How To Install ROS Melodic, on Raspberry Pi 4 (Raspbian Buster)

The process of building an autonomous robot starts before the first sensor or actuator mounted on the chassis. The build process requires hardware and software setup, which is the topic of this article.

Different projects might have different requirements. For this reason, I used to work with the most powerful version of Pi: the Raspberry Pi 4 Model B with 4GB of RAM. This version is capable of running ROS, algorithms to detect objects, and up deep learning algorithms.

Install ROS Melodic

ROS (Robot Operating System) is a framework that is running on top of Raspbian. The framework includes a set of libraries and tools that we will use to build robots.

ROS is an all-in-one solution if you want to build different robots. Once knowing how to write nodes, services, actions, and use the software packages, you can make different kinds of robots, including robot arms, drones, or autonomous mobile robots.

ROS provides a wide range of packages for almost any robotic application. Mapping, navigation and localization, are just a few examples of useful packages to build robots.

Do you want to use Python instead of C++ to control your robot? You are allowed to write your Python nodes and make it communicate with C++ nodes.

Do you want to build your robot in a virtual world? You need a simulation tool. ROS has excellent tools for simulation and visualization, such as Rviz and Gazebo.

You can run multiple independent robots and make them communicate with each other. The list with the benefits of using ROS is much longer and covers different areas in robot development.

There is currently no official ROS Melodic installation guide for the Raspberry Pi 4 and Raspbian Buster operating system. Hence we will need to build it from the source.

We are very close to beginning to run the commands to install ROS. Before running the first command, I will do a short introduction to the ROS distribution. For this tutorial and my next projects, I choose to work with ROS Melodic distribution. This distribution was released in 2018 and will have support until 2023.

Let’s begin to install ROS Melodic:
Step 1: Setting up the repositories:

```
$sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

```
$sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Step 2: Install build tools (compiler, CMake, etc.)

- *rosdep* is a command-line tool for installing system dependencies.
- *rosinstall_generator* generates rosinstall files containing information about repositories with ROS packages/stacks.
- *wstool* is a tool for maintaining a workspace of projects from multiple version-control systems.

```
$sudo apt-get update
$sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-
essential cmake
```

Step 3: Initializing *rosdep*
rosdep has rules for all the dependencies, and if something is not working, it might be worth reporting it as an issue.

```
$sudo rosdep init
$rosdep update
```

Step 4: Create a catkin workspace to build the core packages.

```
$mkdir ~/ros_catkin_ws
$cd ~/ros_catkin_ws
```

Step 5: Fetch the core packages
The minimal Desktop install might be the preferred choice for Raspberry Pi since the board probably will be running on top of the robot. This installation doesn't include 2D/3D simulators, navigation, and 2D/3D perception.

```
$rosinstall_generator desktop --rosdistro melodic --deps --wet-only --tar > melodic-desktop-wet.rosinstall
$wstool init -j4 src melodic-desktop-wet.rosinstall
```

Step 6: Install the compatible version of Assimp (Open Asset Import Library) to fix collada_urdf dependency problem and OGRE for rviz.

```
$cd ~
$mkdir -p ~/ros_catkin_ws/external_src
$cd ~/ros_catkin_ws/external_src
$wget https://sourceforge.net/projects/assimp/files/assimp-3.1/assimp-
3.1.1_no_test_models.zip/download -O assimp-3.1.1_no_test_models.zip
$unzip assimp-3.1.1_no_test_models.zip
$cd assimp-3.1.1/
$cmake .
$make
$sudo make install
$sudo apt-get install libogre-1.9-dev
```

Step 7: Resolving Dependencies
We must make sure that all the required dependencies are in place:

```
$cd ~/ros_catkin_ws/
$rosdep install --from-paths src --ignore-src --rosdistro melodic -y
```

Step 8: Build and Source the Installation
In the steps above, we complete downloading the packages and resolving the dependencies. At this step, we are ready to build the catkin packages.
We use the *catkin_make_isolated* command because there are both catkin and plain *cmake* packages in the base install when developing on your catkin only workspaces you may choose to use *catkin/commands/catkin_make* which only works with catkin packages.

```
$sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/melodic -j2

$echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

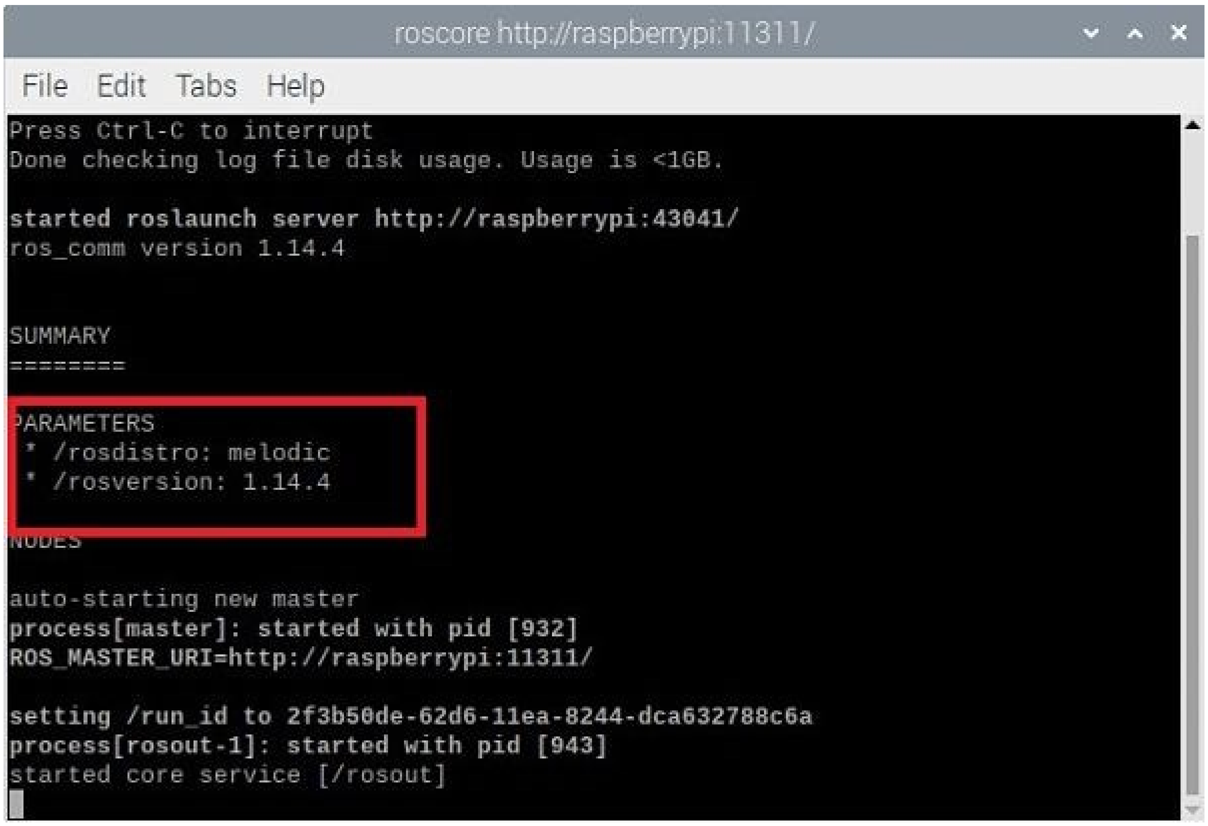
At this point, we finish the ROS installation. The last thing is to restart the Raspberry Pi

```
$sudo reboot
```

After the Pi is up and running, we do a check and start a roscore session:

```
$roscore
```

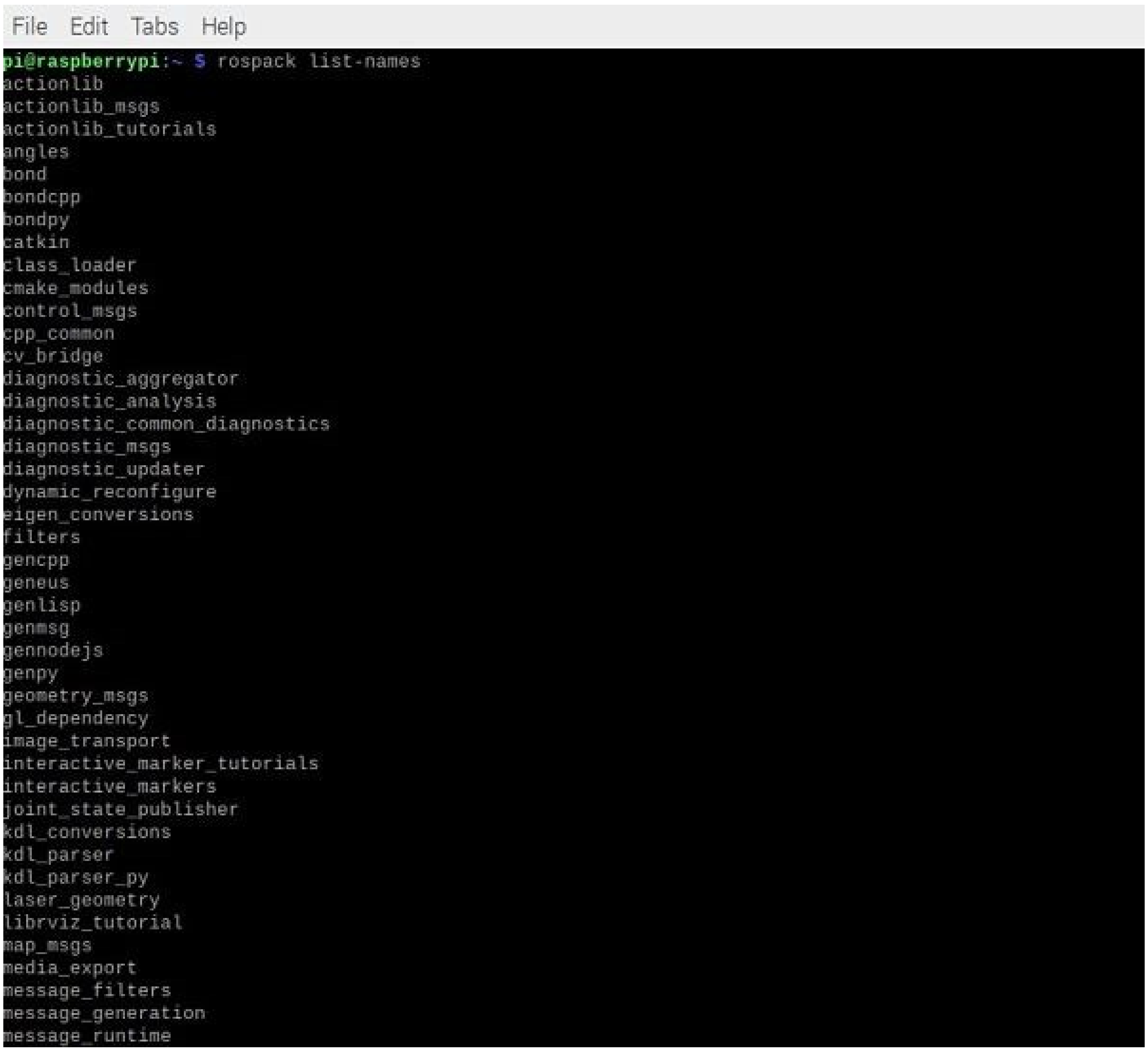
You should see a roscore session like in the bellow image:



```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:43041/
ros_comm version 1.14.4


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.4

NODES

auto-starting new master
process[master]: started with pid [932]
ROS_MASTER_URI=http://raspberrypi:11311/

setting /run_id to 2f3b50de-62d6-11ea-8244-dca632788c6a
process[rosout-1]: started with pid [943]
started core service [/rosout]
```

roscore session

Check the list of ROS packages installed:



```
pi@raspberrypi:~ $ rospack list-names
actionlib
actionlib_msgs
actionlib_tutorials
angles
bond
bondcpp
bondpy
catkin
class_loader
cmake_modules
control_msgs
cpp_common
cv_bridge
diagnostic_aggregator
diagnostic_analysis
diagnostic_common_diagnostics
diagnostic_msgs
diagnostic_updater
dynamic_reconfigure
eigen_conversions
filters
gencpp
geneus
genlisp
genmsg
gennodejs
genpy
geometry_msgs
gl_dependency
image_transport
interactive_marker_tutorials
interactive_markers
joint_state_publisher
kdl_conversions
kdl_parser
kdl_parser_py
laser_geometry
librviz_tutorial
map_msgs
media_export
message_filters
message_generation
message_runtime
```

rospack list-names

```
$rospack list-names
```

To end the *roscore* session run the command *Ctrl+c*.

At this point, we are ready to start the work. The first thing is to create workspace to build your packages to run on the robot:

```
$mkdir -p ~/catkin_ws/src
$cd ~/catkin_ws/
$catkin_make
$echo "source $HOME/catkin_ws/devel/setup.bash" >> ~/.bashrc
```