org”/> –> <!– <include file=”$(find pr2_bringup)/pr2.launch” /> –> <!– </group> –>

<!– Web ui –> <!– include file=”$(find webui)/webui.launch” /> –>

<!– Android app –> <include file=”$(find local_app_manager)/app_manager.launch” > <arg name=”ROBOT_NAME” value=”pr1012” /> <arg name=”ROBOT_TYPE” value=”pr2” />

</include>

<!– RobotWebTools –> <include file=”$(find rwt_image_view)/launch/rwt_image_view.launch”/>

<!– kinect –> <include file=”$(find jsk_pr2_startup)/jsk_pr2_sensors/kinect_head.launch”>

<arg name=”respawn” value=”false” />

</include> <rosparam file="/etc/ros/robot.yaml"/>

</launch>

```

### launch mongodb for multiple users

Different users in same unix group can't run mongod against single db owned by that group. This is because *mongod* opens database files using the *O_NOATIME* flag to the open system call. Open with *O_NOATIME* only works if the UID completelly matchs or the caller is priviledged (*CAP_FOWNER*) for security reasons. So if you want to launch mongodb with shared database resouces, it's better to use POSIX Capabilities in Linux.

`bash # In Ubuntu $ sudo aptitude install libcap2-bin $ sudo setcap cap_fowner+ep /usr/bin/mongod `

### Hark with Microcone

#### documentation - Hark installation: http://www.hark.jp/wiki.cgi?page=HARK+Installation+Instructions - hark jsk installation: https://github.com/jsk-ros-pkg/jsk_3rdparty/blob/master/hark_jsk_plugins/INSTALL - Microcone: http://www.hark.jp/wiki.cgi?page=SupportedHardware#p10

# jsk_naoqi_robot

JSK original ROS package for NAO and Pepper. The package name comes from Naoqi OS they use.

## How to start up ROS nodes for a naoqi robot?

![](doc/img/relationship_of_robot_and_PC.png)

Your PC becomes ROS master. Your PC connects to a naoqi robot and starts up ROS nodes (*jsk_nao_startup.launch* and *jsk_pepper_startup.launch*). You can control NAO and Pepper via roseus (*naoeus* and *peppereus*). For more information about these programs, please refer to [here for NAO](https://github.com/jsk-ros-pkg/jsk_robot/tree/master/jsk_naoqi_robot#nao) and [here for Pepper](https://github.com/jsk-ros-pkg/jsk_robot/tree/master/jsk_naoqi_robot#pepper).

### How to turn on/off a naoqi robot?

- On

Please refer to [NAO's page](http://doc.aldebaran.com/2-1/nao/nao-turn-on.html) and [Pepper's page](http://doc.aldebaran.com/2-4/family/pepper_user_guide/turn_on_pep.html).

- Off

Please refer to [NAO's page](http://doc.aldebaran.com/2-1/nao/nao-turn-off.html) and [Pepper's page](http://doc.aldebaran.com/2-4/family/pepper_user_guide/turn_off_pep.html).

- Disable AutonomousLife

Naoqi robot has [AutonomousLife](http://doc.aldebaran.com/2-4/family/pepper_user_guide/life_pep.html) in addition to a normal concept of servo on and off.

When you're a developer, you'll want to disable AutonomousLife (it includes servo off) and servo on a robot to try codes you write.

If you want to know how to disable it, please refer to [using a chest button](http://doc.aldebaran.com/2-4/family/pepper_user_guide/freeze_pep.html). Link is for Pepper, but same as NAO.

Or please refer to [using ROS service](doc/disable_autonomous_life_from_ros_service.md).

### How to connect your PC and a robot to a same network?

- Network with DHCP

To connect NAO and Pepper to wifi for the first time, please refer to [here](doc/connect_to_wifi.md).
```

- Network without DHCP (link-local addressing)

Please refer to [here](http://doc.aldebaran.com/2-1/nao/connectivity.html#local-link-an-alternative-to-dhcp).

[2019.03.01: Trouble shooting]

When you connect Pepper and your PC via network without DHCP, power on Pepper and launch Setting from Pepper's tablet, setting wizard sometimes becomes zombie process. You may not exit Setting as described [here](https://github.com/jsk-ros-pkg/jsk_robot/blob/master/jsk_naoqi_robot/doc/connect_to_wifi.md#pepper-only-how-to-access-to-a-robot-web-page-via-peppers-tablet), which causes a failure of AutonomousLife setting.

` [ERROR] [1550641271.575637]: Exception while disabling life state: ALAutonomousLife::setState AutonomousLife::setState Calls to the setState method are not currently allowed. Did you finish the getting started wizard? `

If this happens, please connect Pepper to network with DHCP and exit Setting.

Setup Environment

% First, you need to install ros. For ros kinetic, please refer to install guide like [here](http://wiki.ros.org/kinetic/Installation). For ros melodic, please refer to install guide like [here](http://wiki.ros.org/melodic/Installation).

1. Install `Python NAOqi SDK` You can download it (version = 2.5.5) from [here](https://drive.google.com/file/d/1xHuYREDa78xGiikEpsjxfZQ7Gfvo1E9D/view?usp=sharing). Please unzip the downloaded file. Please create `pynaoqi` folder in your home directory. Then put the file under your `pynaoqi` folder.

% You can download other version SDKs from [here](https://www.softbankrobotics.com/emea/en/support/nao-6/downloads-softwares/former-versions?os=49&category=39). Please change the tab to SDKs. Version < 2.5.5 may cause error.

2. Export environment variables in your `.bashrc`

``` # Please use Python NAOqi SDK version >= 2.5.5 (https://github.com/jsk-ros-pkg/jsk_robot/issues/1099) export PYTHONPATH=$HOME/pynaoqi/pynaoqi-python2.7-2.5.5.5-linux64/lib/python2.7/site-packages:$PYTHONPATH

export NAO_IP="olive.jsk.imi.i.u-tokyo.ac.jp" % OR IP address like "133.11.216.xxx" export ROS_IP="133.11.216.yyy" % OR run rossetip command to set ROS_IP ``` % *pose_controller.py* in *naoqi_pose* package imports *NaoqiNode* from *naoqi_node.py* in *naoqi_driver_py* package.

% *naoqi_node.py* imports *ALProxy* from *naoqi.py*.

% *naoqi.py* is located under *pynaoqi-python2.7-2.5.5.5-linux64/lib/python2.7/site-packages/*

% NAO_IP is IP address of Pepper. Pepper tells you their address when pushing their belly button.

% Please install `ros-${ROS_DISTRO}-jsk-tools` to use `rossetip` command.

3. Install ROS packages for NAO and Pepper

` mkdir -p catkin_ws/src cd catkin_ws wstool init src wstool merge -t src https://raw.githubusercontent.com/jsk-ros-pkg/jsk_robot/master/jsk_naoqi_robot/pepper.rosinstall wstool update -t src source /opt/ros/${ROS_DISTRO}/setup.bash rosdep install -y -r --from-paths src --ignore-src `

Then, please install Nao/ Pepper mesh files from deb with manual approval of license.

` sudo apt-get install ros-${ROS_DISTRO}-pepper-meshes sudo apt-get install ros-${ROS_DISTRO}-nao-meshes `

If you have ROS >= kinetic, please use [naoqi_dashboard (kochigami-develop)](https://github.com/kochigami/naoqi_dashboard/tree/kochigami-develop). This includes [Important PR](https://github.com/ros-naoqi/naoqi_dashboard/pull/3) for ROS >= kinetic.
```

` cd catkin_ws/src wstool set naoqi_dashboard --git https://github.com/ros-naoqi/naoqi_dashboard wstool update naoqi_dashboard cd naoqi_dashboard git remote add kochigami https://github.com/kochigami/naoqi_dashboard.git git fetch kochigami git checkout -b modify-for-kinetic kochigami/kochigami-develop `

Finally, please compile them.

` cd ../.. # catkin_ws catkin build --continue-on-failure source devel/setup.bash `

% Inside *jsk_robot* package, there are many packages which are not required for *jsk_naoqi_robot*. If we fail to compile them, building process might stop and *jsk_naoqi_robot* packages might not be compiled. We might need to continue compiling (*catkin build –continue-on-failure*) in that case.

4. (optional) For NAO and Pepper developers

Please add following source codes which you need for debugging.

` cd catkin_ws/src wstool set nao_robot --git https://github.com/ros-naoqi/nao_robot (if you use ROS >= melodic) wstool update nao_robot wstool set pepper_robot --git https://github.com/ros-naoqi/pepper_robot wstool set naoqi_driver --git https://github.com/ros-naoqi/naoqi_driver wstool update naoqi_driver wstool set naoqi_bridge --git https://github.com/ros-naoqi/naoqi_bridge wstool set naoqi_bridge_msgs --git https://github.com/ros-naoqi/naoqi_bridge_msgs `

5. (optional) If you want to use [*ros_speech_recognition*](https://github.com/jsk-ros-pkg/jsk_3rdparty/tree/master/ros_speech_recognition) with NAO microphone

Please check the version of ros-$ROS_DISTRO-ros-speech-recognition

` sudo apt update apt search ros-$ROS_DISTRO-ros-speech-recognition `

If version of ros_speech_recognition < 2.1.18

` cd catkin_ws/src mkdir ros_speech_recognition cd ros_speech_recognition git init git config core.sparsecheckout true git remote add origin https://github.com/jsk-ros-pkg/jsk_3rdparty.git echo ros_speech_recognition > .git/info/sparse-checkout git pull origin master cd ../.. catkin build ros_speech_recognition source devel/setup.bash `

If version of ros_speech_recognition >=2.1.18 ` sudo apt-get install ros-$ROS_DISTRO-ros-speech-recognition `

Interface when controlling NAO and Pepper via roseus

Common methods for NAO and Pepper are defined in *naoqieus/naoqi-interface.l*. NAO-specific methods are defined in *naoeus/nao-interface.l*. Pepper-specific methods are defined in *peppereus/pepper-interface.l*. For further details about each method, please refer to [_**naoqieus**_](naoqieus/README.md), [_**naoeus**_](naoeus/README.md), and [_**peppereus**_](peppereus/README.md) respectively. For some methods, they require specific branch (kochigami-develop) because they are not merged into master. If you need this, please change your branch of *naoqi_bridge* and *naoqi_bridge_msgs* as follows:

``` cd catkin_ws/src wstool set naoqi_bridge –git https://github.com/ros-naoqi/naoqi_bridge wstool update naoqi_bridge cd naoqi_bridge git remote add kochigami https://github.com/kochigami/naoqi_bridge.git git fetch kochigami git checkout -b kochigami-develop kochigami/kochigami-develop

cd .. # catkin_ws/src wstool set naoqi_bridge_msgs –git https://github.com/ros-naoqi/naoqi_bridge_msgs wstool update naoqi_bridge_msgs cd naoqi_bridge_msgs git remote add kochigami https://github.com/kochigami/naoqi_bridge_msgs.git git fetch kochigami git checkout -b kochigami-develop kochigami/kochigami-develop ```

In addition, if you have ROS >= kinetic, please fetch the source of *nao_interaction* (master branch) for the time being ([related issue](https://github.com/ros-naoqi/nao_interaction/issues/12)).

` cd .. # catkin_ws/src wstool set nao_interaction --git https://github.com/ros-naoqi/nao_interaction wstool update nao_interaction `
```

Finally, please compile them.

` cd .. # catkin_ws catkin clean catkin build --continue-on-failure source devel/setup.bash `

NAO & Pepper

[_naoqieus_](naoqieus/README.md)

- common interface package for controlling NAO and Pepper via roseus

To connect NAO and Pepper to wifi, please refer to [here](doc/connect_to_wifi.md).

To control multiple robots in one PC, please refer to [here](doc/control_multiple_robots_in_one_pc.md).

To control NAO and Pepper via gazebo simulator and roseus, please refer to [here](doc/simulator.md).

NAO

[_jsk_nao_startup_](jsk_nao_startup/README.md)

- contains ROS launch files for NAO

[_naoeus_](naoeus/README.md)

- package for controlling NAO via roseus

Pepper

[_jsk_pepper_startup_](jsk_pepper_startup/README.md)

- contains ROS launch files for Pepper

[_peppereus_](peppereus/README.md)

- package for controlling Pepper via roseus

[_jsk_201504_miraikan_](jsk_201504_miraikan/README.md)

- demo package which Pepper introduces themselves