HOW to design ROS-powered robots

Luca Marchionni, CTO at PAL Robotics

September 22, Vancouver

ROSCon 2017
Outline

- PAL Robotics
- Software overview and deployment
- Continuous integration
- Software release system
- Control architecture
- Whole Body Control
- Space Robotics Challenge
Why do we use ROS?

- Shorten time to market
- Selling argument
- Leverage powerful development tools
- If you have the code, you can improve it
- Share the effort with community
- Contribute to create “standards”
- Build value, focus on core IP
Public repositories

wiki.ros.org/Robots/REEM
wiki.ros.org/Robots/REEM/Tutorials

wiki.ros.org/Robots/TIAGo
wiki.ros.org/Robots/TIAGo/Tutorials

wiki.ros.org/Robots/REEM-C
wiki.ros.org/Robots/REEM-C/Tutorials

https://github.com/pal-robotics
72 repositories

https://github.com/pal-robotics-forks
60 repositories
Public repositories (coming soon)

- New generation of biped robot
- Advanced mobility and manipulation skills for industrial tasks
- Full torque controllable robot
- EtherCAT communication bus
- High power/speed actuators
- 6 Kg arm payload
# Software overview

<table>
<thead>
<tr>
<th></th>
<th>Stable</th>
<th>Work in progress</th>
<th>Future?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating System</strong></td>
<td>● Ubuntu 14.04 LTS 64-bit</td>
<td>● Ubuntu 16.04 LTS 64-bit</td>
<td>● Linux Real Time</td>
</tr>
<tr>
<td></td>
<td>● Xenomai real-time</td>
<td>● Linux Preemp-rt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Linux Preemp-rt</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Robotics middleware</strong></td>
<td>● Orocos 2.8</td>
<td>● Orocos 2.8</td>
<td>● ROS 2.0</td>
</tr>
<tr>
<td></td>
<td>● ROS Indigo</td>
<td>● ROS Kinetic</td>
<td>● PAL Fermium</td>
</tr>
<tr>
<td></td>
<td>● PAL Dubnium</td>
<td>● PAL Erbium</td>
<td></td>
</tr>
</tbody>
</table>

- **Ubuntu 14.04 LTS 64-bit**
- **Ubuntu 16.04 LTS 64-bit**
- **Linux Preemp-rt**
- **Linux Real Time**
- **ROS Indigo**
- **ROS Kinetic**
- **PAL Dubnium**
- **PAL Erbium**
- **ROS 2.0**
- **PAL Fermium**
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version

`pal_deploy`

```
/opt/ros/indigo
```
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version

```bash
$ pal_deploy
```

```
/opt/pal/dubnium
```

```
/opt/ros/indigo
```
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version

```
/home/pal/deployed_ws
/opt/pal/dubnium
/opt/ros/indigo
```

pal_deploy
Software deployment

- Add/Overlay packages to a robot
- Validate package installation rules
- Discourage file editing on robot
- Multiple workspaces
- Test before release
- Restore original package version

```
/api/abi compatibility!!!
/home/pal/deployed_ws
/opt/pal/dubnium
/opt/ros/indigo
```

pal_deploy
Continuous integration

- Builds in clean environment
- Unit testing
- Coverage
- API/ABI check
- Functional testing
Continuous integration

Notify developers and maintainers if:

- Compilation error
- Test failure
- Coverage below 70%
- API/ABI breaking

Run it on master/develop branch
(GitFlow branching model)
Continuous integration

- Jeremie Deray @jeremiederay added 1 commit 2 weeks ago
  - fe56d1f3 - moar tests
  
  Compare with previous version

PAL Bot @anonymous commented 2 weeks ago

| extend-tests | build: stable | tests: 34/34 | C++ coverage: 99% | Py coverage: -- | API: compatible | ABI: compatible |
| master       | build: success| tests: 22/22 | C++ coverage: 88% | Py coverage: -- | API: No Info   | ABI: No Info   |


Edited less than a minute ago by Victor Lopez

Jeremie Deray @jeremiederay commented 2 weeks ago

As pointed by @victor giving the PropertyBag a name may not be of any use... @victor @hilariotome let me know your opinion before I remove it from this PR.
Software release system
Software release system
Software release system

{dubnium-devel} version depends on:

- depthimage_to_laserscan
- pal_local_planner
- pal_pcl
- sick_tin
- pal_navigation_sm
- pal_laser_filters
- pal_vo_server
- slam_gmapping
- rviz_plugin_covariance
- pal_karto
- navigation

Released at version: 0.10.0-staging.0

Released against:

- depthimage_to_laserscan: 1.0.8-staging.0
- pal_local_planner: 2.0.1-staging.0
- pal_pcl: 0.1.12-staging.0
- sick_tin: 0.0.9-staging.0
- pal_navigation_sm: 0.1.8-staging.0
- pal_laser_filters: 0.0.2-staging.0
- pal_vo_server: 0.0.16-staging.0
- slam_gmapping: 1.3.6-staging.0
- rviz_plugin_covariance: 0.0.5-staging.0
- pal_karto: 0.7.1-staging.0
- navigation: 1.11.28-staging.0

Depended on by:

- pal_menapkg_development_pmb2
- pmb2_simulation
- pal_menapkg_pmb2
- pmb2_navigation_specifics
Software release system
Control architecture
Control architecture

Hardware

Real Time

ROS Control

Non Real Time

ROS

Actuators Manager

Trajectory Controller

Walking Controller

Whole body Controller

Navigation

Perception

Interaction

Manipulation
Control architecture

**Why?**
- Hardware needs real time communication
- Controllers need determinism regardless of system load

**How?**
- Xenomai co-kernel or Preempt-RT patch
- Never block the real time thread!
Control architecture

Hardware
- URDF
- .yaml
- .launch

Real Time
- Actuators Manager
- ROS Control
  - Trajectory Controller
  - Walking Controller
  - Whole body Controller

Non Real Time
- Navigation
- Perception
- Interaction
- Manipulation

ROS

ROSCon 2017
Whole body control

1) set of simple, low-dimensional rules

2) the rules are sufficient to guarantee the correct execution of any single task or of simultaneous multiple tasks

3) exploiting the full capabilities of the entire body of redundant, floating-based robots in compliant multi-contact interaction with the environment

source: http://www.ieee-ras.org/whole-body-control
Whole body control
Whole body control

Hierarchical Quadratic Program solver

Lexicographic minimization

\[ f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x) \]

subject to

\[ f_1(x) < f_2(x) < f_3(x) < f_4(x) \]
Whole body control

<table>
<thead>
<tr>
<th>Stack of Tasks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>High priority</td>
<td>Joint limits</td>
</tr>
<tr>
<td></td>
<td>Self collision avoidance</td>
</tr>
<tr>
<td></td>
<td>Fixed feet + CoM centered</td>
</tr>
<tr>
<td></td>
<td>Gaze</td>
</tr>
<tr>
<td></td>
<td>Hands position</td>
</tr>
<tr>
<td></td>
<td>Torso orientation upright</td>
</tr>
<tr>
<td>Low priority</td>
<td>Joint reference posture</td>
</tr>
</tbody>
</table>
Whole body control
Whole body control
Space Robotics Challenge

- 440 inscribed teams, 20 finalists
- Control Valkyrie robot in simulated Mars mission
- Team composed by current and former PAL employees
- Ineligible for prize, motivated by passion for robotics
Space Robotics Challenge
TALOS Robotics Challenge (work in progress)
Thank you ROS community!!!

www.pal-robotics.com