UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERIA

IMPI INSTITUTO DE INGENIERIA MECANICA Y PRODUCCION INDUSTRIAL

Representación Gráfica

Fundamentos de Robótica Industrial

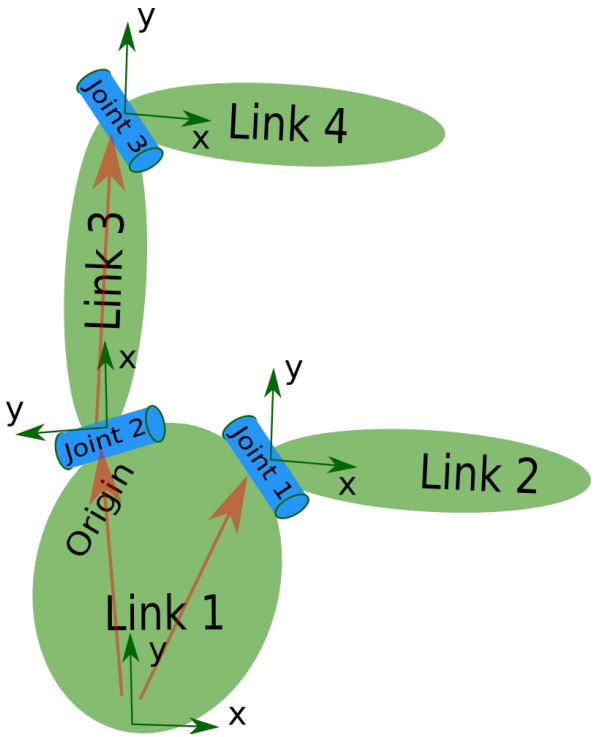
Introducción

- Hoy comentaremos los métodos de representación gráfica que pueden utilizarse en ROS
- Dichos métodos nos permitirán no solo una visualización más simple sino que serán un insumo para el diseño y evaluación de manipuladores
    - Dimensiones
    - Repetición de experimentos
    - Propiedades físicas

XML Unified Robot Description Format (URDF)

- El formato de descripción de robot unificado (URDF) es una especificación XML para describir un robot.
- Esta especificación es lo más general posible, pero no puede describir todos los robots. La principal limitación es que solo se pueden representar estructuras de árbol, descartando todos los robots paralelos.
- Además, asume que el robot consta de enlaces rígidos conectados por juntas; los elementos flexibles no son compatibles.
- La especificación cubre:
  - Descripción cinemática y dinámica del robot.
  - Representación visual del robot.
  - Modelo de colisión del robot.

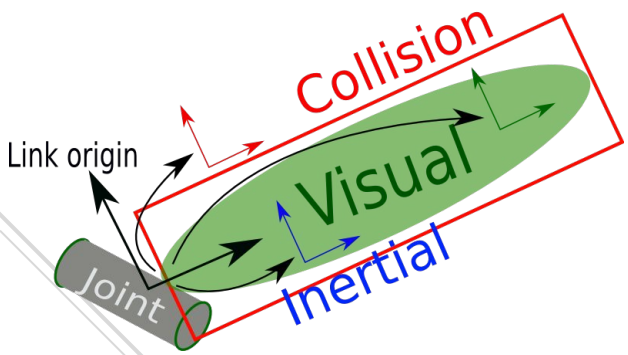
XML Unified Robot Description Format (URDF)



Elemento: Eslabón

El elemento de enlace describe un cuerpo rígido con inercia, características visuales y propiedades de colisión

Aquí hay un ejemplo:




```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

Elemento: Eslabón - <inertial>

<inertial> (opcional: el valor predeterminado es masa cero e inercia cero si no se especifica)

- Determina la masa del eslabón (en kg.), la posición de su centro de masa (las distribuciones de masa se toman como uniformes) y sus propiedades de inercia.

<**origin**> (opcional)

- Esta pose (traslación, rotación) describe la posición y la orientación del marco C del centro de masa del vínculo en relación con el marco del vínculo L.

xyz (opcional: por defecto es vector cero): Representa el vector de posición desde Lo (el origen del sistema del eslabón) hasta Co (el centro de masa del eslabón) como x Ľx + y Ľy + z Ľz, donde Ľx, Ľy, Ľz son los vectores unitarios ortogonales L del sistema del eslabón.

rpy (opcional): Representa la orientación de los vectores unitarios de C Ĉx, Ĉy, Ĉz en relación con el sistema L como una secuencia de rotaciones de Euler (r p y) en radianes. Nota: Ĉx, Ĉy, Ĉz no necesitan estar alineados con los ejes principales de inercia del eslabón.

<**masa**> La masa del enlace está representada por el atributo de valor de este elemento.

<**inercia**> Los momentos de inercia de este eslabón **ixx, iyy, izz** y los productos de inercia **ixy, ixz, iyz** sobre Co (el centro de masa del vínculo) para los vectores unitarios Ĉx, Ĉy, Ĉz fijados en el marco del centro de masa C. Nota: la orientación de Ĉx, Ĉy, Ĉz relativa a Ľx, Ľy, Ľz se especifica mediante los valores rpy en la etiqueta <origin>.La forma más sencilla de evitar problemas de compatibilidad asociados con la convención de signos negativos para el producto de inercia es alinear Ĉx, Ĉy, Ĉz con las direcciones principales de inercia para que todos los productos de inercia sean cero.

Elemento: Eslabón - <visual>

<visual> (opcional)

- Determina las propiedades visuales del eslabón. Este elemento especifica la forma del objeto (caja, cilindro, etc.) para fines de visualización. Pueden existir varias instancias de etiquetas <visual> para el mismo enlace. La unión de la geometría que definen forma la representación visual del eslabón.

Name (Opcional)

- Especifica un nombre para una parte de la geometría de un vínculo. Esto es útil para poder referirse a bits específicos de la geometría de un enlace.

<**origin**> (opcional)

- Determina el sistema de referencia del elemento visual con respecto al sistema de referencia del eslabón.

  xyz (opcional: por defecto es vector cero): Representa el desplazamiento x, y, z.

  rpy (opcional): Representa los ángulos de balanceo, cabeceo y guiñada del eje fijo en radianes.

Elemento: Eslabón - <visual>

<visual> (opcional)

<geometría> (obligatorio)

- Determina la forma del objeto visual. Este puede ser uno de los siguientes:

  <box> El atributo de tamaño contiene las tres longitudes laterales de la caja. El origen de la caja está en su centro.

  <cylinder> Especifica el radio y la longitud. El origen del cilindro está en su centro.

  <esfera> Especifica el radio. El origen de la esfera está en su centro.

  <malla> Un elemento trimesh especificado por un nombre de archivo y una escala opcional que escala el cuadro delimitador alineado con el eje de la malla. Cualquier formato de geometría es aceptable, pero la compatibilidad de aplicaciones específicas depende de la implementación. El formato recomendado para obtener la mejor compatibilidad con texturas y colores son los archivos .dae.

<material> (opcional)

- Determina el material del elemento visual. Se permite especificar un elemento material fuera del objeto 'enlace', en el elemento 'robot' de nivel superior. Desde dentro de un elemento de enlace, puede hacer referencia al material por su nombre.

  name nombre del material

  <color> (opcional) rgba El color de un material especificado por un conjunto de cuatro números que representan rojo/verde/azul/alfa, cada uno en el rango de [0,1].

  <textura> (opcional) La textura de un material se especifica mediante un nombre de archivo

Elemento: Eslabón - <collision>

<collision> (opcional)

- Describe las propiedades de colisión de un eslabón. Puede ser diferente de las propiedades visuales, por ejemplo, a menudo se utilizan modelos de colisión más simples para reducir el tiempo de cálculo. Pueden existir varias instancias de etiquetas <collision> para el mismo enlace. La unión de la geometría que definen forma la representación de colisión del enlace.

Name: (Opcional)

- Especifica un nombre para una parte de la geometría de un vínculo. Esto es útil para poder referirse a bits específicos de la geometría de un enlace.

<origin> (opcional)

- El marco de referencia del elemento de colisión, relativo al marco de referencia del enlace.

  xyz (opcional: por defecto es vector cero): Representa el desplazamiento x, y, z.

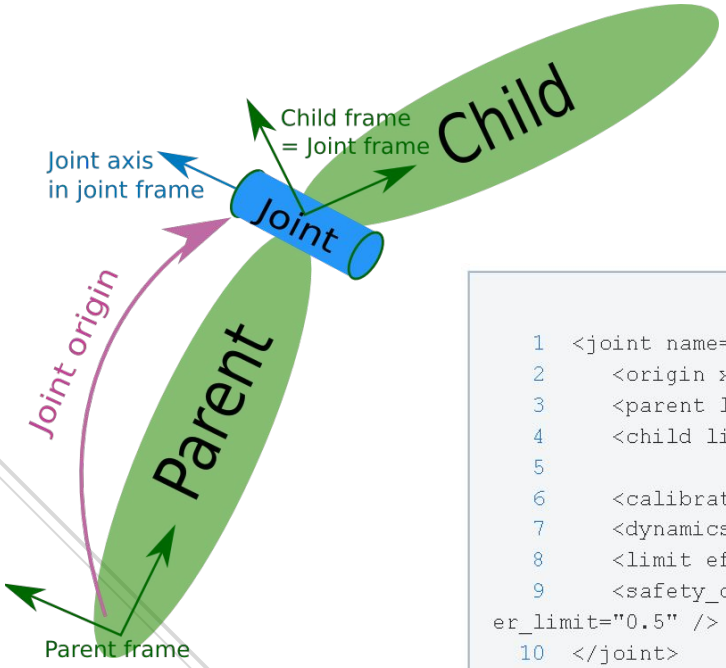
  rpy (opcional): Representa los ángulos de Roll, Pitch y Yaw del eje fijo en radianes.

<geometry>

- Igual a la descripción de la geometría en el elemento visual .

Elemento: Junta

El elemento de junta describe la cinemática y la dinámica de la unión y también especifica los límites de seguridad de la unión.




```
  1  <joint name="my_joint" type="floating">
  2     <origin xyz="0 0 1" rpy="0 0 3.1416"/>
  3     <parent link="link1"/>
  4     <child link="link2"/>
  5
  6     <calibration rising="0.0"/>
  7     <dynamics damping="0.0" friction="0.0"/>
  8     <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
  9     <safety_controller k_velocity="10" k_position="15" soft_lower_limit="-2.0" soft_upp
er_limit="0.5" />
 10  </joint>
```

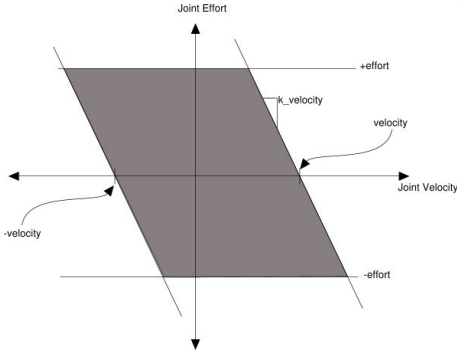
Elemento: Junta

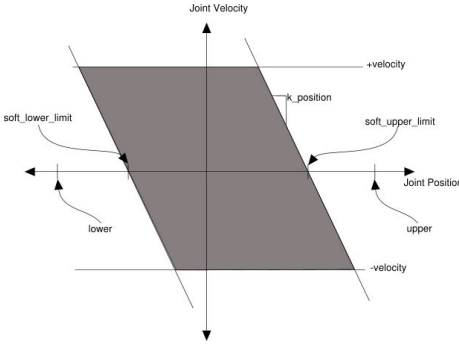
El **elemento de junta describe la cinemática y la dinámica de la unión** y también especifica los límites de seguridad de la unión.

Límites de seguridad: Manejado por la etiqueta “limit effort”. Un controlador no puede comandar un esfuerzo de más de 30 -en el ejemplo- N (Nm en revolución) en la articulación. Si el controlador intenta ordenar un esfuerzo más allá del límite de esfuerzo, la magnitud del esfuerzo se trunca.

El límite de velocidad especifica los límites de la magnitud de la velocidad de la articulación (m/s para prismática, rad/s para revolución). El límite de velocidad se hace cumplir limitando el esfuerzo ordenado de modo que no se pueda aplicar ningún esfuerzo para empujar la articulación más allá del límite de velocidad, y se aplica un esfuerzo de amortiguación si la articulación se mueve a una velocidad más allá del límite. El término k_velocity determina la escala de la limitación de esfuerzo.

Safety_length_min y safety_length_max determinan las ubicaciones de los límites de unión de software (los límites "soft"). El límite “soft” superior es el límite de hardware superior menos la safety_length_max y el límite “soft” inferior es el límite de hardware inferior más Safety_length_min. Cuando la articulación está cerca de los límites “soft”, las posibles velocidades se limitan para evitar que la posición cruce los límites “soft”. El término k_position determina la escala de la limitación de velocidad.





Elemento: Junta - Atributos

El elemento junta tiene dos atributos:

Name (requerido)

- Especifica un nombre único de la articulación.

type (requerido)

- Especifica el tipo de unión, donde el tipo puede ser uno de los siguientes:
  - **revolute** — una articulación de bisagra que gira a lo largo del eje y tiene un rango limitado especificado por los límites superior e inferior.
  - **continuous** — una junta de bisagra continua que gira alrededor del eje y no tiene límites superior e inferior.
  - **prismatic** — una junta deslizante que se desliza a lo largo del eje y tiene un rango limitado especificado por los límites superior e inferior.
  - **fixed**: esto no es realmente una articulación porque no se puede mover. Todos los grados de libertad están bloqueados. Este tipo de articulación no requiere **<axis>**, **<calibration>**, **<dynamics>**, **<limits>** o **<safety_controller>**
  - **floating**: esta articulación permite el movimiento para todos los 6 grados de libertad.
  - **planar** — esta articulación permite el movimiento en un plano perpendicular al eje.

Elemento: Junta - Elementos

El elemento de unión tiene los siguientes elementos:

<origin> (opcional)

- Esta es la transformación del eslabón principal al eslabón secundario. La unión se encuentra en el origen del eslabón secundario, como se muestra en la figura anterior.

  xyz (opcional: por defecto es vector cero): Representa el desplazamiento x, y, z. Todas las posiciones se especifican en metros.

  rpy (opcional: predeterminado en vector cero): Representa la rotación alrededor de un eje fijo: primero rueda alrededor de x, luego alrededor de y, y finalmente gira alrededor de z. Todos los ángulos se especifican en radianes.

<parent> (obligatorio)

- Nombre del eslabón principal con atributo obligatorio:

  link: El nombre del eslabón que es el “padre” de este eslabón en la estructura de árbol del robot.

<child> (requerido)

- Nombre de eslabón secundario con atributo obligatorio:

  link: El nombre del eslabón que es el eslabón secundario.

Elemento: Junta - Elementos

<axis> (opcional: por defecto es (1,0,0))

- El eje de junta especificado en el sistema de la junta. Este es el eje de rotación para las articulaciones giratorias, el eje de traslación para las articulaciones prismáticas y la superficie normal para las articulaciones planas. El eje se especifica en el sistema de referencia conjunto. Las uniones fijas y flotantes no utilizan el campo del eje.

  xyz (requerido): Representa los componentes (x, y, z) de un vector. El vector debe ser normalizado.

<calibration> (opcional)

- Las posiciones de referencia de la articulación, utilizadas para calibrar la posición absoluta de la articulación.

  rising (opcional): Cuando la articulación se mueve en una dirección positiva, esta posición de referencia activará un flanco ascendente.

  falling (opcional): Cuando la articulación se mueve en una dirección positiva, esta posición de referencia activará un borde descendente.

Elemento: Junta - Elementos

<dynamics> (opcional)

- Un elemento que especifica las propiedades físicas de la junta. Estos valores se utilizan para especificar las propiedades de modelado de la articulación, particularmente útiles para la simulación.

damping (opcional, predeterminado en 0): El valor de amortiguamiento físico de la junta (en newton-segundos por metro [N·s/m] para juntas prismáticas, en newton-segundos por radianes [N·m·s/rad] para juntas giratorias).

friction (opcional, por defecto es 0): El valor físico de fricción estática de la junta (en newtons [N] para juntas prismáticas, en newton-metros [N·m] para juntas giratorias).

<límite> (requerido solo para articulación angular y prismática)

- Un elemento puede contener los siguientes atributos:

lower (opcional, por defecto es 0): Un atributo que especifica el límite inferior de la articulación (en radianes para articulaciones giratorias, en metros para articulaciones prismáticas). Omitir si la junta es continua.

upper (opcional, por defecto es 0): Un atributo que especifica el límite superior de la articulación (en radianes para articulaciones giratorias, en metros para articulaciones prismáticas). Omitir si la junta es continua.

effort (requerido): Un atributo para hacer cumplir el esfuerzo conjunto máximo (|esfuerzo aplicado| < |esfuerzo|). Ver límites de seguridad.

velocity (requerido): Un atributo para hacer cumplir la velocidad máxima de articulación (en radianes por segundo [rad/s] para articulaciones giratorias, en metros por segundo [m/s] para articulaciones prismáticas).

Elemento: Junta - Elementos

<mimic> (opcional)

- Esta etiqueta se utiliza para especificar que la articulación definida imita otra articulación existente. El valor de esta unión se puede calcular como *value = multiplier * other_joint_value + offset*

  Atributos esperados y opcionales:

  joint (requerido): Esto especifica el nombre de la unión a imitar.

  multiplier (opcional: Especifica el factor multiplicativo en la fórmula anterior.

  offset (opcional): Especifica el desplazamiento para agregar en la fórmula anterior. El valor predeterminado es 0 (radianes para articulaciones giratorias, metros para articulaciones prismáticas)

<safety_controller> (opcional)

Un elemento puede contener los siguientes atributos:

soft_lower_limit (opcional, por defecto es 0: Un atributo que especifica el límite inferior de la articulación donde el controlador de seguridad comienza a limitar la posición de la articulación. Este límite debe ser mayor que el límite inferior de la junta (ver arriba).

soft_upper_limit (opcional, por defecto es 0): Un atributo que especifica el límite superior de la unión donde el controlador de seguridad comienza a limitar la posición de la unión. Este límite debe ser menor que el límite superior de la junta (ver arriba).

k_position (opcional, por defecto es 0): Un atributo que especifica la relación entre la posición y los límites de velocidad.

k_velocidad (obligatorio): Un atributo que especifica la relación entre el esfuerzo y los límites de velocidad.

Elemento: Transmisión

El elemento de transmisión es una extensión del modelo de descripción de robot URDF que se utiliza para describir la relación entre un actuador y una articulación. Esto permite modelar conceptos tales como relaciones de transmisión y enlaces paralelos. Una transmisión transforma las variables de esfuerzo/flujo de modo que su producto, la potencia, permanezca constante. Múltiples actuadores pueden estar vinculados a múltiples articulaciones a través de una transmisión compleja.

```
<transmission name="simple_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="foo_joint">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="foo_motor">
    <mechanicalReduction>50</mechanicalReduction>
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </actuator>
</transmission>
```

Elemento: Transmisión - Atributos

El elemento de transmisión tiene un atributo:

Name (requerido)

- Especifica el nombre único de una transmisión.

Elemento: Transmisión - Elementos

La transmisión tiene los siguientes elementos:

<type>

- Especifica el tipo de transmisión.

<joint>

- Una junta a la que está conectada la transmisión. La unión se especifica por su atributo de nombre y los siguientes subelementos:

  <hardwareInterface>: Especifica una interfaz de hardware de espacio conjunto admitida.

<actuator>

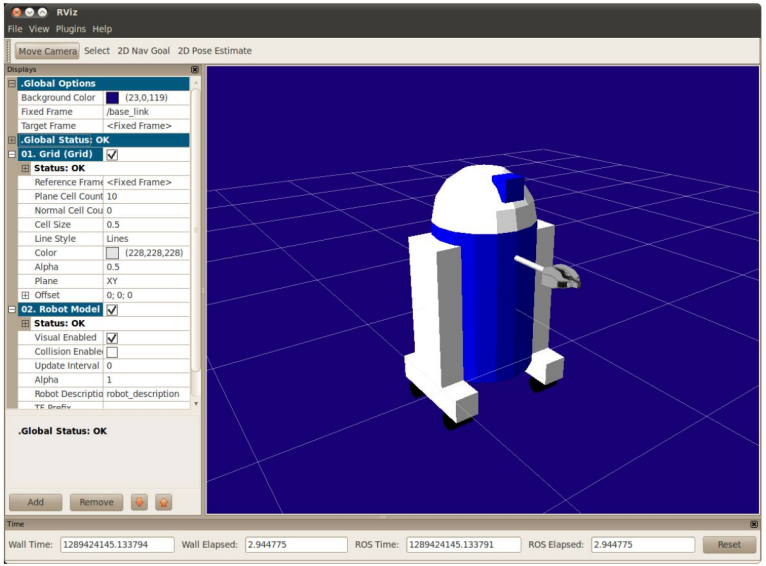
- Un actuador al que está conectada la transmisión. El actuador se especifica por su atributo de nombre y los siguientes subelementos:

  <reducciónmecánica> (opcional): Especifica una reducción mecánica en la transmisión de la junta/actuador. Es posible que esta etiqueta no sea necesaria para todas las transmisiones.

  <hardwareInterface> (opcional): Especifica una interfaz de hardware de espacio conjunto admitida.

Construyendo un modelo visual

- Seguiremos el tutorial de ROS para armar un R2D2
- En esta etapa nos concentramos exclusivamente en el aspecto visual
- Objetivo final:

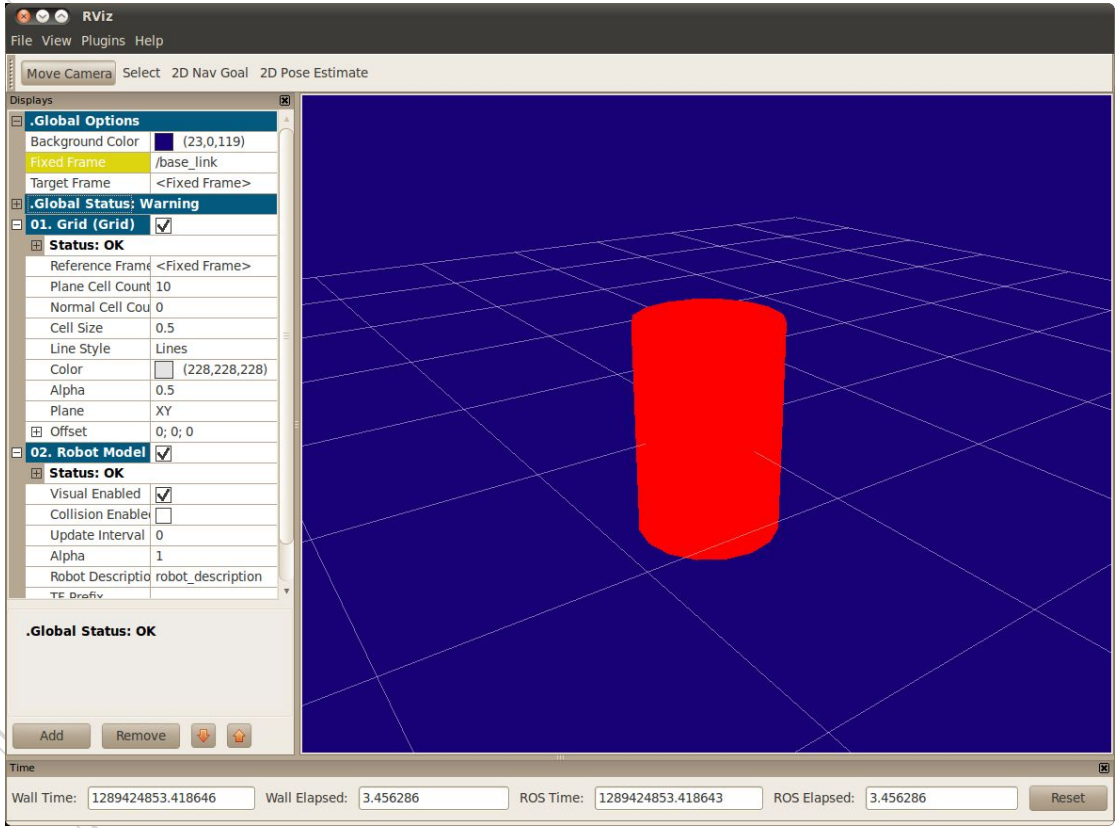


Paso 1: Construir figuras

```
<?xml version="1.0"?>
<robot name="myfirst">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>
</robot>
```

- En español esto es un robot con el nombre myfirst, que contiene solo un eslabón (part), cuyo componente visual es solo un cilindro de 0,6 metros de largo con un radio de 0,2 metros.
- Para examinar el modelo, inicie el archivo display.launch:
  - `roslaunch urdf_tutorial display.launch model:=urdf/01-myfirst.urdf`
- Esto hace tres cosas.
  - Carga el modelo especificado en el servidor de parámetros.
  - Ejecuta nodos para publicar sensor_msgs/JointState y transformaciones (tf)
  - Inicia Rviz con un archivo de configuración
- Tener en cuenta que la línea roslaunch anterior asume que se está ejecutando desde el directorio del paquete urdf_tutorial

Paso 1: Construir figuras

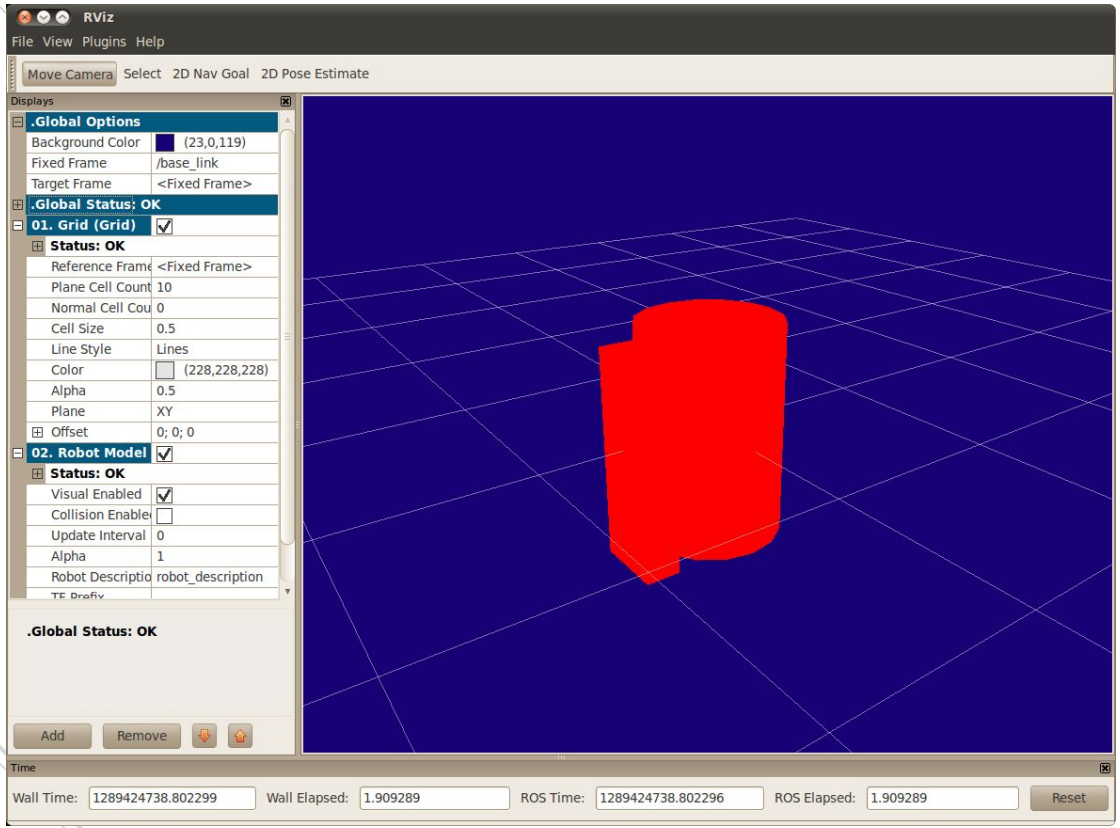


Paso 2: Varias figuras

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
  </joint>

</robot>
```

- Al agregar mas elementos, debemos también incluir sus juntas
- Definimos una caja de 0,6 mx 0,1 mx 0,2 m
- La articulación se define en términos de un parent y un child. URDF es, en última instancia, una estructura de árbol con un eslabón raíz. Esto significa que la posición de la pierna depende de la posición de base_link.
- `roslaunch urdf_tutorial display.launch model:=urdf/02-multipleshapes.urdf`

Paso 2: Varias figuras



Paso 3: Orígenes

```xml
<?xml version="1.0"?>
<robot name="origins">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0 -0.22 0.25"/>
  </joint>

</robot>
```

- `roslaunch urdf_tutorial display.launch model:=urdf/03-origins.urdf`

Paso 4: Colores

```
<?xml version="1.0"?>
<robot name="materials">

  <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>

  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>


  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>
```

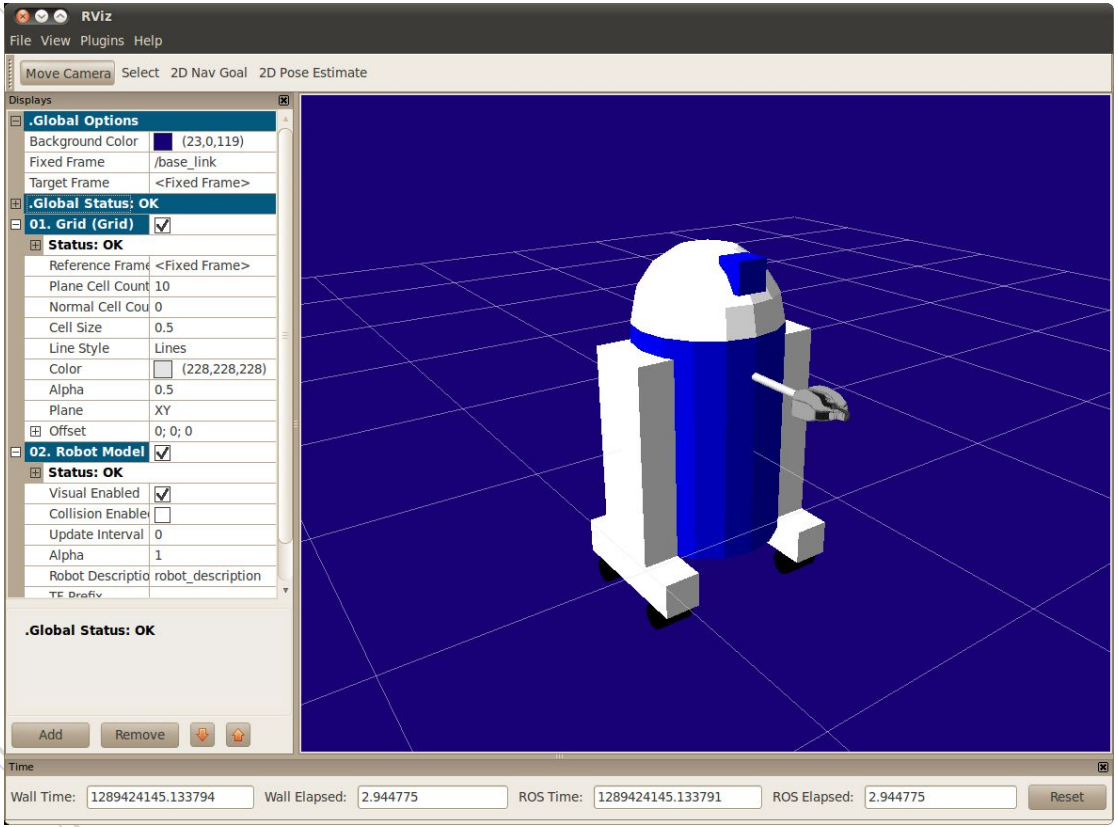
- `roslaunch urdf_tutorial display.launch model:=urdf/04-materials.urdf`

Paso 5: Modelo final

```xml
<?xml version="1.0"?>
<robot name="visual">

  <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>
  <material name="black">
    <color rgba="0 0 0 1"/>
  </material>
  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <link name="head">
    <visual>
      <geometry>
        <sphere radius="0.2"/>
      </geometry>
      <material name="white"/>
    </visual>
  </link>
  <joint name="head_swivel" type="fixed">
    <parent link="base_link"/>
    <child link="head"/>
    <origin xyz="0 0 0.3"/>
  </joint>

  <link name="box">
    <visual>
      <geometry>
        <box size="0.08 0.08 0.08"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <joint name="tobox" type="fixed">
    <parent link="head"/>
    <child link="box"/>
    <origin xyz="0.1814 0 0.1414"/>
  </joint>
</robot>
```

- `roslaunch urdf_tutorial display.launch model:=urdf/05-visual.urdf`

Paso 5: Modelo final



Ahora que se mueva. 1: Cabeza

```
<joint name="head_swivel" type="continuous">
  <parent link="base_link"/>
  <child link="head"/>
  <axis xyz="0 0 1"/>
  <origin xyz="0 0 0.3"/>
</joint>
```

La conexión entre el cuerpo y la cabeza es una unión continua, lo que significa que puede tomar cualquier valor de ángulo. Las ruedas también están modeladas así, para que puedan rodar en ambas direcciones.

La única información adicional que tenemos que agregar es el eje de rotación, aquí especificado por una tríada xyz, que especifica un vector alrededor del cual girará la cabeza. Como queremos que gire alrededor del eje z, especificamos el vector "0 0 1".

Ahora que se mueva. 2: Pinza

```
<joint name="left_gripper_joint" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" lower="0.0" upper="0.548" velocity="0.5"/>
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="gripper_pole"/>
  <child link="left_gripper"/>
</joint>
```

Las articulaciones de agarre derecha e izquierda se modelan como articulaciones giratorias. Esto significa que giran de la misma manera que lo hacen las juntas continuas, pero tienen límites estrictos. Por lo tanto, debemos incluir la etiqueta de límite que especifica los límites superior e inferior de la articulación (en radianes). También debemos especificar una velocidad y esfuerzo máximos para esta articulación.

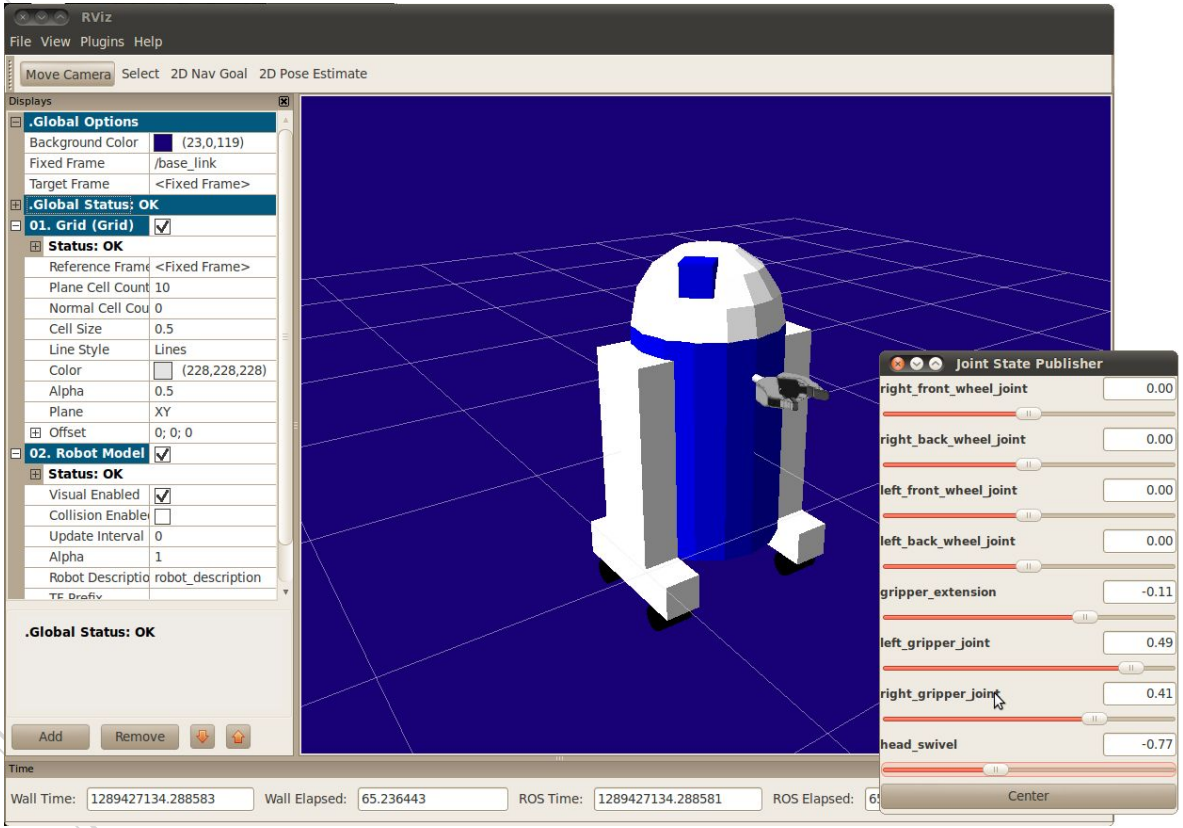
Ahora que se mueva. 3: Brazo

```
<joint name="gripper_extension" type="prismatic">
  <parent link="base_link"/>
  <child link="gripper_pole"/>
  <limit effort="1000.0" lower="-0.38" upper="0" velocity="0.5"/>
  <origin rpy="0 0 0" xyz="0.19 0 0.2"/>
</joint>
```

El brazo tiene una articulación prismática. Esto significa que se mueve a lo largo de un eje, no alrededor de él. Este movimiento de traslación es lo que permite que nuestro modelo de robot extienda y retraiga su brazo de agarre.

Los límites del brazo prismático se especifican de la misma manera que una articulación giratoria, excepto que las unidades son metros, no radianes.

Ahora que se mueva. 3: Brazo



Inclusión de propiedades: Colisión

Hasta ahora, solo hemos especificado nuestros eslabones con un solo subelemento, visual, que define (como era de esperar) cómo se ve el robot. Sin embargo, para que la detección de colisiones funcione o para simular el robot con un motor de física en algo como Gazebo, también debemos definir un elemento de colisión.

```
<link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue">
        <color rgba="0 0 .8 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </collision>
  </link>
```

Inclusión de propiedades: Colisión

- El elemento de colisión es un subelemento directo del objeto de link, al mismo nivel que la etiqueta visual.
- El elemento de colisión define su forma de la misma manera que lo hace el elemento visual, con una etiqueta de geometría. El formato de la etiqueta de geometría es exactamente el mismo aquí que con el objeto visual.
- También puede especificar un origen de la misma manera que un subelemento de la etiqueta de colisión (como con el objeto visual).

En muchos casos, se deseará que la geometría y el origen de la colisión sean exactamente iguales a la geometría y el origen visuales. Sin embargo, hay dos casos principales en los que no.

Procesamiento más rápido: hacer la detección de colisiones para dos mallas es mucho más complejo a nivel computacional que para dos geometrías simples. Por lo tanto, es posible que desee reemplazar las mallas con geometrías más simples en el elemento de colisión.

Zonas seguras: es posible que desee restringir el movimiento cerca de equipos sensibles. Por ejemplo, si no quisiéramos que nada chocara con la cabeza de R2D2, podríamos definir la geometría de colisión como un cilindro que encierra su cabeza para evitar que algo se acerque demasiado a su cabeza.

Inclusión de propiedades Físicas: Inercia

```
<link name="base_link">
  <visual>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
    <material name="blue">
      <color rgba="0 0 .8 1"/>
    </material>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" ixz="0.0" iyy="0.4" iyz="0.0" izz="0.2"/>
  </inertial>
</link>
```

Inclusión de propiedades Físicas: Inercia

Este elemento también es un subelemento del objeto de enlace.

La masa se define en kilogramos.

La matriz de inercia rotacional de 3x3 se especifica con el elemento de inercia. Como es simétrica, puede representarse por solo 6 elementos, como tal.

ixx	**ixy**	**ixz**
ixy	**iyy**	**iyz**
ixz	iyz	**izz**

Inclusión de propiedades Físicas: Inercia

Esta información la pueden proporcionar programas de modelado como MeshLab. La inercia de las primitivas geométricas (cilindro, caja, esfera) se puede calcular usando la lista de tensores de momento de inercia de cualquier libro.

El tensor de inercia depende tanto de la masa como de la distribución de masa del objeto. Una buena primera aproximación es asumir una distribución uniforme de la masa en el volumen del objeto y calcular el tensor de inercia en función de la forma del objeto, como se describe anteriormente.

Si no estamos seguros de qué poner, una matriz con ixx/iyy/izz=1e-3 o menor suele ser un valor predeterminado razonable para un eslabón de tamaño medio (corresponde a una caja de 0,1 m de longitud lateral con una masa de 0,6 kg). La matriz identidad es una elección particularmente mala, ya que a menudo es demasiado alta (corresponde a una caja de 0,1 m de lado con una masa de 600 kg).

También puede especificar una etiqueta de origen para especificar el centro de gravedad y el marco de referencia inercial (en relación con el marco de referencia del eslabón).

Cuando se utilizan controladores en tiempo real, los elementos de inercia de cero (o casi cero) pueden hacer que el modelo de robot colapse sin previo aviso, y todos los eslabones aparecerán con sus orígenes coincidiendo con el origen global.

Inclusión de propiedades Físicas: Coeficientes de Fricción y Amortiguamiento

También se puede definir cómo se comportan los eslabones cuando están en contacto entre sí. Esto se hace con un subelemento de la etiqueta de colisión llamado contact_coefficients. Hay tres atributos para especificar:

mu - Coeficiente de fricción

kp - Coeficiente de rigidez

kd - Coeficiente de amortiguamiento

La forma en que se mueve la articulación se define mediante la **etiqueta dinámica** de la articulación. Hay dos atributos:

fricción - La fricción estática física. Para juntas prismáticas, las unidades son Newtons. Para juntas giratorias, las unidades son Newton metros.

amortiguamiento - El valor de amortiguamiento físico. Para juntas prismáticas, las unidades son Newton segundos por metro. Para juntas giratorias, Newton metro segundos por radian.

Si no se especifica, estos coeficientes por defecto son cero.

Uso del URDF en Gazebo

Podemos generar el modelo que ya creamos en Gazebo usando gazebo.launch:

```
roslaunch urdf_sim_tutorial gazebo.launch
```

Este archivo de tipo launch carga:

- el URDF del tutorial
- un mundo Gazebo vacío
- ejecuta el script para leer el urdf del parámetro y generarlo en Gazebo.
- De forma predeterminada, la interfaz gráfica de usuario de gazebo también se mostrará

Sin embargo, no hace nada y le falta mucha información clave que ROS necesitará para usar este robot. Previamente habíamos estado usando joint_state_publisher para especificar la pose de cada articulación. Sin embargo, el propio robot debería proporcionar esa información en el mundo real o en Gazebo. Sin especificar eso, Gazebo no sabe publicar esa información. Para que el robot sea interactivo (contigo y ROS), necesitamos especificar dos cosas: complementos (plugins) y transmisiones.

Gazebo plugin

- Para que ROS interactúe con Gazebo, tenemos que vincularnos dinámicamente a la biblioteca de ROS que le dirá a Gazebo qué hacer. En teoría, esto permite que otros sistemas operativos de robots interactúen con Gazebo de forma genérica.
- Para vincular Gazebo y ROS, especificamos el complemento en el URDF, justo antes de la etiqueta de cierre </robot>
- `roslaunch urdf_sim_tutorial gazebo.launch model:=urdf/09-publishjoints.urdf.xacro`

```
1  <gazebo>
2    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
3      <robotNamespace>/</robotNamespace>
4    </plugin>
5  </gazebo>
```

Gazebo generar controladores

- Ahora que hemos vinculado ROS y Gazebo, necesitamos especificar algunos fragmentos de código ROS que queremos ejecutar dentro de Gazebo, que genéricamente llamamos controladores. Estos se cargan inicialmente en el espacio de parámetros de ROS. Tenemos un archivo .yaml, **joints.yaml** que especifica nuestro primer controlador.

```
type: "joint_state_controller/JointStateController"
publish_rate: 50
```

Este controlador se encuentra en el paquete joint_state_controller y publica el estado de las articulaciones del robot en ROS directamente desde Gazebo. En 09-joints.launch se puede ver cómo cargar este archivo yaml en el espacio de nombres r2d2_joint_state_controller. Luego llamamos al script [[controller_manager]]/spawner con ese espacio de nombres que lo carga en Gazebo.

Podemos iniciarlo, pero todavía no está terminado.

```
roslaunch urdf_sim_tutorial 09-joints.launch
```

Esto ejecutará el controlador y, de hecho, publicará en el tema /joint_states pero sin nada en ellos.

Gazebo: Transmisiones

- Para cada junta no fija, necesitamos especificar una transmisión, que le dice a Gazebo qué hacer con la junta. Comenzamos con la articulación de la cabeza.

```
<transmission name="head_swivel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="$head_swivel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="head_swivel">
    <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>
</transmission>
```

```
roslaunch urdf_sim_tutorial 09-joints.launch model:=urdf/10-firsttransmission.urdf.xacro
```

Gazebo: Control de juntas

- Agregamos un nuevo controlador

```
type: "position_controllers/JointPositionController"
joint: head_swivel
```

Esto especifica el uso de un JointPositionController del paquete position_controllers para controlar la transmisión head_swivel. Tener en cuenta que la interfaz de hardware en URDF para esta articulación coincide con el tipo de controlador.

Ahora podemos lanzar esto con la configuración agregada como antes:

```
roslaunch urdf_sim_tutorial 10-head.launch
```

Ahora Gazebo está suscrito a un nuevo tópico y luego puede controlar la posición de la cabeza publicando un valor en ROS.

```
rostopic pub /r2d2_head_controller/command std_msgs/Float64 "data: -0.707"
```

Cuando se publica este comando, la posición cambiará inmediatamente al valor especificado. Esto se debe a que no especificamos ningún límite para la articulación en nuestro urdf.

```
roslaunch urdf_sim_tutorial 10-head.launch model:=urdf/11-limittransmission.urdf.xacro
```

Gazebo: Control de la pinza

Podemos cambiar el URDF para las juntas del Gripper de forma similar. Sin embargo, en lugar de controlar individualmente cada articulación de la pinza con su propio tema ROS, podríamos querer agruparlos. Para esto, solo necesitamos especificar un controlador diferente.

```
type: "position_controllers/JointGroupPositionController"
joints:
  - gripper_extension
  - left_gripper_joint
  - right_gripper_joint
```

```
roslaunch urdf_sim_tutorial 12-gripper.launch
```

Con esto, podemos especificar la posición con una matriz de tipo float. Abierto y fuera o cerrado y retraído.

Gazebo: Control de las ruedas

Para conducir el robot, especificamos otra transmisión para cada una de las ruedas desde dentro del macro de ruedas.

Esto es como las otras transmisiones, excepto

- Utiliza parámetros macro para especificar nombres.
- Utiliza una VelocityJointInterface.

Dado que las ruedas en realidad tocarán el suelo y, por lo tanto, interactuarán físicamente con él, también especificamos información adicional sobre el material de las ruedas.

```
roslaunch urdf_sim_tutorial 13-diffdrive.launch
```

FIN!