Introduction to Robot Operating System (ROS 1)

Dr. Essa Alghannam

ما هو XACRO؟

Xacro (اختصار لـ XML Macro Language for Robotics) هي أداة تُستخدم لتبسيط إنشاء وتعديل ملفات وصف الروبوت (URDFs) في نظام ROS1 Noetic (وغيره من توزيعات ROS).

ملفات URDF مكتوبة بلغة XML، وتصف البنية الفيزيائية للروبوت، بما في ذلك وصلاته (Links)، ومفاصله (Joints)، وخصائصه البصرية. تُعاني ملفات URDF الكبيرة والمعقدة من مشاكل الصيانة وقابلية القراءة. هنا يأتي دور Xacro لحل هذه المشاكل.

كيف يعمل Xacro؟

Xacro تُوسّع لغة XML المستخدمة في URDF من خلال إضافة إمكانيات:

1. المعلمات (Properties): يمكنك تعريف متغيرات قابلة للتعديل بسهولة. هذه المتغيرات تمثل قيمًا مثل أبعاد الأجزاء، وكتلتها، أو أي خصائص أخرى. يمكنك بعد ذلك استخدام هذه المتغيرات داخل ملف Xacro لوصف مكونات الروبوت. بذلك يمكنك تغيير قيم هذه المتغيرات لإنشاء تصميمات مختلفة بنفس الملف الأساسي.
2. المعايير الكلية (Macros): يسمح Xacro بتعريف "معايير كلية" (Macros)، وهي عبارة عن كتل من التعليمات البرمجية XML قابلة لإعادة الاستخدام. يمكن تعريف معيار كلّي لوصف جزء معين من الروبوت (مثل عجلة، وصلة، أو مفصل) واستخدامه في أماكن متعددة في الملف. هذا يُقلل من تكرار الكود ويُسهّل عملية التعديل.
3. الدمج (Inclusion): يمكنك دمج ملفات Xacro أخرى داخل ملف Xacro الرئيسي، ما يُسهّل تنظيم المشروع وفصل ملفات الوصف إلى وحدات منطقية أصغر.

مثال توضيحي:

لنفرض نريد وصف روبوت بسيط يحتوي على ذراع مكونة من وصلتين ومفصلين. بدون Xacro، سيكون علينا كتابة كل خصائص كل وصلة ومفصل في ملف URDF واحد، ما سيُؤدي إلى كود طويل ومعقد. باستخدام Xacro، يمكنك تقسيم الوصف إلى ملفات أصغر:

1. `arm_link.xacro` : يحتوي على معيار كلّي لوصف وصلة الذراع.
2. `arm_joint.xacro` : يحتوي على معيار كلّي لوصف مفصل الذراع.
3. `robot_description.xacro` : يستخدم المعايير الكلية من الملفين السابقين لوصف الروبوت بأكمله.

Example:

1. Navigate the **scr** folder of the **workspace 'mycatkin_ws**'. And run:
   essa@essa:~/mycatkin_ws$ catkin_create_pkg simple_cube rospy std_msgs geometry_msgs roscpp xacro urdf
2. go back to the workspace folder and run:
   essa@essa:~/mycatkin_ws$ catkin_make
3. Add the following code named "**simple_cube.xacro**" to the **urdf** folder of the package:
   ~/mycatkin_ws/src/simple_cube/**urdf**

~/mycatkin_ws/src/simple_cube/**urdf**/simple_cube.xacro

```
<?xml version="1.0"?>
<robot name="simple_cube" xmlns:xacro="http://www.ros.org/wiki/xacro">
 <xacro:property name="size" value="0.2"/>
 <xacro:property name="mass" value="1.0"/>
<xacro:property name="x" value="1.0"/>
 <xacro:property name="y" value="0.5"/>
 <xacro:property name="z" value="0.0"/>
 <xacro:property name="roll" value="0.0"/>
 <xacro:property name="pitch" value="0.0"/>
 <xacro:property name="yaw" value="0.0"/>
```

Commented [EA1]: This line is the root element of the XML file. It declares that this XML describes a robot and gives it the name "simple_cube". This name will be important for identifying the robot in your ROS system.

Commented [EA2]: This line defines a namespace for the Xacro elements. `xmlns` stands for "XML namespace," and this declaration tells the Xacro processor that elements with the prefix `xacro:` belong to the Xacro language. It's essential for the Xacro preprocessor to understand the special Xacro commands.

```
<link name="base_link">
  <visual>
    <geometry>
      <box size="${size} ${size} ${size}"/>
    </geometry>
    <origin rpy = "${roll} ${pitch} ${yaw}" xyz = "${x} ${y} ${z}" />
    <material name="red"/>
  </visual>
  <collision>
    <geometry>
      <box size="${size} ${size} ${size}"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="${mass}"/>
    <inertia ixx="0.01" ixy="0.0" ixz="0.0" iyy="0.01" iyz="0.0" izz="0.01"/>
  </inertial>
</link>
</robot>
```

Commented [EA3]: This specifies that the visual representation of the cube should be red.

Commented [EA4]: This defines the inertia matrix of the cube. The values provided are placeholders and might need adjustment for accurate physics simulation, depending on the cube's density and distribution of mass. These values represent the moments and products of inertia.

4. ADD Launch file:

~/mycatkin_ws/src/simple_cube/launch/launch.launch

```
<?xml version="1.0"?>
<launch>
  <param name="robot_description" command="$(find xacro)/xacro '$(find simple_cube)/urdf/simple_cube.xacro'"/>

  <node name="robot_state_publisher"
      pkg="robot_state_publisher"
```

Commented [EA5]: **define parameters named robot_description that can be accessed by nodes in the ROS network.**
The robot_description parameter is commonly used in ROS to hold the description of the robot model
command="$(find xacro)/xacro '$(find simple_cube)/urdf/simple_cube.xacro'":
This command is executed to generate the robot description from a XACRO file.
$(find xacro): This part of the command locates the XACRO package within the ROS environment, which is a tool used to simplify the creation of URDF files.
/xacro: This indicates the executable that processes the XACRO file.
'$(find simple_cube)/urdf/simple_cube.xacro': This part specifies the path to the XACRO file that describes the robot model. The find command locates the simple_cube package and points to its URDF folder.

```
      type="robot_state_publisher"
      output="screen">
  </node>
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
 simple_cube)/config/rviz.rviz" />

 </launch>
```

roslaunch simple_cubelaunch.launch

Another way:

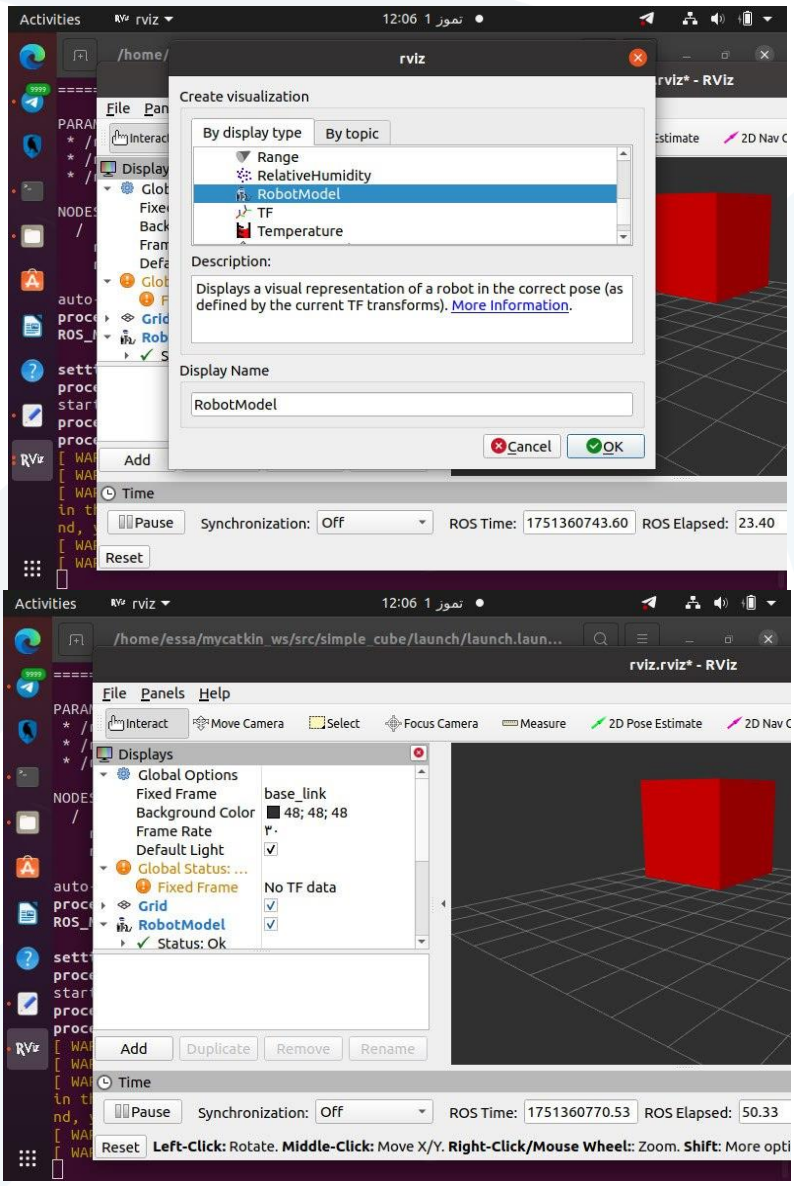
~/mycatkin_ws/src/simple_cube/**urdf**/parameters.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
          <xacro:property name="size" value="2"/>
          <xacro:property name="mass" value="1.0"/>
          <xacro:property name="x" value="1.0"/>
         <xacro:property name="y" value="0.5"/>
         <xacro:property name="z" value="0.0"/>
          <xacro:property name="roll" value="0.0"/>
          <xacro:property name="pitch" value="0.0"/>
          <xacro:property name="yaw" value="0.0"/>


</robot>
```

~/mycatkin_ws/src/simple_cube/**urdf**/simple_cube1.xacro

```
 <?xml version="1.0"?>
 <robot name="simple_cube" xmlns:xacro="http://www.ros.org/wiki/xacro">


  <xacro:include filename="$(find simple_cube)/urdf/parameters.xacro"/>
```

```
  <link name="base_link">
    <visual>
      <geometry>
        <box size="${size} ${size} ${size}"/>
      </geometry>
      <origin rpy = "${roll} ${pitch} ${yaw}" xyz = "${x} ${y} ${z}" />
      <material name="red"/>
    </visual>
    <collision>
      <geometry>
        <box size="${size} ${size} ${size}"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="${mass}"/>
      <inertia ixx="0.01" ixy="0.0" ixz="0.0" iyy="0.01" iyz="0.0" izz="0.01"/>
    </inertial>
  </link>
</robot>
```


Activities
rviz
12:06 1 تموز
rviz
Create visualization
By display type
By topic
Range
RelativeHumidity
RobotModel
TF
Temperature
Description:
Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.
Display Name
RobotModel
Cancel
OK
Add
Time
Pause
Synchronization:
Off
ROS Time:
1751360743.60
ROS Elapsed:
23.40
Reset
Activities
rviz
12:06 1 تموز
/home/essa/mycatkin_ws/src/simple_cube/launch/launch.laun...
rviz.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Displays
Global Options
Fixed Frame
base_link
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: ...
Fixed Frame
No TF data
Grid
RobotModel
Status: Ok
Add
Duplicate
Remove
Rename
Time
Pause
Synchronization:
Off
ROS Time:
1751360770.53
ROS Elapsed:
50.33
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More opti