

ROSViTA

robot programming software

Fast and intuitive robot programming based on ROS!

Dr. Andre Lemme
Xamla Robotics Team



ROS



xiimea



IDS:



YASKAWA



UR



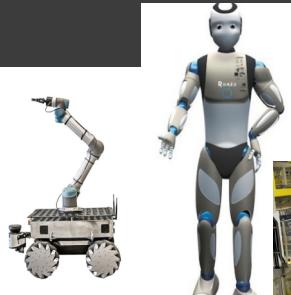
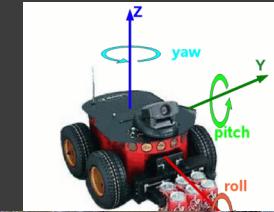
ROBOTIQ



Table of contents

- 1 Overview robotics field**
- 2 Introduction of ROS**
- 3 ROS vs. ROSVITA concepts**
- 4 Getting started with ROSVITA**
- 5 Docker as deployment tool**
- 6 Robot configuration**
- 7 World view**
- 8 Status monitoring**
- 9 Costum robot part**
- 10 Concluding**





1.1 Overview of Robotics



Mobile Robots



Humanoid Robots



Service Robots

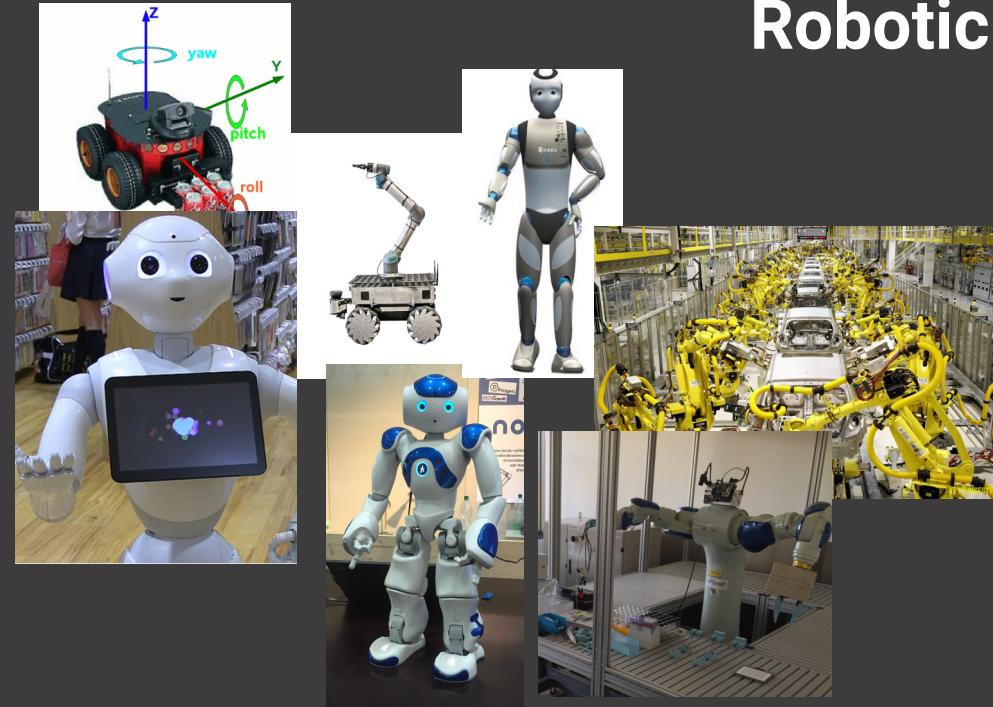


Industrial Robots

Robotics is a very diverse field where restrictions and requirements come in different combinations!

How can we handle this? → Robotic-Framework

1.2 Requirements and Tasks of a Robotic-Framework



- ✓ Integration of sensors and actuators
- ✓ Software interfaces

High-Level functionality can be used on different hardware platforms!
ROS represents such a Robotic-Framework.

2.1 Introduction of ROS

- Robot Operating System (ROS)**
- Development started in context of the Standford-AI-Robot-Projects in 2007**
- Since 2009, the research institute Willow Garage took over**
- Since 2013, the Open Source Robotics Foundation (OSRF) is in charge**
- Mobile Robots**
- Humanoid Robots**
- Service Robots**
- Acknowledged standard in research and development**

2.1 Introduction of ROS

- ✓ **Robot Operating System (ROS)**
- ✓ **Linux, MacOS (experimental)**
- ✓ **Open-Source Robotics-Framework**
 - ✓ Core parts of ROS → BSD License
 - ✓ Community packages → Apache 2.0, GPL, MIT License
 - ✓ e.g. MoveIt: path-planning with collision avoidance
- ✓ **Mobile Robots**
- ✓ **Humanoid Robots**
- ✓ **Service Robots**
- ✓ **Industrial Robots**

What about industrial applications?

2.2 ROS-Industrial

① started with the collaboration of

- Yaskawa Motoman Robotics,
- Southwest Research Institute
- Willow Garage

② founded by Shaun Edwards (SwRI)
in January 2012



Industrial Robots



Shows promising applications

① Welding

② Blending process

③ Pick and Place

④ Etc...

Community size?

2.3 ROS-Industrial

Consortium Americas
(led by SwRI and located in San Antonio)

Consortium Europe
(led by Fraunhofer IPA and located in Stuttgart, Germany)

Consortium Asia Pacific
(led by Advanced Remanufacturing and Technology Center (ARTC) and Nanyang Technological University (NTU) located in Singapore)

Release cycle of ROS?

2.4 ROS Distribution



ROS
Box Turtle

March 2, 2010



Box Turtle



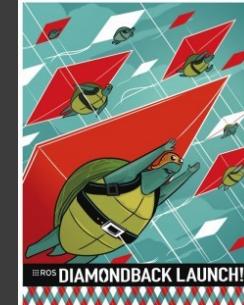
ROS
C Turtle

August 2, 2010



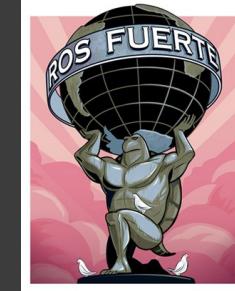
ROS
Diamondback

March 2, 2011



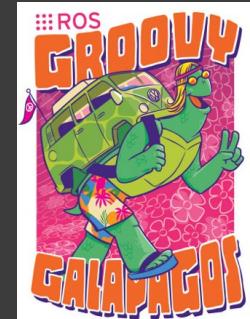
ROS
Fuerte Turtle

April 23, 2012



ROS
Groovy Galapagos

December 31, 2012



2.4 ROS Distribution

ROS Hydro Medusa	ROS Indigo Igloo	ROS Jade Turtle	ROS Kinetic Kame	ROS Lunar Loggerhead
September 4, 2013	July 22, 2014	May 23, 2015	May 23, 2016	May 23, 2017
				

Synced with ubuntu releases
Every two years new LTS release

2.5 ROS Concepts

1 Modularity

- ✓ Packages for organization
- ✓ Basic build and release item (catkin workspace)

2 Robot programs

- ✓ Nodes and launch-files

3 Communication

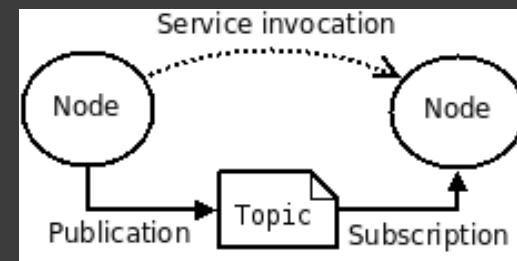
- ✓ Topics
- ✓ Services
- ✓ Actions

4 Tools

- ✓ `rostopic`, `rosservice`, `rosparam`, `roscd`, `rviz`

5 Processes

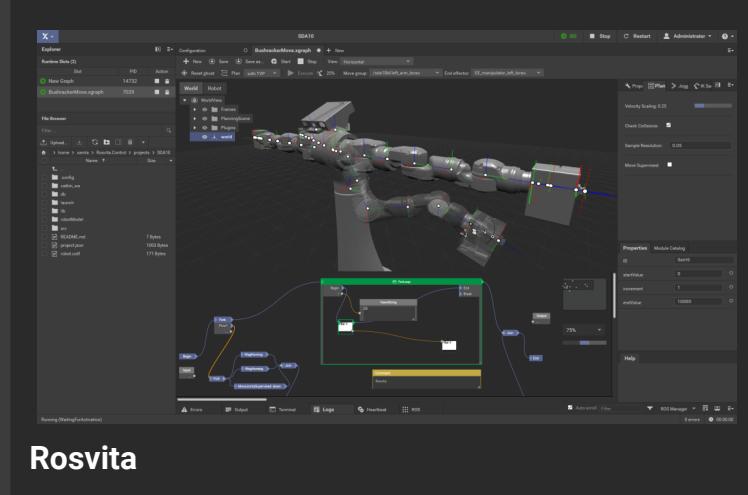
- ✓ Represented in graph architecture



Goals:
Scale-ability
Flexibility
?Robustness?

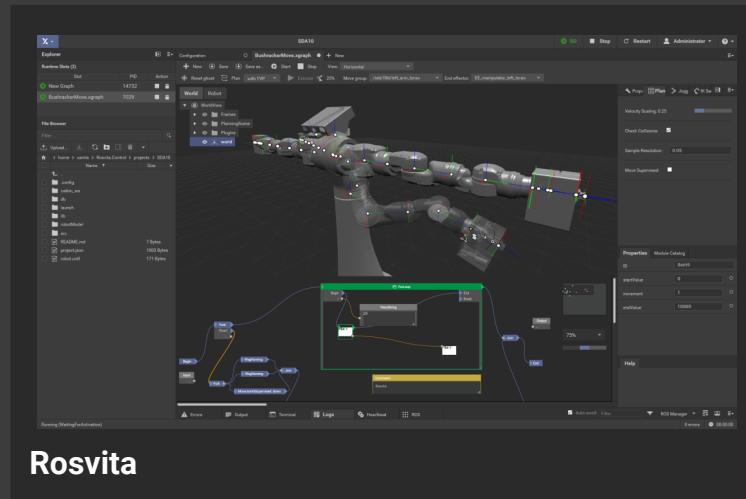
From ROS to ROSVITA

3.1 ROS vs. ROSVITA concepts



- ✓ Rosvita is a browser based programming IDE on top of ROS and therefore also using the ROS concepts!
- ✓ Web-Server
- ✓ RosGardener
- ✓ RosServices
- ✓ Motion planning with xamlamoveit

3.2 ROSVITA



Rosvita

- ✓ Browser based IDE on top of ROS and therefore also using the ROS concepts!
- ✓ High quality 2D and 3D visualization
- ✓ Ready-for-Industry robot drivers and other reliable ROS components
- ✓ Visual workflow editor with an expandable module system
- ✓ All features bundled in docker container for easy deployment

4.1 Getting started with ROSVITA

Getting Started

To use ROSVITA, no complex installation process is necessary, because the ROSVITA Docker image already contains all the necessary packages and dependencies. All you need is an up-to-date Ubuntu operating system (preferentially **Ubuntu 16.04 or higher**) and an up-to-date internet browser (preferentially Google Chrome). Moreover, please read the ROSVITA license agreement.

To get ROSVITA, simply download the shell script `get_rosvita`, then change permissions to be able to execute the script and finally run the script, i.e. in a terminal type:

```
 wget -q https://raw.githubusercontent.com/Xamla/docs.xamla.com/gh-pages/rosvita/downloads/get_rosvita
 chmod u+x get_rosvita
 ./get_rosvita
```

Here, we use `wget` to download the script. If not already installed, simply install `wget` via
`sudo apt install wget`.

The script will:

1. Install Docker if not already installed, and add user to group docker.
2. Download the **ROSVITA Docker image** from Docker Hub.
3. Download the **ROSVITA start script**: `rosvita_start`.
4. Download the **ROSVITA stop script**: `rosvita_stop`.
5. Create a **desktop icon** for starting ROSVITA.
6. If not already in group docker, **restart the computer** to activate group changes.

Now you can start ROSVITA by double clicking the desktop icon, or by opening a terminal and running the start script by simply typing:

```
rosvita_start
```

(To run ROSVITA in **debug mode** type `rosvita_start --debug`. Afterwards enter `rosvita` into the terminal, open a web browser and enter `localhost:5000` into the address bar.)

VIDEO



1. Install Docker if not already installed, and add user to group docker.
2. Download the **ROSVITA Docker image** from Docker Hub.
3. Download the **ROSVITA start script**: `rosvita_start`.
4. Download the **ROSVITA stop script**: `rosvita_stop`.
5. Create a **desktop icon** for starting ROSVITA.
6. If not already in group docker, **restart the computer** to activate group changes.

Now you can start ROSVITA by double clicking the desktop icon, or by opening a terminal and running the start script by simply typing:

```
rosvita_start
```

(To run ROSVITA in **debug mode** type `rosvita_start --debug`. Afterwards enter `rosvita` into the terminal, open a web browser and enter `localhost:5000` into the address bar.)

As a result, the ROSVITA login screen appears in your default web browser. After successful login with username and password (default: `admin` and `r2d2c3po`), the ROSVITA main development environment opens (see next chapter).

To **stop** ROSVITA simply run the ROSVITA stop script:

```
rosvita_stop
```

To **update** ROSVITA simply run the `get_rosvita` script again, or use the following command:

```
docker pull xamla/rosvita:latest
```

With the terminal command `docker images` you will see all downloaded ROSVITA versions with corresponding image ids. Use `docker rmi <image-id>` to remove old versions and save disk space.

If you have any trouble or would like to give us feedback, feel free to visit our ROSVITA forum. Moreover, check out our **ROSVITA tutorial videos**, especially the ROSVITA quick start video.

Note

We strongly recommend to run ROSVITA in **Google Chrome**, because in other internet browsers like Firefox the user experience will be limited due to the significantly slower browser performance.

[Previous](#)

[Next](#)

4.2 Docker as deployment tool



“Docker – Build, Ship, and Run Any App, Anywhere.”

- ✓ Docker image
- ✓ Persistent data located in host system
 - ✓ ~/Rosvita/data
 - ✓ ~/Rosvita/projects
 - ✓ ~/Rosvita/robot_parts
- ✓ Advantages
 - ✓ Available on Linux/MacOs/Windows
 - ✓ Robust system → image restart
 - ✓ Dependencies bundled

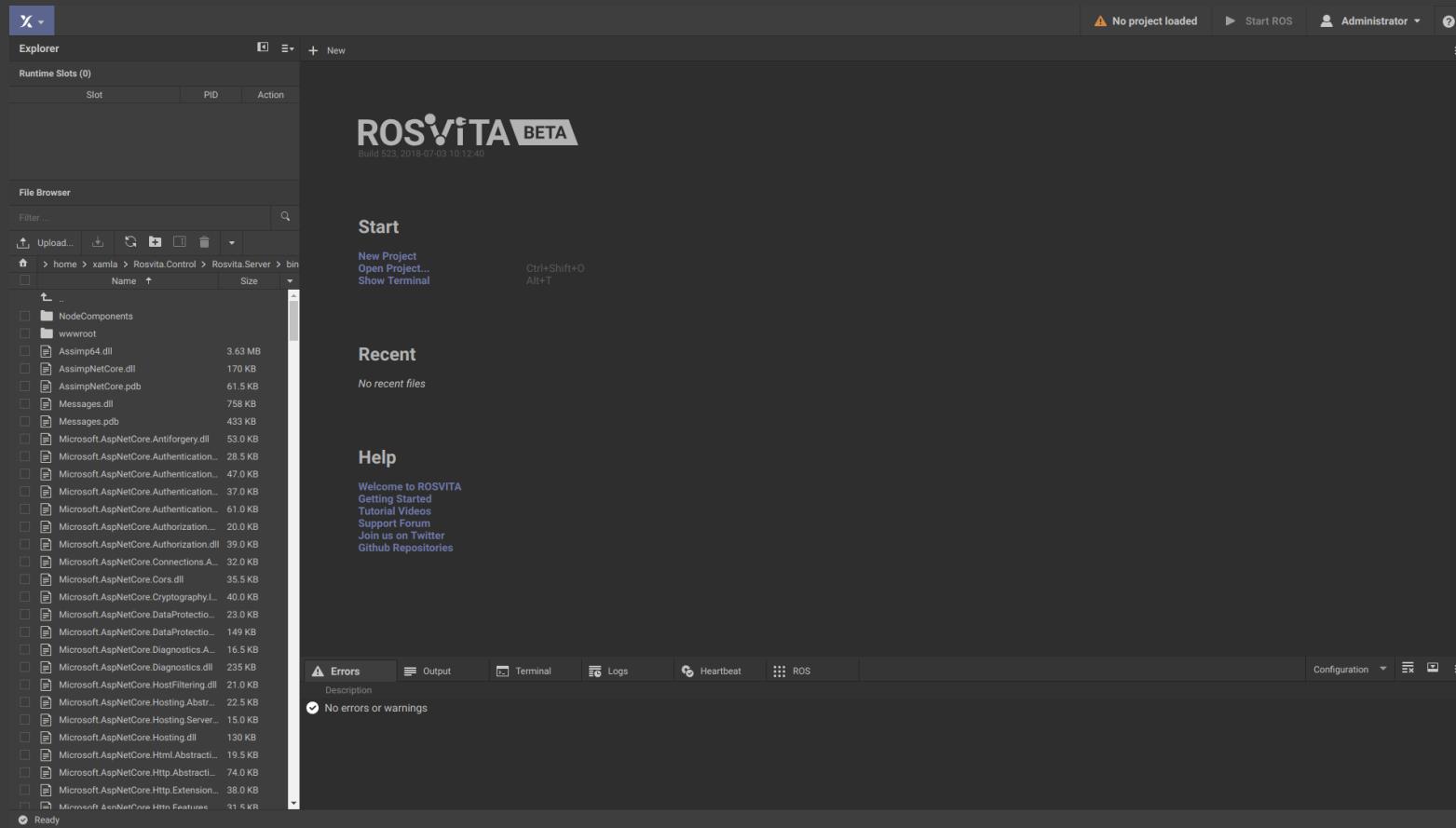
4.3 What is docker and why is it useful?



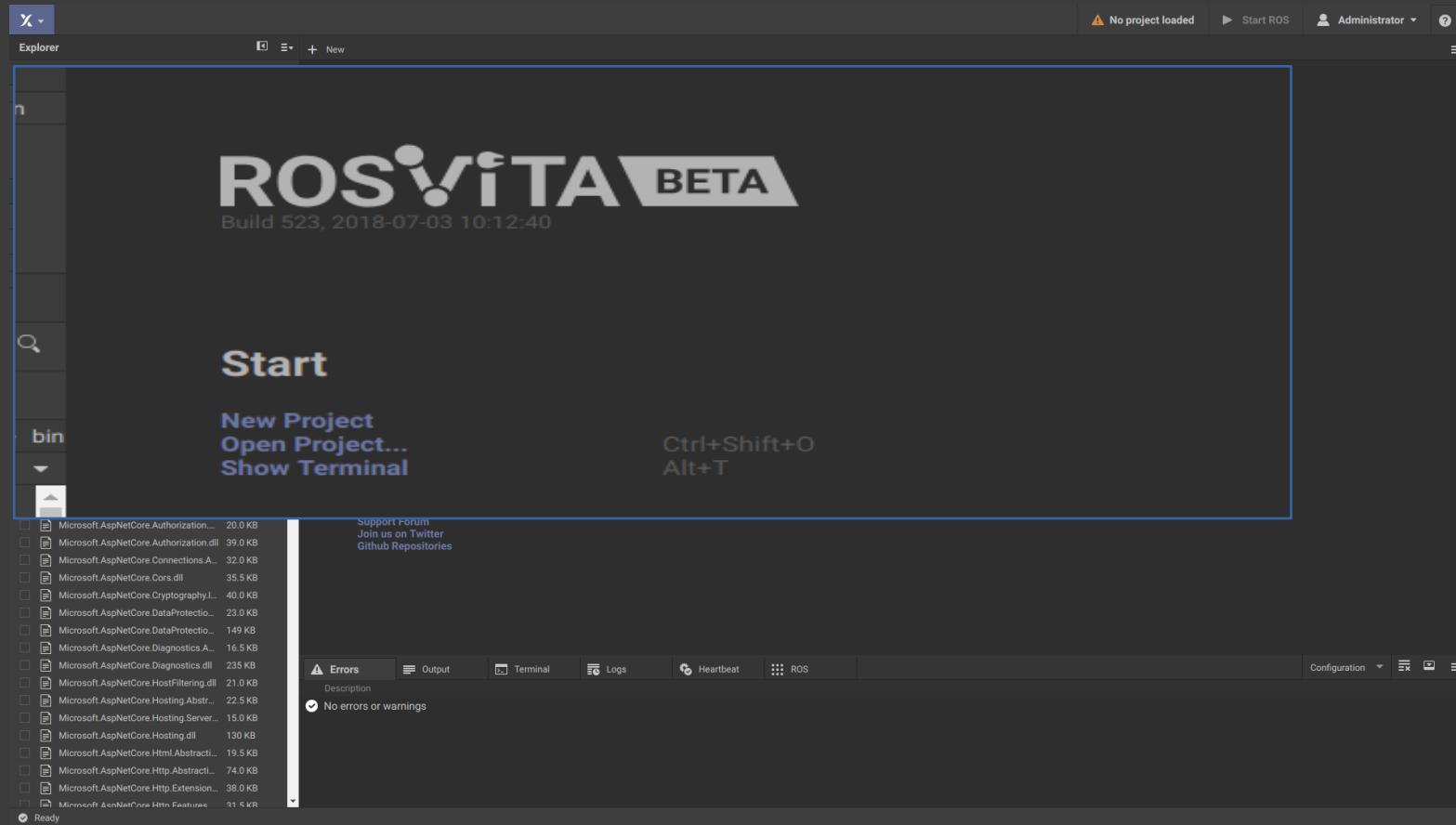
“Docker – Build, Ship, and Run Any App, Anywhere.”

- ✓ Docker image
- ✓ Persistent data located in host system
- ✓ Advantages
 - ✓ Available on Linux/MacOs/Windows
 - ✓ Robust system → image restart
 - ✓ Dependencies bundled
- ✓ Disadvantages
 - ✓ Difficult to connect to hardware

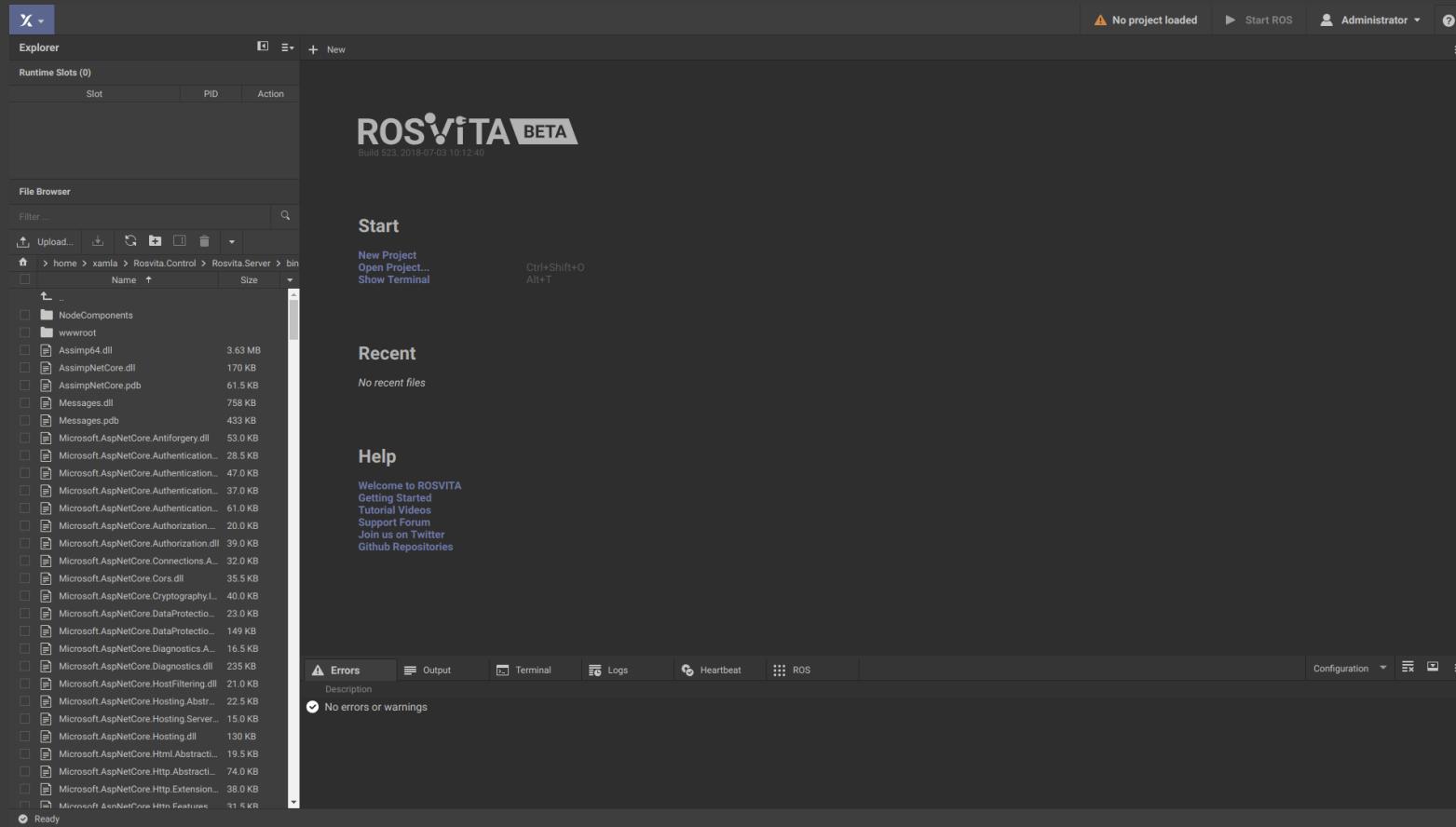
5.1 Welcome to ROSVITA



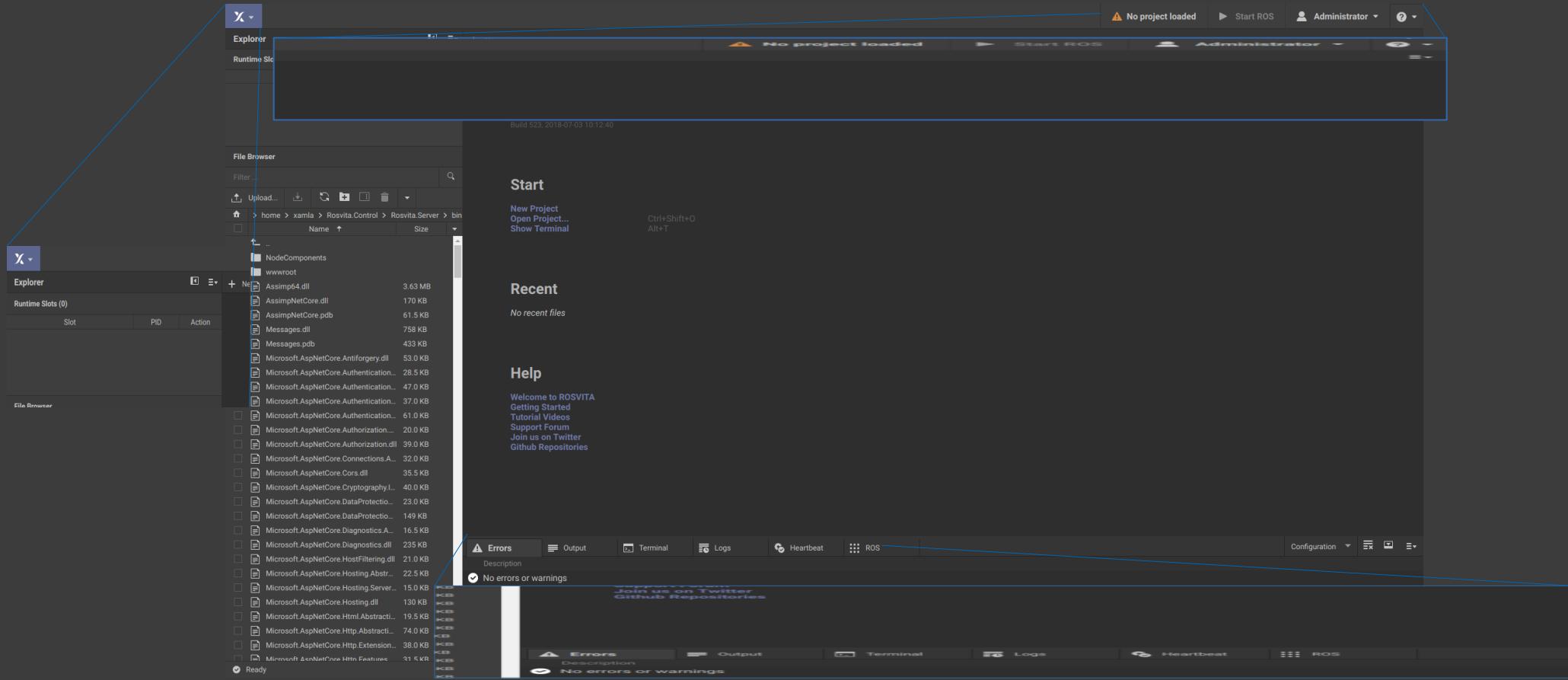
5.2 Welcome to ROSVITA



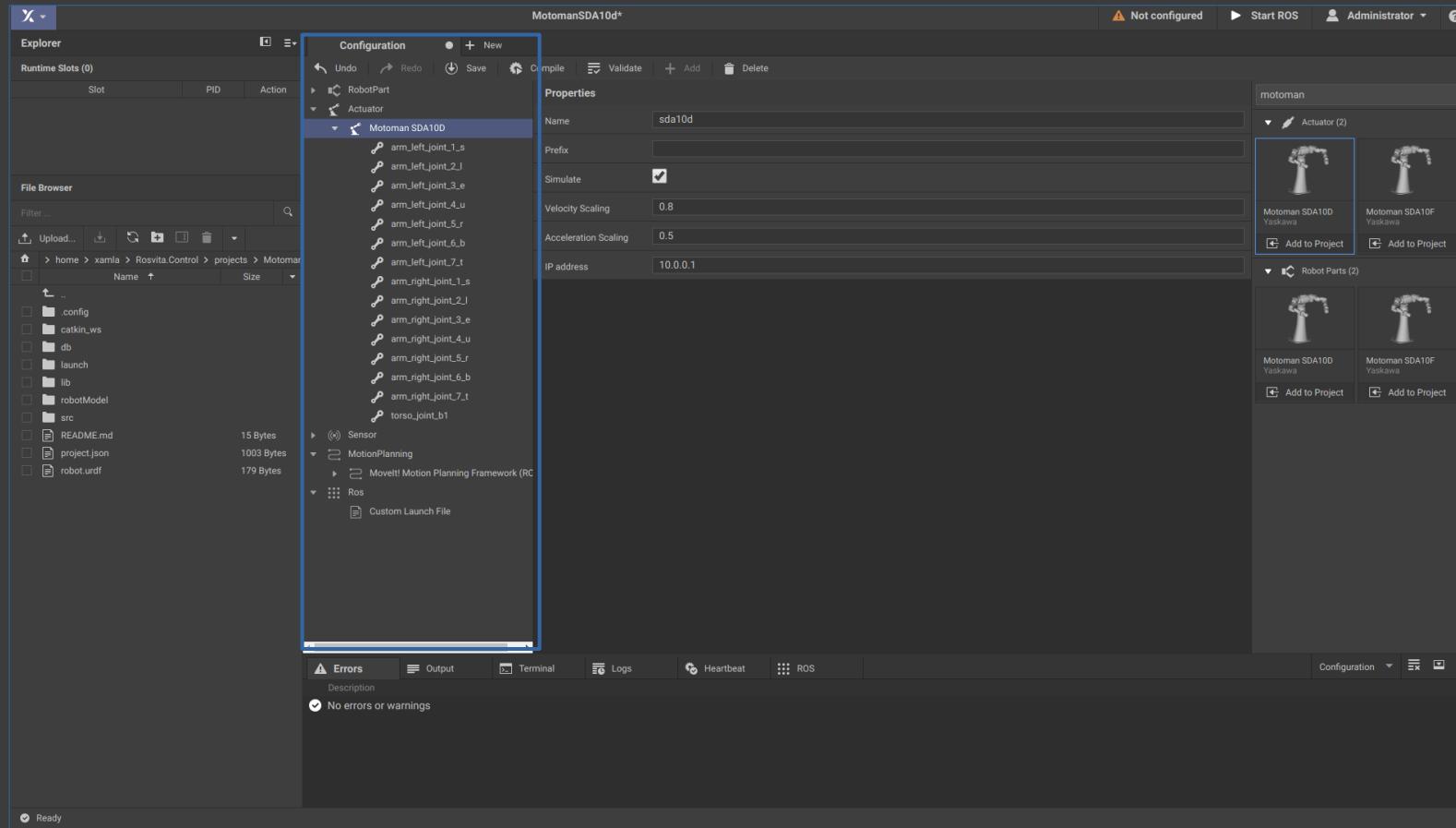
5.2 Welcome to ROSVITA



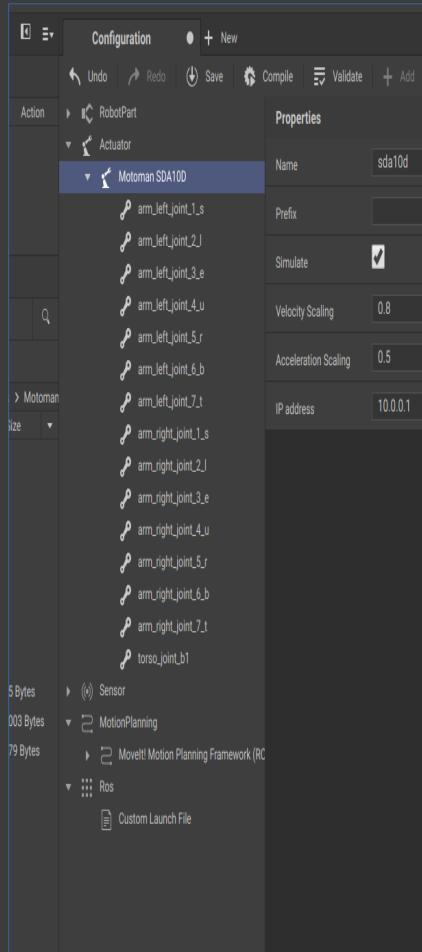
5.2 Welcome to ROSVITA



6.1 Robot configuration



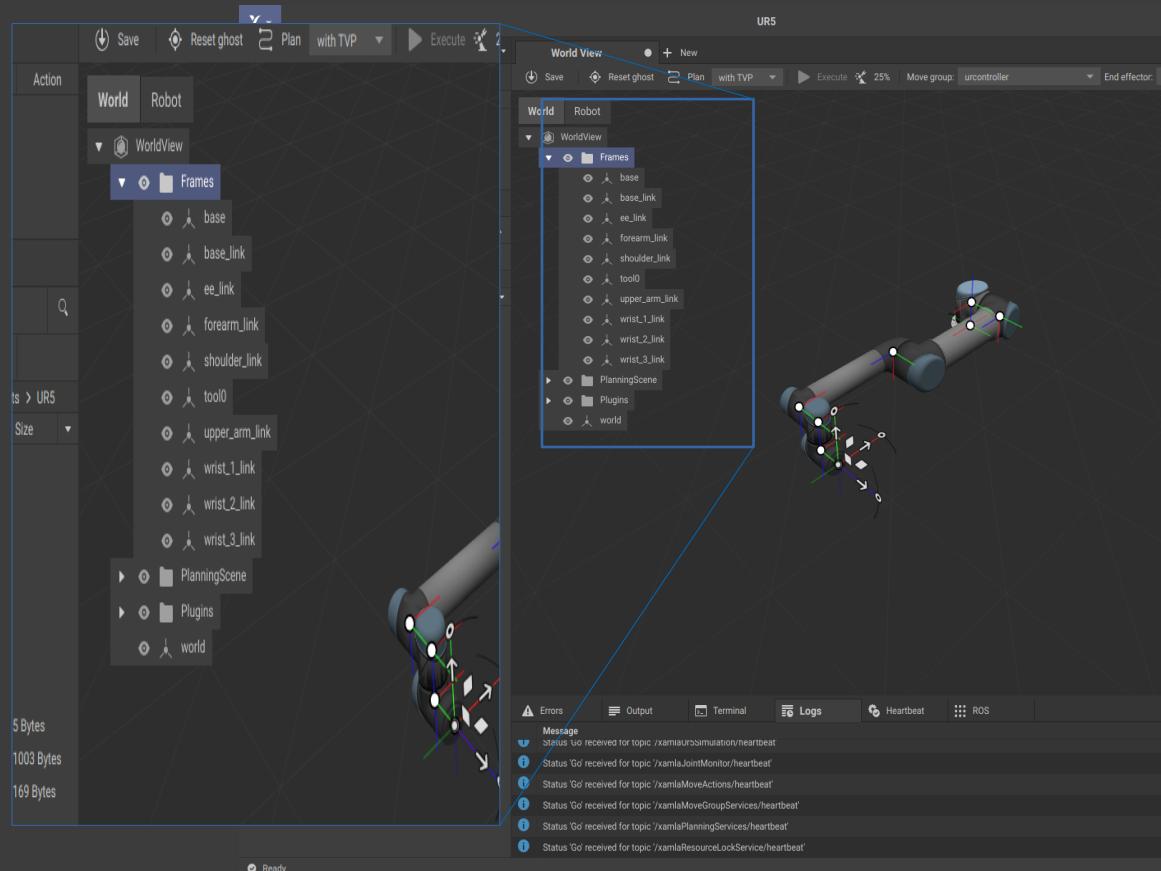
6.2 Robot configuration



- ✓ **RobotPart**
 - ✓ URDF, Xacro, standard interface
- ✓ **Actuator**
 - ✓ Driver, corresponding to urdf
- ✓ **Sensor**
 - ✓ Cameras etc...
- ✓ **MotionPlanning**
 - ✓ MoveIt configuration and extras
- ✓ **ROS Launch File**

7.1 World view

7.1 World view



✓ **Frames**

✓ **CollisionObjects**

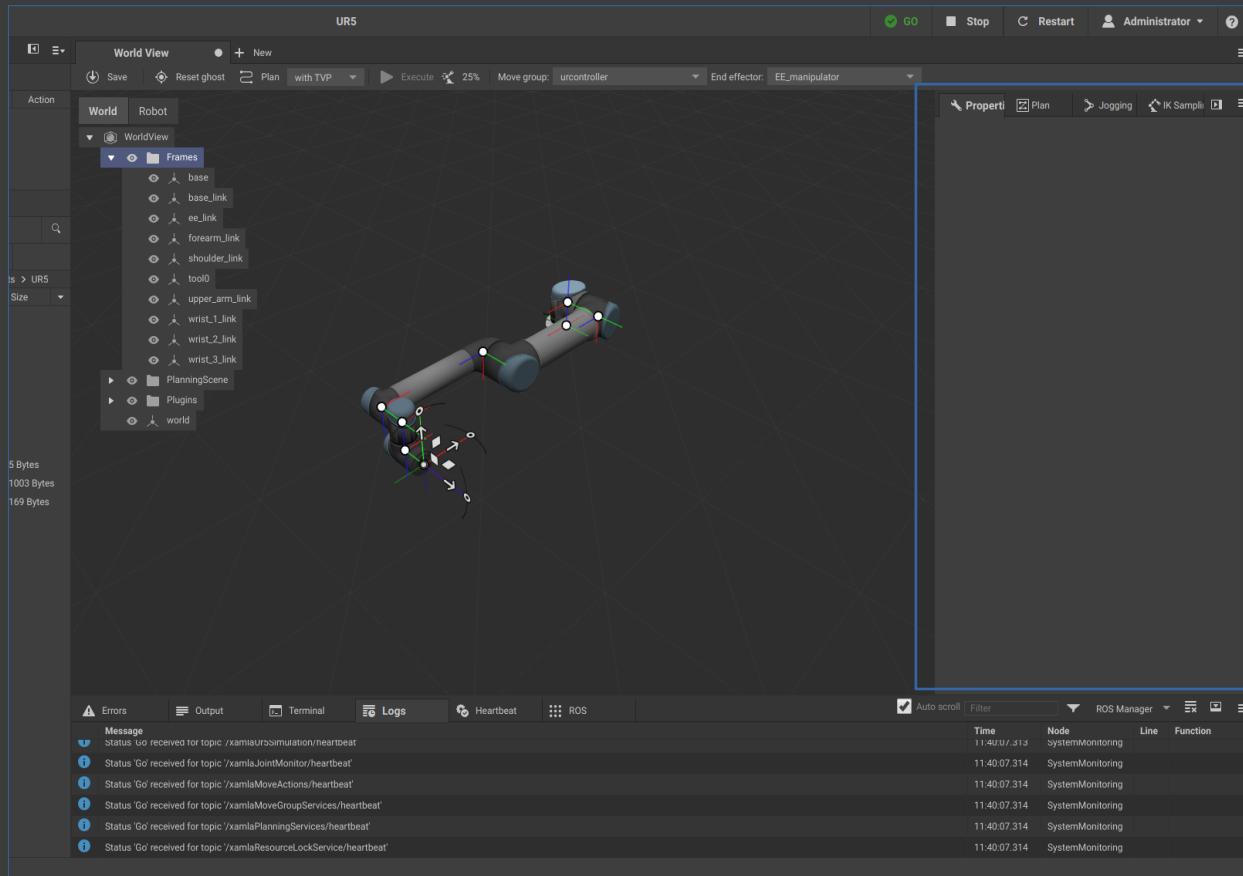
✓ **Plugins**

✓ **Poses**

✓ **JointValues**

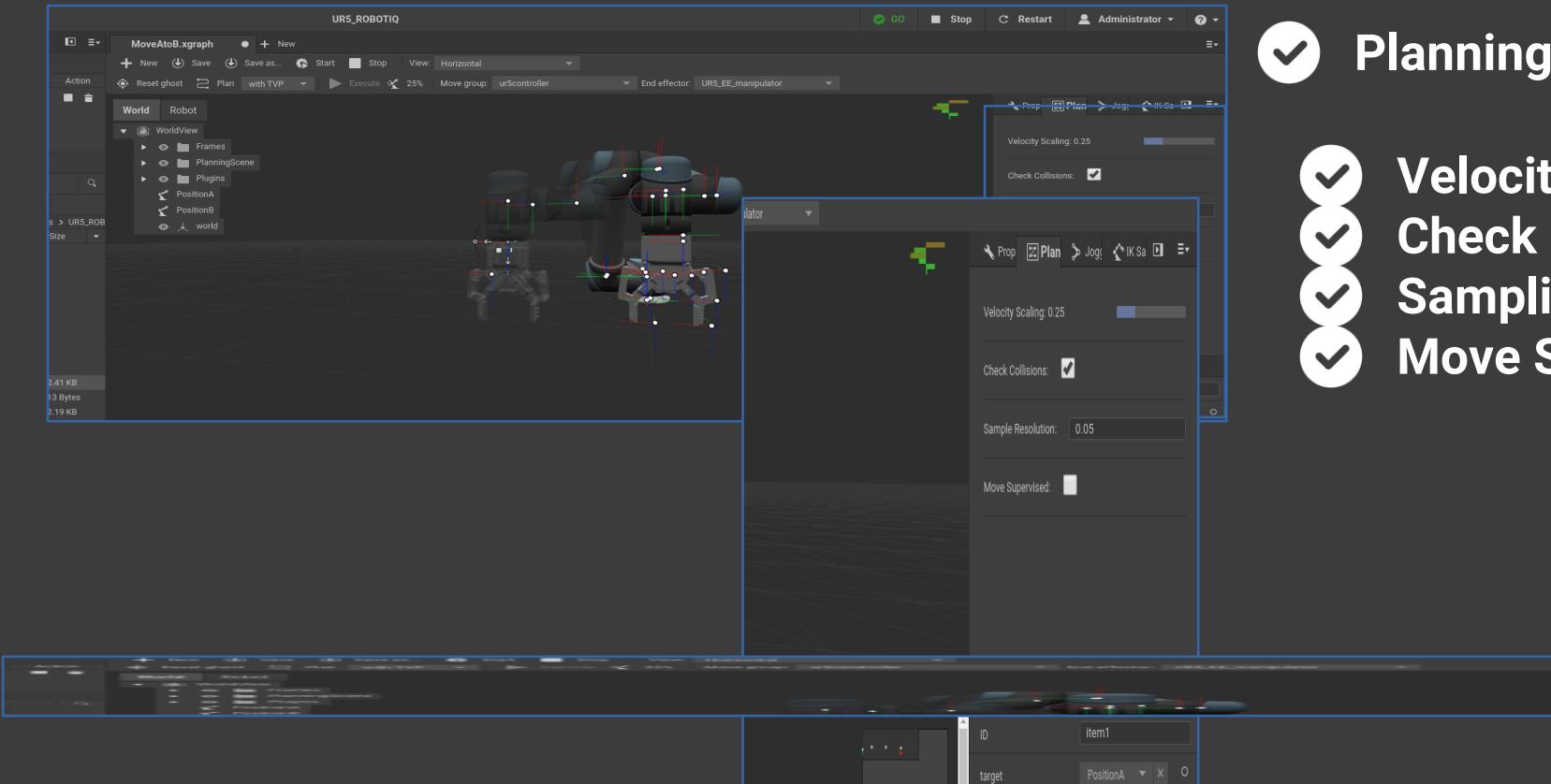
7.1 World view

7.2 World view



✓ Property view
✓ Detailed information of all datatypes in the tree view.

7.3 World view



Planning view



Velocity scaling



Check collisions

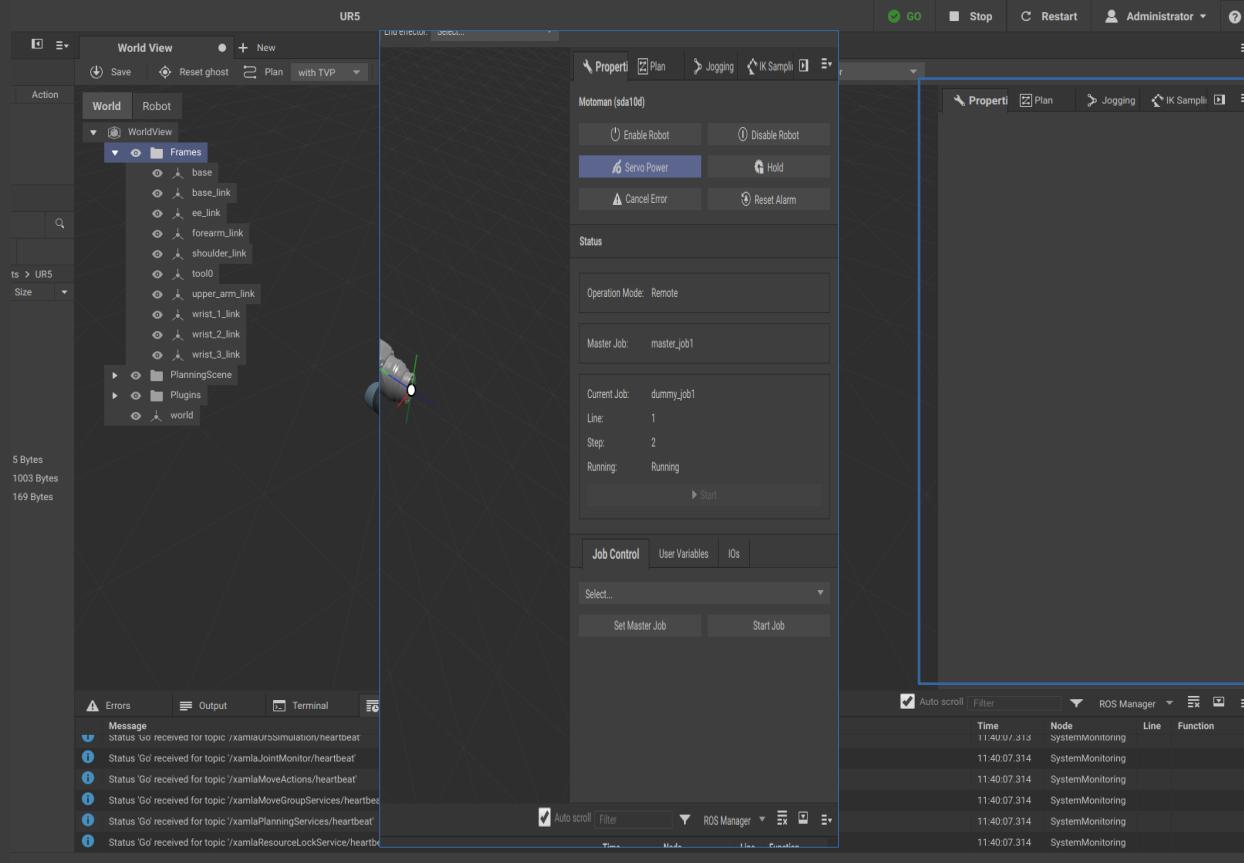


Sampling Resolution

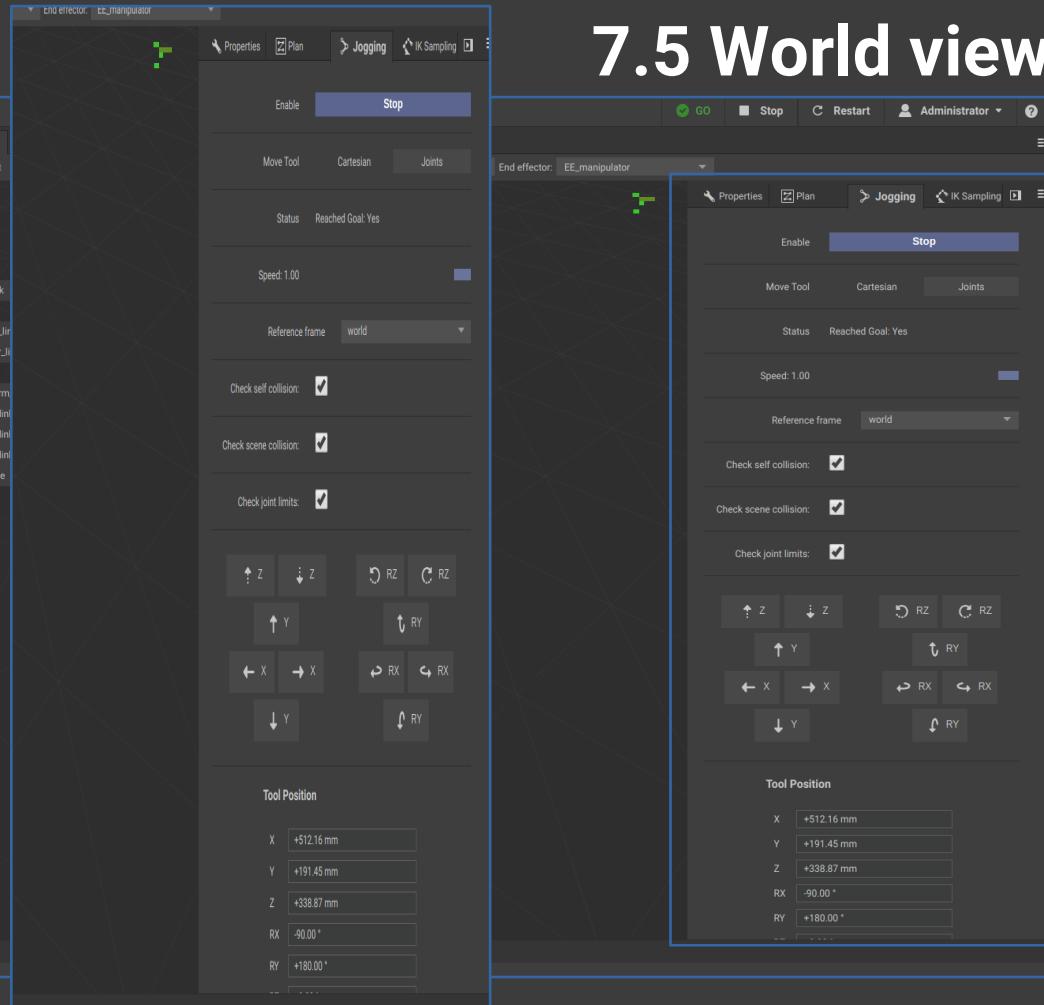


Move Supervised

7.4 World view



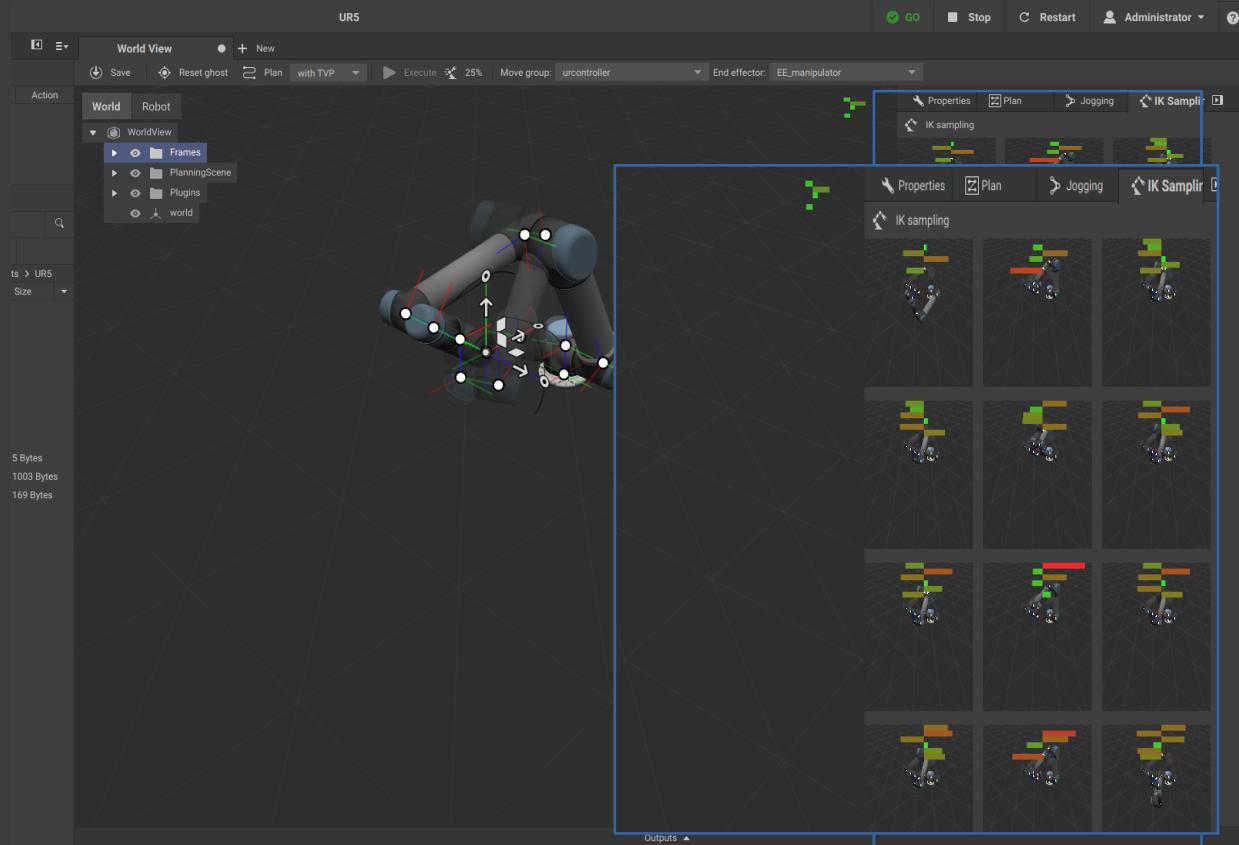
- ✓ Property view
- ✓ Detailed information of all datatypes in the tree view.
- ✓ Plugins



7.5 World view

- ✓ Property view
- ✓ Detailed information of all datatypes in the tree view.
- ✓ Plugins
- ✓ Jogging view
- ✓ Joint Space
- ✓ Cartesian Space

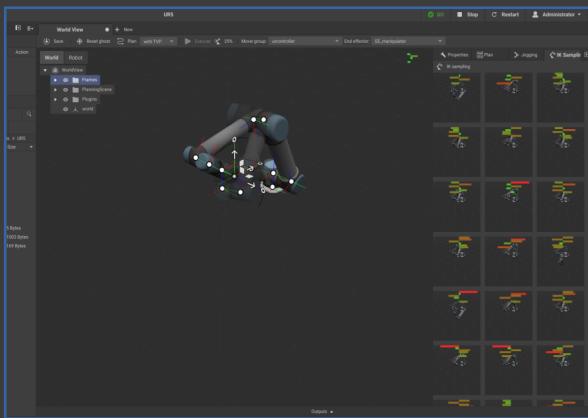
7.6 World view



- ✓ Property view
- ✓ Detailed information of all datatypes in the tree view.
- ✓ Plugins
- ✓ Jogging view
- ✓ Joint Space
- ✓ Cartesian Space
- ✓ IK-Sampling
- ✓ Find better joint values for Pose

7.8 World view

Conclusion:



- ✓ Positioning the robot
- ✓ Store data
- ✓ Try different things

- ✓ Jogging view
- ✓ Joint Space
- ✓ Cartesian Space

- ✓ IK-Sampling
- ✓ Find better joint values for Pose

8.1 Status Monitoring

UR5

System Status + New

Action System Process List

PID	User	Priority	Nice	Mem	CPU%	Mem%	Command
857	xamla	20	0	57.5 KB	70.0%	0.4%	luajit
60	xamla	20	0	333 KB	60.0%	2.1%	Rosvita
88	xamla	20	0	73.5 KB	30.0%	0.5%	dotnet
87	xamla	20	0	76.7 KB	20.0%	0.5%	dotnet
810	xamla	20	0	50.4 KB	10.0%	0.3%	move_group
813	xamla	20	0	50.0 KB	10.0%	0.3%	luajit
825	xamla	20	0	62.3 KB	10.0%	0.4%	luajit

Managed Process List

ID	PID	Name	Status	Command	Arguments	Action
0	290	roscore	Running	roscore		Stop Restart
2	808	robot_state_publ	Running	/opt/ros/kinetic/l	_name:=robot_s	Stop Restart
3	810	move_group	Running	/opt/ros/kinetic/l	_name:=move_g	Stop Restart
4	813	xamlaUr5Control	Running	/home/xamla/ca	_name:=xamlaU	Stop Restart
5	814	xamlaUr5Simulat	Running	/home/xamla/ca	_name:=xamlaU	Stop Restart
6	816	xamlaJointMonit	Running	/home/xamla/ca	_name:=xamlaJ	Stop Restart
7	825	xamlaJoint.Joggi	Running	/home/xamla/ca	_name:=xamlaJ	Stop Restart

Process Outputs

roscore (Running)

```
... logging to /home/xamla/.ros/log/3e29711e-803f-11e8-a8d4-00012e6ea834/ros.launch-roscore-a-controller-1-290.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

5 Bytes

Process Outputs

roscore (Running)

```
... logging to /home/xamla/.ros/log/3e29711e-803f-11e8-a8d4-00012e6ea834/ros.launch-roscore-a-controller-1-290.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```

2.67 KB

Process Outputs

move_group (Running)

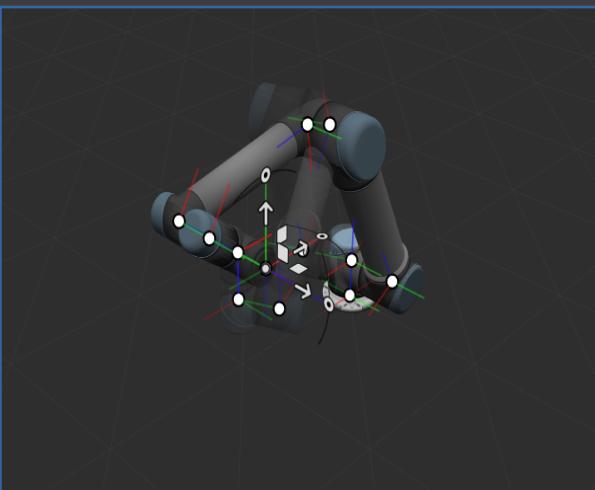
```
Loading 'move_group/MoveGroupMoveAction'...
Loading 'move_group/MoveGroupPickPlaceAction'...
Loading 'move_group/MoveGroupPlanService'...
Loading 'move_group/MovegroupQueryPlannersService'...
Loading 'move_group/MovegroupStateValidationService'...
[ INFO] [1530786980.338684229]: ****
[ INFO] [1530786980.338684229]: MoveGroup using:
[ INFO] [1530786980.338684229]:   - ApplyPlanningSceneService
[ INFO] [1530786980.338684229]:   - ClearOctomapService
[ INFO] [1530786980.338684229]:   - CartesianPathService
[ INFO] [1530786980.338684229]:   - ExecuteTrajectoryAction
[ INFO] [1530786980.338684229]:   - GetPlanningSceneService
[ INFO] [1530786980.338684229]:   - KinematicsService
[ INFO] [1530786980.338684229]:   - MoveAction
[ INFO] [1530786980.338684229]:   - PickPlaceAction
[ INFO] [1530786980.338684229]:   - MotionPlanService
[ INFO] [1530786980.338684229]:   - QueryPlannersService
[ INFO] [1530786980.338684229]:   - StateValidationService
[ INFO] [1530786980.338684229]: ****
[ INFO] [1530786980.33872077]: MoveGroup context using planning plugin ompl_interface/OMPLplanner
[ INFO] [1530786980.338733502]: MoveGroup context initialization complete
```

You can start planning now!

```
[ INFO] [1530787029.728922867]: Execution request received
[ INFO] [1530787033.716734129]: Completed trajectory execution with status SUCCEEDED ...
[ INFO] [1530787033.716837534]: Execution completed: SUCCEEDED
```

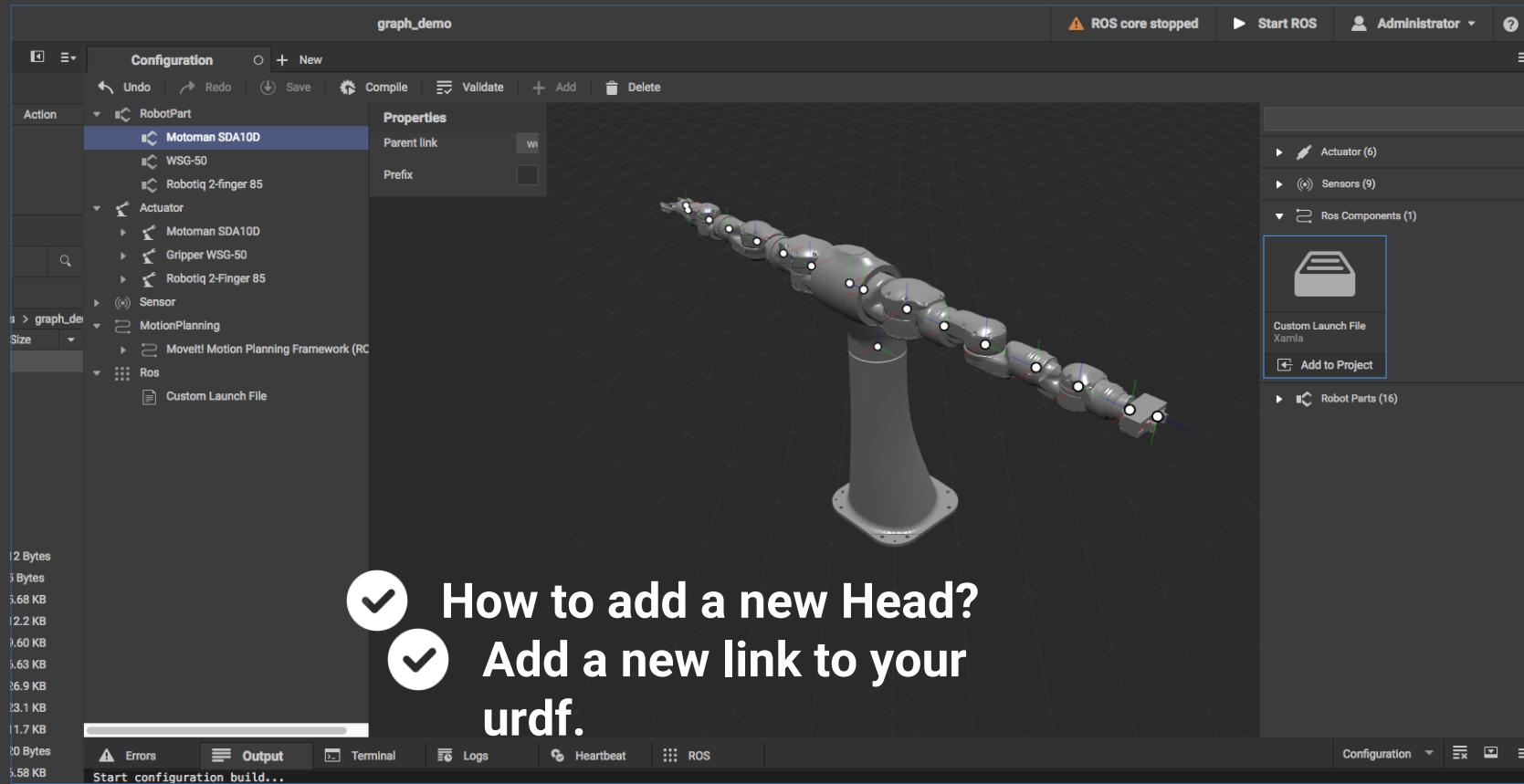
Outputs ▲

9.1 Customizing



- Add your own link to your setup**
- Add full robot part**
- Add ros package**
- Add your own hardware plugin**

9.2 Costum robot part



9.2 Costum robot part

The screenshot shows the RosVita Control software interface. On the left is a 'File Browser' pane with a tree view of files under 'Rosvita.Control'. A file named 'main.xacro' is selected. In the center is a code editor displaying XACRO XML code for a custom robot. On the right is a '3D View' pane showing a wireframe model of the robot's joints and links.

Runtime Slots (0)		
Slot	PID	Action

Save Compile

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <robot name="graph_demo" xmlns:xacro="http://ros.org/wiki/xacro" xmlns:xamla="http://www.xamla.com/">
3   <xacro:property name="pi" value="3.1415926535897931" />
4   <link name="world" />
5   <xacro:include filename="part_motoman_sda10d/sda10d_macro.xacro" xamla:xref="yaskawa$motoman_sda10d_1" />
6   <xacro:motoman_sda10d xamla:xref="yaskawa$motoman_sda10d_1" prefix="" parent="world">
7     <origin xyz="0 0 0" rpy="0 0 0" />
8   </xacro:motoman_sda10d>
9   <xacro:include filename="part_wsg50/weiss-wsg50.xacro" xamla:xref="weiss_robotics$wsg_50_1" />
10  <xacro:part_wsg50 xamla:xref="weiss_robotics$wsg_50_1" prefix="test" parent="arm_left_link_tool0">
11    <origin xyz="0 0 0" rpy="0 0 0" />
12  </xacro:part_wsg50>
13  <xacro:include filename="part_robotiq2finger85/part_robotiq2finger85.xacro" xamla:xref="robotiq$2finger85_1" />
14  <xacro:part_robotiq2finger85 xamla:xref="robotiq$2finger85_1" prefix="" parent="arm_right_link_tool0">
15    <origin xyz="0 0 0" rpy="0 0 0" />
16  </xacro:part_robotiq2finger85>
17 </robot>
```

3D View URDF

- Robot model
- Joints
- Links
- View Options

9.3 Costum robot part

The screenshot shows the xamla software interface for configuring a custom robot part. The main window is divided into several sections:

- Runtime Slots (0)**: A table with columns Slot, PID, and Action.
- Save** and **Compile** buttons.
- File Browser**: Shows the file structure under "home > xamla > Rosvita.Control". The file **main.xacro** is selected, showing its size as 1.02 KB.
- Code Editor**: Displays the XML code for the custom robot part. The code defines a robot with a base link, a link named "my_own_link" with a visual and collision geometry, and a joint named "my_joint" of type "fixed" connecting the base link to the custom link.
- 3D View**: A 3D visualization of the robot model, showing a wireframe representation of the custom link and its connection to the base.
- URD**: A tab for Unified Robot Description, showing options for joints and links.

```
10  <xacro:part_wsg50 xamla:xref="weiss_robotics$wsg_50_1" prefix="test" parent="arm_left_link_tool0">
11    <origin xyz="0 0 0" rpy="0 0 0" />
12  </xacro:part_wsg50>
13  <xacro:include filename="part_robotiq2finger85/part_robotiq2finger85.xacro" xamla:xref="robotiq$2finger85_1"
14  <xacro:part_robotiq2finger85 xamla:xref="robotiq$2finger85_1" prefix="" parent="arm_right_link_tool0">
15    <origin xyz="0 0 0" rpy="0 0 0" />
16  </xacro:part_robotiq2finger85>
17
18  <link name="my_own_link">
19    <visual>
20      <geometry>
21        <cylinder length="0.6" radius="0.2"/>
22      </geometry>
23      <material name="blue">
24        <color rgba="0 0 .8 1"/>
25      </material>
26    </visual>
27    <collision>
28      <geometry>
29        <cylinder length="0.6" radius="0.2"/>
30      </geometry>
31    </collision>
32  </link>
33
34  <joint name="my_joint" type="fixed">
35    <origin xyz="0 0 0.2" rpy="0 0 3.1416"/>
36    <parent link="_head_base"/>
37    <child link="my_own_link"/>
38  </joint>
39 </robot>
```

9.3 Costum robot part

```
15     <origin xyz="0 0 0" rpy="0 0 0" />
16 </xacro:part_robottiq2finger85>
17
18 <link name="my_own_link">
19   <visual>
20     <geometry>
21       <cylinder length="0.6" radius="0.2"/>
22     </geometry>
23     <material name="blue">
24       <color rgba="0 0 .8 1"/>
25     </material>
26   </visual>
27   <collision>
28     <geometry>
29       <cylinder length="0.6" radius="0.2"/>
30     </geometry>
31   </collision>
32 </link>
33
34 <joint name="my_joint" type="fixed">
35   <origin xyz="0 0 0.2" rpy="0 0 3.1416"/>
36   <parent link="_head_base"/>
37   <child link="my_own_link"/>
38 </joint>
```



Simple primitives



Mesh descriptions



Visual (Collada files)
Collision (step files)

How to connect to my model?

9.3 Costum robot part

```
15   <origin xyz="0 0 0" rpy="0 0 0" />
16 </xacro:part_robotiq2finger85>
17
18 <link name="my_own_link">
19   <visual>
20     <geometry>
21       <cylinder length="0.6" radius="0.2"/>
22     </geometry>
23     <material name="blue">
24       <color rgba="0 0 .8 1"/>
25     </material>
26   </visual>
27   <collision>
28     <geometry>
29       <cylinder length="0.6" radius="0.2"/>
30     </geometry>
31   </collision>
32 </link>
33
34 <joint name="my_joint" type="fixed">
35   <origin xyz="0 0 0.2" rpy="0 0 3.1416"/>
36   <parent link="_head_base"/>
37   <child link="my_own_link"/>
38 </joint>
39 </robot>
```



Connect with joints



Add to the chain



Parent



Child

Press compile and you are done!

9.3 Costum robot part

The screenshot shows the Acro software interface. On the left, there is a code editor with ROS XML code for a robot configuration. The code defines a robot with various joints, links, and a custom link named "my_own_link" with a blue cylinder visual and collision geometry. On the right, a 3D view shows a detailed model of a robotic arm with a large cylindrical end effector. The 3D view has a toolbar with options like "Robot model", "Joints", "Links", and "View Options". At the bottom, there is a status bar with tabs for Errors, Output, Terminal, Logs, Heartbeat, and ROS, and a message indicating successful compilation.

Ln 35, Col 26

Errors Output Terminal Logs Heartbeat ROS Xacro compilation succeeded Xacro Compiler

9.4 Costum robot part

The screenshot shows the RosVisualizer interface with the following components:

- Top Bar:** Includes tabs for "graph_demo", "ROS core stopped", "Start ROS", and "Administrator".
- Left Sidebar:** "File Browser" showing project structure with files like CMakeLists.txt, package.xml, part.json, and sda10d_cell.xacro.
- Central Area:** "sda10d_cell.xacro" file open in a code editor, displaying URDF XML code for a robot model.
- Right Area:** "3D View" showing a 3D visualization of a multi-tiered metal frame structure, representing the robot's physical model. The "URDF Output" tab is also visible.

9.4 Costum robot part

The screenshot shows the ROSVITA software interface. On the left, there is a code editor window titled "sda10d_cell.xacro" displaying the XML code for a robot part. On the right, there is a 3D viewer showing a geometric model of the part. A toolbar with various icons is visible at the top of the interface.

```
<robot name="sda10d_cell" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:macro name="sda10d_cell" params="prefix parent *origin">
    <xacro:property name="pi" value="3.1415926535897931" />
    <link name="${prefix}cell_body">
      <inertial>
        <mass value="1.0" />
        <!-- center of mass (com) is defined w.r.t. link local coordinate system -->
        <origin xyz="0.25 -0.5 0" />
        <inertia ixz="1.0" ixy="0.0" iyy="1.0" izz="0.0" ixz="1.0" />
      </inertial>
      <visual>
        <!-- visual origin is defined w.r.t. link local coordinate system -->
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <mesh filename="package://part_sda10d_cell/visual/cell.dae" scale="1.00 1.00 1.00"/>
        </geometry>
      </visual>
      <collision>
        <!-- collision origin is defined w.r.t. link local coordinate system -->
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <mesh filename="package://part_sda10d_cell/collision/cell.stl" scale="1.00 1.00 1.00"/>
        </geometry>
      </collision>
    </link>
    <link name="${prefix}cell_tool0"/>
    <joint name="${prefix}world_to_cell_body" type="fixed">
      <parent link="${parent}" />
      <child link="${prefix}cell_body" />
      <xacro:insert_block name="origin" />
    </joint>
    <joint name="${prefix}cell_body_to_tool0" type="fixed">
      <parent link="${prefix}cell_body" />
      <child link="${prefix}cell_tool0" />
      <origin xyz="0 0 0.15" rpy="0 0 0" />
    </joint>
    <gazebo reference="cell_body">
      <material>Gazebo/LightWood</material>
    </gazebo>
  </xacro:macro>
  <!--{BEGIN_XAMLA_STANDALONE_TEMPLATE}>
  <xacro:property name="pi" value="3.1415926535897931" />
  <link name="world" />
  <xacro:sda10d_cell prefix="" parent="world">
    <origin xyz="0 0 0" rpy="0 0 0" />
  </xacro:sda10d_cell>
  <!--{END_XAMLA_STANDALONE_TEMPLATE}-->

```

Ln 1, Col 1

The screenshot shows the ROSVITA software interface. On the left, there is a code editor window titled "Action" displaying the XML code for a robot part. On the right, there is a 3D viewer showing a geometric model of the part. A toolbar with various icons is visible at the top of the interface.

```
<robot name="sda10d_cell" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:macro name="sda10d_cell" params="prefix parent *origin">
    <xacro:property name="pi" value="3.1415926535897931" />
    <link name="${prefix}cell_body">
      <inertial>
        <mass value="1.0" />
        <!-- center of mass (com) is defined w.r.t. link local coordinate system -->
        <origin xyz="0.25 -0.5 0" />
        <inertia ixz="1.0" ixy="0.0" iyy="1.0" iyz="0.0" izz="1.0" />
      </inertial>
      <visual>
        <!-- visual origin is defined w.r.t. link local coordinate system -->
        <origin xyz="0 0 0" rpy="0 0 0" />
      </visual>
    </link>
  </xacro:macro>

```

✓ Prefix
✓ Make sure your joints have unique identifier

✓ Parent
✓ Where to attache to

✓ Origin
✓ Releative transformation to parent link

9.5 Costum robot part

The screenshot shows a CAD application window. On the left is a code editor displaying the XML definition of a robot part named "sda10d_cell". The XML includes definitions for joints, links, and collision models. On the right is a 3D view of the part, which appears to be a rectangular prism with a grid pattern. A toolbar at the top has buttons for Save and Compile.

```
<robot name="sda10d_cell" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:macro name="sda10d_cell" params="prefix parent *origin">
    <xacro:property name="pi" value="3.1415926535897931" />
    <link name="${prefix}cell_body">
      <inertial>
        <mass value="1.0" />
        <!-- center of mass (com) is defined w.r.t. link local coordinate system -->
        <origin xyz="0.25 -0.5 0" />
        <inertia ixz="1.0" ixy="0.0" iyy="1.0" izz="1.0" />
      </inertial>
      <visual>
        <!-- visual origin is defined w.r.t. link local coordinate system -->
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <mesh filename="package://part_sda10d_cell/visual/cell.dae" scale="1.00 1.00 1.00"/>
        </geometry>
      </visual>
      <collision>
        <!-- collision origin is defined w.r.t. link local coordinate system -->
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <mesh filename="package://part_sda10d_cell/collision/cell.stl" scale="1.00 1.00 1.00"/>
        </geometry>
      </collision>
    </link>
    <link name="${prefix}cell_tool0"/>
    <joint name="${prefix}world_to_cell_body" type="fixed">
      <parent link="${parent}" />
      <child link="${prefix}cell_body" />
      <xacro:insert_block name="origin" />
    </joint>
    <joint name="${prefix}cell_body_to_tool0" type="fixed">
      <parent link="${prefix}cell_body" />
      <child link="${prefix}cell_tool0" />
      <origin xyz="0 0 0.16" rpy="0 0 0" />
    </joint>
    <gazebo reference="cell_body">
      <material>Gazebo/LightWood</material>
    </gazebo>
  </xacro:macro>
  <!--@{BEGIN_XAMLA_STANDALONE_TEMPLATE}>
  <xacro:property name="pi" value="3.1415926535897931" />
  <link name="world" />
  <xacro:sda10d_cell prefix="" parent="world">
    <origin xyz="0 0 0" rpy="0 0 0" />
  </xacro:sda10d_cell>
  <!--@{END_XAMLA_STANDALONE_TEMPLATE}-->

```

The screenshot shows a code editor with two snippets of XML code. The top snippet is the same as the one in the first screenshot, defining a robot part "sda10d_cell". The bottom snippet is a template for a "cell_tool0" component, which defines a fixed joint between the "cell_body" and "cell_tool0" links, and specifies a "Gazebo/LightWood" material for the "cell_body". Both snippets use the XACRO macro language.

```
<!--@{BEGIN_XAMLA_STANDALONE_TEMPLATE}>
<xacro:macro name="sda10d_cell" params="prefix parent *origin">
  <xacro:property name="pi" value="3.1415926535897931" />
  <link name="${prefix}cell_body">
    <inertial>
      <mass value="1.0" />
      <!-- center of mass (com) is defined w.r.t. link local coordinate system -->
      <origin xyz="0.25 -0.5 0" />
      <inertia ixz="1.0" ixy="0.0" iyy="1.0" izz="1.0" />
    </inertial>
    <visual>
      <!-- visual origin is defined w.r.t. link local coordinate system -->
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://part_sda10d_cell/visual/cell.dae" scale="1.00 1.00 1.00"/>
      </geometry>
    </visual>
    <collision>
      <!-- collision origin is defined w.r.t. link local coordinate system -->
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://part_sda10d_cell/collision/cell.stl" scale="1.00 1.00 1.00"/>
      </geometry>
    </collision>
  </link>
  <link name="${prefix}cell_tool0"/>
  <joint name="${prefix}world_to_cell_body" type="fixed">
    <parent link="${parent}" />
    <child link="${prefix}cell_body" />
    <xacro:insert_block name="origin" />
  </joint>
  <joint name="${prefix}cell_body_to_tool0" type="fixed">
    <parent link="${prefix}cell_body" />
    <child link="${prefix}cell_tool0" />
    <origin xyz="0 0 0.16" rpy="0 0 0" />
  </joint>
  <gazebo reference="cell_body">
    <material>Gazebo/LightWood</material>
  </gazebo>
</xacro:macro>
<!--@{END_XAMLA_STANDALONE_TEMPLATE}>
```

9.5 Costum robot part

The screenshot shows the xamla IDE interface. The 'File Browser' panel on the left displays the directory structure: home > xamla > Rosvita.Control > library > robot_parts. Inside this folder, there are four files listed: CMakeLists.txt, package.xml, part.json, and sda10d_cell.xacro. The 'CMakeLists.txt' file is currently selected. The main workspace shows the 'package.xml' tab open, displaying the following XML code:

```
1<package>
2  <name>part_sda10d_cell</name>
3  <version>0.1.0</version>
4  <description>The sda10d cell description of Rosvita</description>
5  <maintainer email="andre.lemme@xamla.com">Andre Lemme</maintainer>
6  <license>BSD</license>
7  <url type="website">http://xamla.de</url>
8  <author email="andre.lemme@xamla.com">Andre Lemme</author>
9
10 <buildtool_depend>catkin</buildtool_depend>
11
12 <export>
13 </export>
14</package>
```



ROS package



CMakeLists.txt



Package.xml

9.5 Costum robot_part

The screenshot shows a code editor interface with the following structure:

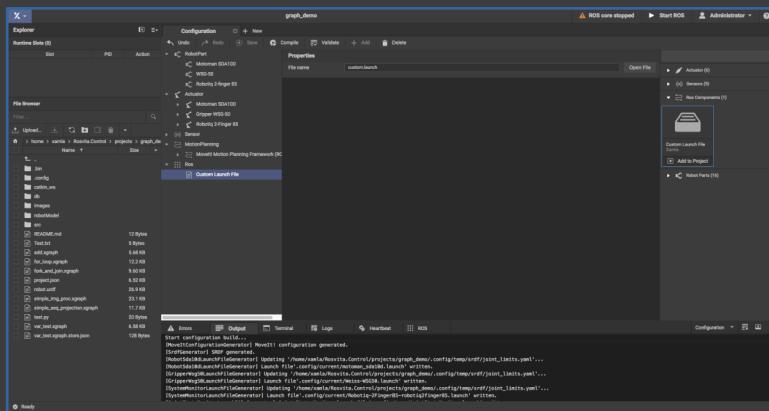
- Explorer** tab: Shows "Runtime Slots (0)".
- File Browser** tab: Shows the directory structure:
 - home > xamla > Rosvita.Control > library > robot_parts
 - collision
 - visual
 - CMakeLists.txt (115 Bytes)
 - package.xml (418 Bytes)
 - part.json** (541 Bytes) - This file is selected.
 - sda10d_cell.xacro (1.85 KB)
- part.json** tab: Displays the JSON content of the selected file.

```
1: {
2:     "id": "object$sda10d_cell",
3:     "displayName": "SAD10D Cell",
4:     "thumbnailUrl": "/images/part-thumbs/xamla-sda-10-cell.png",
5:     "modelFile": "./sda10d_cell.xacro",
6:     "manufacturer": {
7:         "name": "Xamla",
8:         "url": "http://www.xamla.com/"
9:     },
10:    "maintainer": {
11:        "organization": "Xamla",
12:        "url": "http://www.xamla.com/",
13:        "email": "support@xamla.com"
14:    },
15:    "partType": "object",
16:    "baseLink": "${prefix}cell_body",
17:    "endLinks": ["${prefix}cell_tool0"],
18:    "macroName": "sda10d_cell"
19: }
```

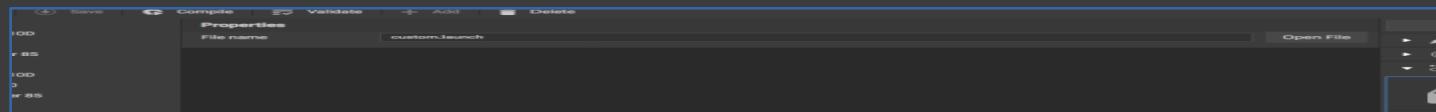
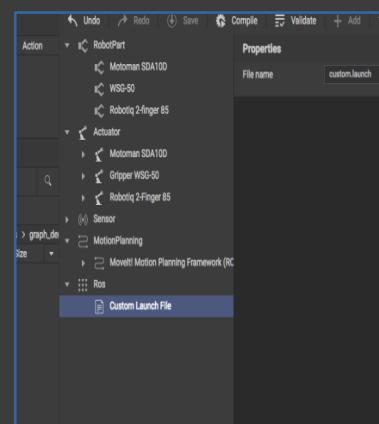
- ✓ ROS package
- ✓ CMakeLists.txt
- ✓ Package.xml

- ✓ ROSVITA additions
- ✓ Part.json

9.6 Costum launch files

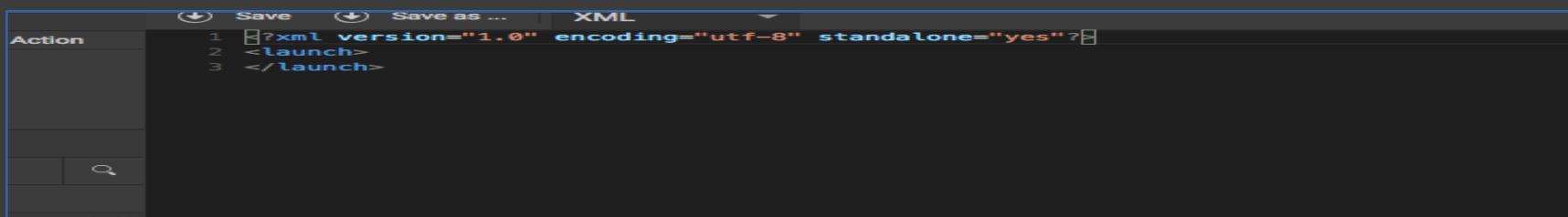


Select ROS Launch File in ros vita library
Open launch file



9.6 Costum launch files

-  **Select ROS Launch File in ros vita library**
-  **Open launch file**
-  **Add any node you want ros vita to launch for you.**



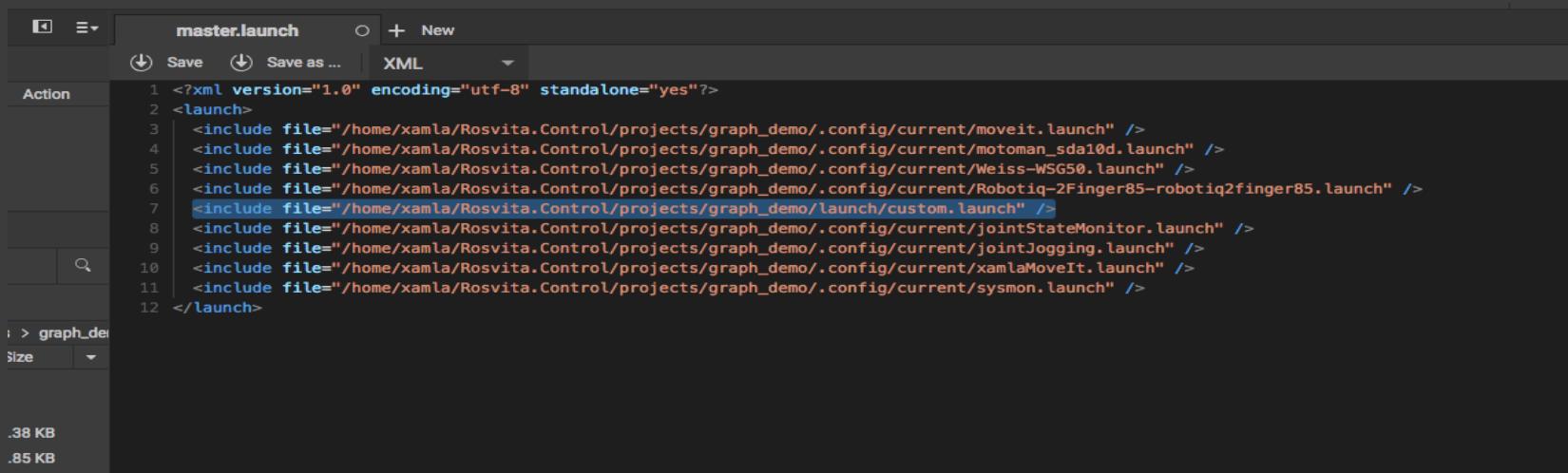
A screenshot of a code editor window titled "Action". The window has a toolbar with "Save" and "Save as ...". The main area shows the XML code for a launch file:

```
1 <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2 <launch>
3 </launch>
```

9.6 Costum launch files

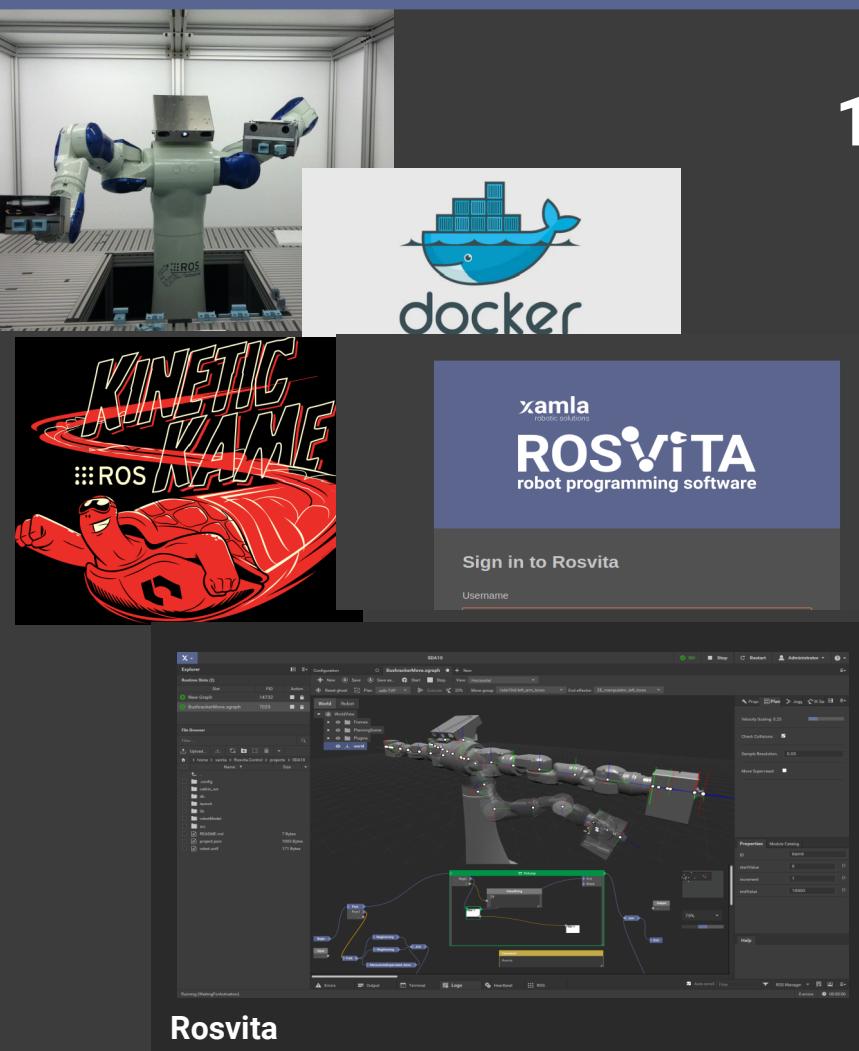


- Select ROS Launch File in ros vita library
- Open launch file
- Add any node you want ros vita to launch for you.



```
master.launch
Action
1 <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2 <launch>
3   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/moveit.launch" />
4   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/motoman_sda10d.launch" />
5   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/Weiss-WSG50.launch" />
6   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/Robotiq-2Finger85-robotiq2finger85.launch" />
7   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/launch/custom.launch" />
8   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/jointStateMonitor.launch" />
9   <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/jointJogging.launch" />
10  <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/xamlaMoveIt.launch" />
11  <include file="/home/xamla/Rosvita.Control/projects/graph_demo/.config/current/sysmon.launch" />
12 </launch>
```

.38 KB
.85 KB



10.1 ROSVITA

- ✓ Based on ROS
- ✓ Library of robot parts and drivers included
- ✓ MotionPlanning
- ✓ MoveIt configuration and extras
- ✓ Customizing on multiple levels
- ✓ Robot parts
- ✓ Launch files
- ✓ Plugins

Quotes

“Technology is nothing. What's important is that you have a faith in people, that they're basically good and smart, and if you give them tools, they'll do wonderful things with them.”

Steve Jobs



Thank you for your attention!