Trabajo final de máster

Extendiendo las capacidades del robot RESCUER en ROS

Planificación, navegación y simulación

Julio Jesús León Pérez

24/11/2015

Universitat Jaume I
Máster Universitario en Sistemas Inteligentes

Tutor
Prof. Enric Cervera Mateu

Fecha de la defensa
24.11.2015

“La simplicidad es el logro final. Después de que uno haya jugado con una cantidad grande de notas, es la simplicidad que emerge como una recompensa del arte.”Frédéric Chopin (1810-1849)

Índice general

1. Introducción

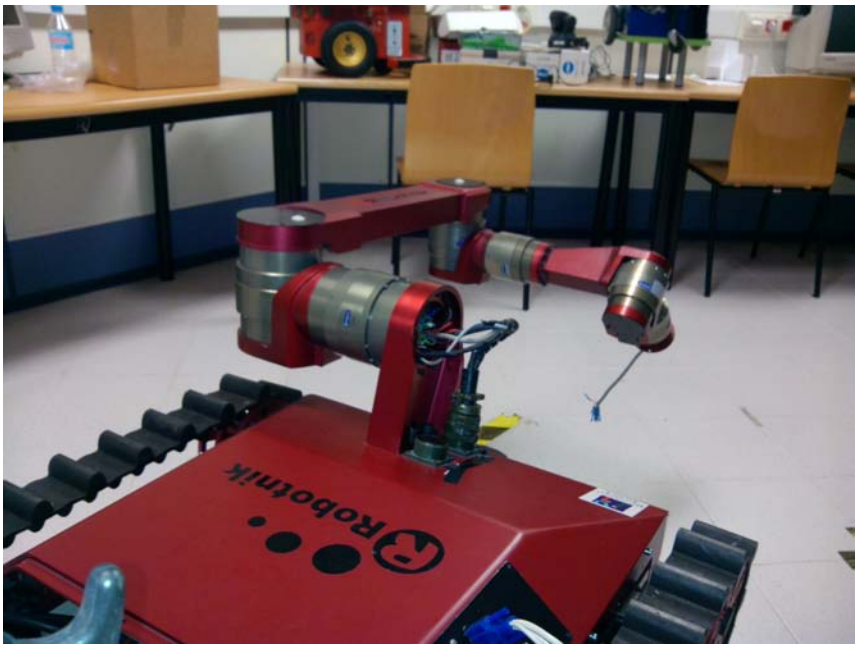


Figura 1.1.: RESCUER

RESCUER es un modelo de robot fabricado por la empresa *Robotnik*. La Universitat Jaume I cuenta con un modelo de este tipo.

RESCUER está formado por una base móvil y un brazo articulado (ver Figura 1.1). La base está traccionada por cadenas, lo que habilita al robot para moverse por terrenos irregulares e incluso subir y bajar escaleras. El robot no tiene montado ningún manipulador en el momento de comenzar el proyecto, por lo que no puede realizar tareas de agarre. Tampoco tiene ningún sensor visual ni de rango.

Al inicio del presente proyecto, *RESCUER* está funcionando utilizando Robotic Operating System (en adelante, ROS). Se puede controlar la base móvil para desplazar el robot y se pueden realizar movimientos de lo motores que forman el brazo.

Tal y como se refleja en "Adaptación del robot *RESCUER* a *ROS*" [Leó15], interesa hacer operativo el robot bajo la plataforma MoveIt!, conjunto de herramientas que funcionan con *ROS*, que permite realizar tareas de planificación automática. También interesa construir una versión simulada del robot para poder realizar estudios sin necesidad de tener físicamente el robot. Por último, ya que se dispone de una

base móvil, también resulta de interés integrar el robot con la pila de navegación de *ROS*, lo que permitiría realizar la planificación de movimientos de la base.

Los objetivos fundamentales del presente trabajo son los siguientes:

- **Realizar tareas de planificación automática para mover el brazo robótico y la mano (no instalada).**
- **Integrar el robot con la pila de navegación de *ROS*, para realizar tareas de planificación de movimiento de la base de forma autónoma.**
- **Crear una versión simulada del robot e integrarlo con la planificación automática y la navegación autónoma.**

El trabajo realizado se refleja en la presente memoria que se compone de los capítulos siguientes:

Capítulo 1, esta introducción.

En el Capítulo 2, se ve la planificación y la metodología de trabajo.

En el siguiente capítulo (Capítulo 3), se explica la arquitectura, basada en *ROS* y el simulador, que se utiliza.

En el Capítulo 4, se construye la descripción del robot, necesaria para la práctica totalidad de componentes del sistema.

En el Capítulo 5, se ven los controladores implementados y configurados para trabajar con el hardware del robot.

En el capítulo siguiente, Capítulo 6, se crea una interfaz, en forma de paquete catkin, para trabajar de manera más sencilla con los controladores y la descripción del robot.

En el Capítulo 7, se crea la configuración de *MoveIt!*.

En el Capítulo 8, se crea la configuración adecuada para el simulador.

En el Capítulo 9, se explican las tareas realizadas que tienen que ver con la navegación.

En el Capítulo 10, se detallan algunos problemas encontrados durante la realización del proyecto.

En el Capítulo 11, se crea una guía rápida para la utiliza*ción de RESCUER*.

En el Capítulo 12, aparecen las conclusiones extraídas tras la ejecución del proyecto.

Finalmente, en el Capítulo 13, se listan las propuestas de trabajo futuro que se han ido viendo a lo largo del proyecto.

2. Planificación

2.1. Objetivos

Teniendo las capacidades básicas de mover la base y el brazo (ver [Leó15]), se pretende integrar RESCUER con MoveIt!, la pila de navegación y crear un robot simulado.

Los objetivos detallados de este proyecto se enumeran a continuación:

1. Integración con MoveIt!
2. Integración con la pila de navegación de ROS.
3. Creación del robot simulado.
4. Pruebas.
5. Generación de documentación.

2.2. Metodología

Investigación sobre los componentes necesarios para cumplir con los objetivos. Creación de los componentes necesarios. Realizar pruebas de planificación automática con el brazo. Realizar pruebas de navegación. Pruebas con el simulador. Finalmente se pretende crear la documentación necesaria para la reutilización de este trabajo en otros proyectos.

2.3. Planificación

Tarea	Horas planificadas	Objetivo
Documentación sobre MoveIt!	30	Integración con MoveIt!
Creación de paquetes para MoveIt!	70	Integración con MoveIt!
Documentación sobre la pila de navegación	30	Pila de navegación
Creación de paquetes para pila de navegación	50	Pila de navegación
Documentación sobre simulación	30	Simulación
Creación del robot simulado	50	Simulación
Pruebas	20	Pruebas
Documentación	20	Documentación

3. Estudio e implantación de la arquitectura

3.1. Arquitectura necesaria

En la Figura 3.1 se puede ver el nodo *move_group*, que forma parte de la arquitectura fundamental de *MoveIt!*. Para comenzar este proyecto se tiene en cuenta que se dispone de un controlador para mover la base del robot y otro para mover el brazo robótico. Los componentes necesarios son los que aparecen en los cuadros en azul en la Figura 3.1. Para utilizar *move_group* parece necesario tener:

- ☑ Controlador de la base.
- ☑ Controlador del brazo robótico.
- ☒ Un controlador capaz de seguir trayectorias para el brazo.
- ☒ Un sensor de puntos 3D.
- ☑ Publicar el estado de las articulaciones del robot. Esto se cumple mediante el tópico */joint_states* que muestra la información sobre las articulaciones del brazo.
- ☒ Publicar el estado del robot y las transformaciones entre las distintas articulaciones del robot.

El simulador que se va a utilizar es *Gazebo.* Para poder integrarlo con *RESCUER* es necesario crear los componentes que aparecen en la Figura 3.2 marcados en azul:

- ☒ Tener la descripción *URDF* completa del robot.
- ☒ Describir las transmisiones de las articulaciones en la descripción *URDF.*
- ☒ Definir los controladores a utilizar por el robot simulado, utilizando *gazebo_ros_control.*

En cuanto a la navegación, como se indica en la Figura 3.3 los componentes necesarios son:

- ☑ Controlador de la base móvil.
- ☒ Publicar el estado del robot y las transformaciones entre las distintas articulaciones del robot.
- ☒ Publicar la odometría del robot.

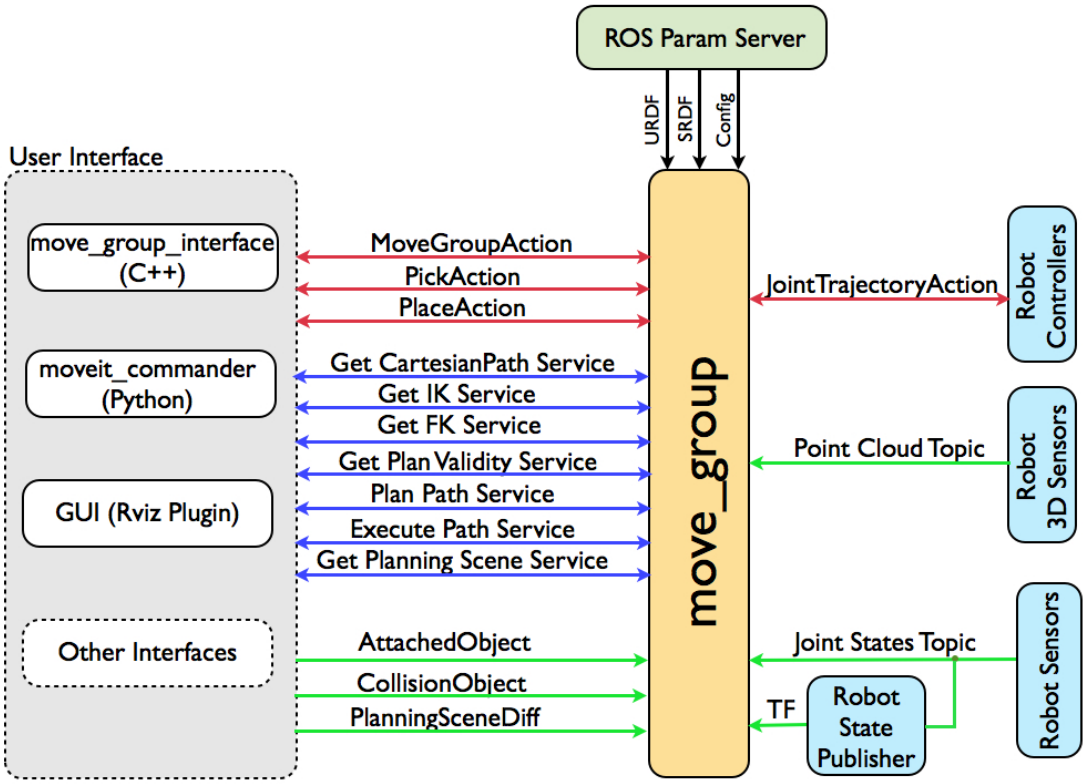


Figura 3.1.: El nodo move_group de MoveIt!

- ☒ Un sensor de rango o de nube de puntos.

En resumen, las necesidades para el proyecto y que el robot **no** cumple son:

- ☒ Tener la descripción *URDF* completa del robot.
- ☒ Publicar el estado del robot y las transformaciones entre las distintas articulaciones del robot.
- ☒ Describir las transmisiones de las articulaciones en el *URDF.*
- ☒ Un controlador capaz de seguir trayectorias para el brazo.
- ☒ Publicar la odometría del robot.
- ☒ Un sensor de puntos 3D.
- ☒ Un sensor de rango o de nube de puntos (similar al requisito anterior).
- ☒ Definir los controladores a utilizar por el robot simulado, utilizando *gazebo_ros_control.*

Otro requerimiento que se introduce en el proyecto es instalar físicamente una mano de Barret en el robot e integrarla dentro del sistema, planificación, visualización, etc.

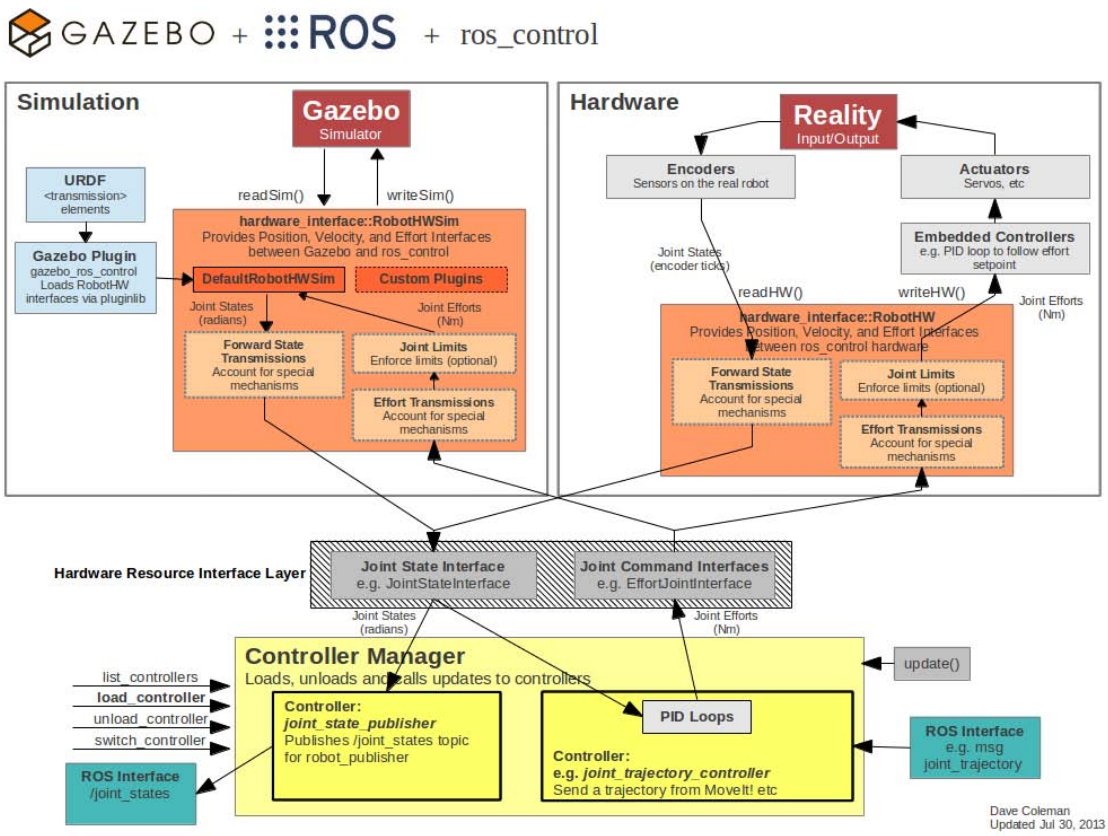


Figura 3.2.: Gazebo + ROS + ros_control

3.2. Repositorio

Para salvaguardar el software desarrollado se sigue utilizando el repositorio creado en bitbucket.org (ver [Leó15]).

La dirección del repositorio es https://bitbucket.org/xulioxesus/rescuer_catkin

Como puede verse en la Figura 3.4, la raíz del repositorio es el entorno de trabajo de catkin.

También puede observarse que se está versionando el código fuente y esta propia documentación.

En la Figura 3.5 se ve listan los distintos paquetes de ROS que forman el software de RESCUER.

Para descargar el repositorio y versionarlo se utiliza el siguiente comando:

```
git clone https://bitbucket.org/xulioxesus/rescuer_catkin.git
```

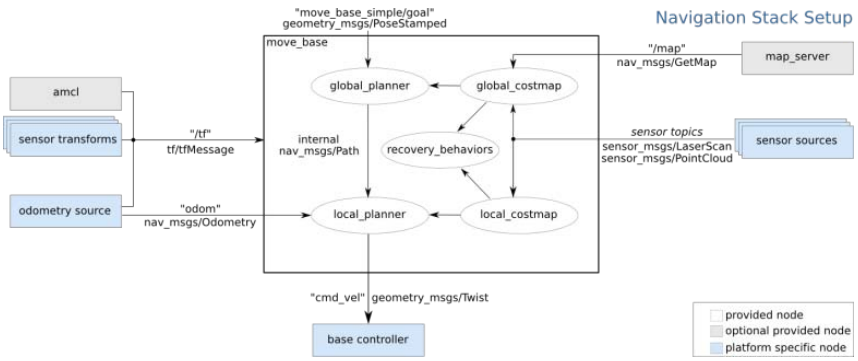


Figura 3.3.: Esquema de la pila de navegación

Es necesario tener permiso de acceso al repositorio y un usuario de *bitbucket,* se puede solicitar el acceso a xulioxesus@gmail.com.

Una vez versionado el espacio de trabajo catkin se puede construir de forma muy sencilla utilizando el comando *catkin_make* desde el directorio creado.

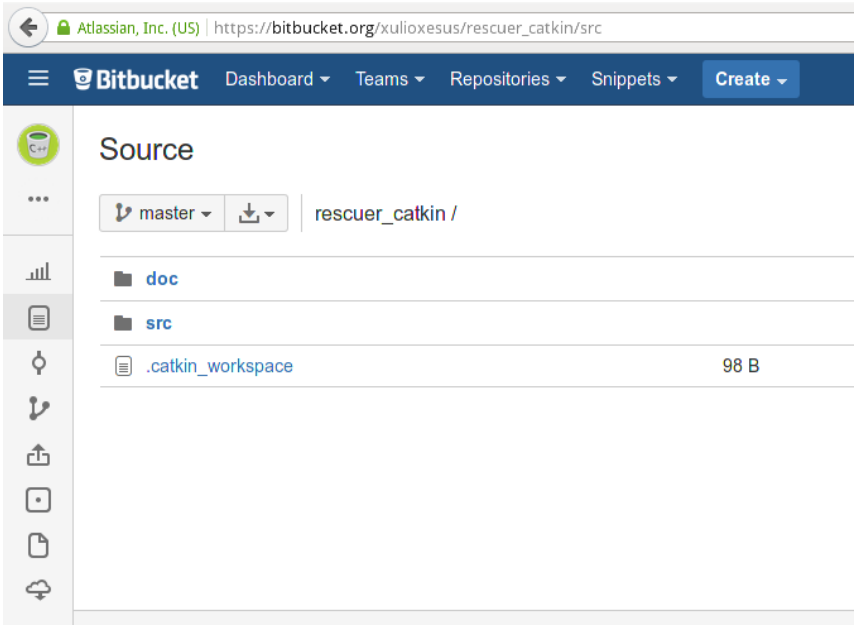


Figura 3.4.: Repositorio

3.3. Paquetes catkin

La descripción de los paquetes catkin que aparecen en el repositorio es la siguiente:

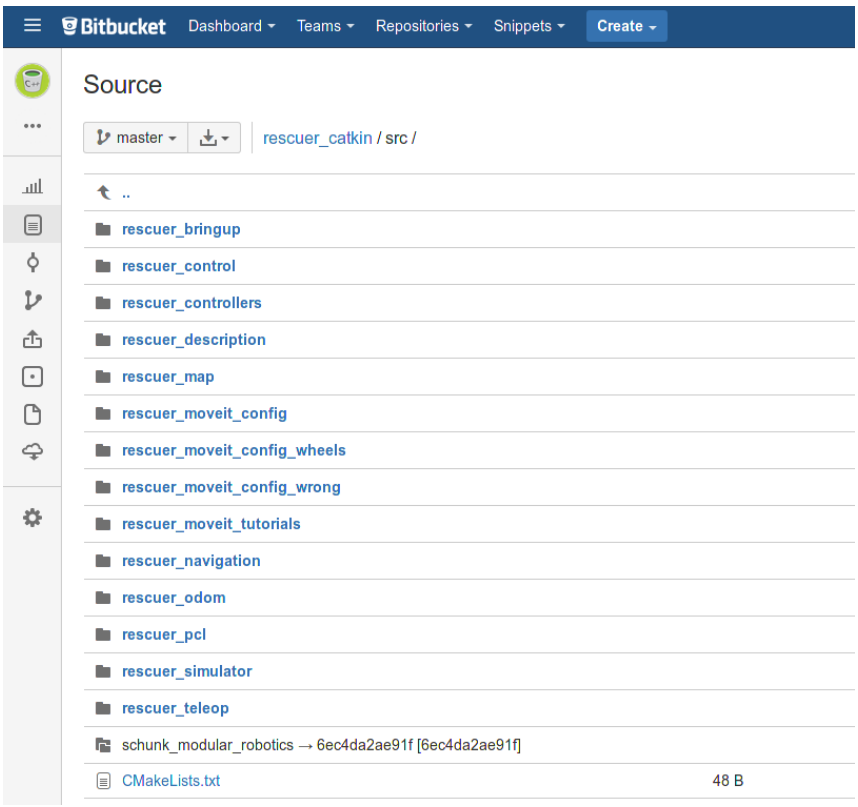


Figura 3.5.: Repositorio, detalle de src

rescuer_bringup

Creado como interfaz para cargar los controladores y la descripción del robot. Podría ser sustituido por ros_control.

rescuer_control

Necesario para definir los controladores que usa el simulador *Gazebo*.

rescuer_controllers

Controladores hardware.

rescuer_description

Descripción del robot mediante URDF.

rescuer_map

Para la creación de mapas necesarios para la navegación.

rescuer_moveit_config

Generado mediante MoveIt! Setup Assistant.

rescuer_moveit_config_wheels

Candidato a ser eliminado.

Generado mediante MoveIt! Setup Assistant para incluir las ruedas invisibles. La necesidad de añadir un par de ruedas invisibles se explica en el Capítulo 9.

rescuer_moveit_config_wrong

Se encuentra sin uso, está propuesto para ser eliminado.

Utilizado como copia de seguridad para evitar sobrescrituras de Moveit! Setup Assistant.

rescuer_moveit_tutorials

Utilizado para guardar experimentos relacionados con la planificación automática.

rescuer_navigation

Creado para la configuración de la navegación.

rescuer_pcl

Creado para hacer reconocimiento de objetos en 3D.

rescuer_teleop

Teleoperación del robot.

rescuer_odom

Se encuentra sin uso, está propuesto para ser eliminado.

rescuer_simulator

Configura y lanza *Gazebo.*

schunk_modular_robotics

Paquete versionado de la pila de schunk_modular_robotics, para arreglar el fallo del módulo cinco del brazo, ver [Leó15].

3.4. Hardware

Se procede a realizar una serie de trabajos respecto al hardware para cumplir con las necesidades descritas en el primer apartado del presente capítulo .

Kinect

Se instala físicamente una Kinect en la parte frontal del robot, como puede verse en la Figura 3.6.

Barret Hand

Se monta una mano de Barret en el robot, realizando las siguientes tareas:

- Revisión del cableado que circula por el interior del brazo hacia la mano robótica.
- Se monta monta el conector adecuado para la mano (ver Figura 3.7).
- Creación de una pieza de acople entre la mano y el brazo (ver Figura 3.7).

Tras intentar infructuosamente controlar la mano desde ROS, se descubre que la mano está estropeada tras probarla en otro robot del laboratorio.

Marcas cero en el brazo

En el brazo robótico se marcan las posiciones cero de los módulos (ver Figura 3.8) y se reconfigura esa posición en la memoría de los módulos mediante el programa PowerCube explicado en los anexos de [Leó15].

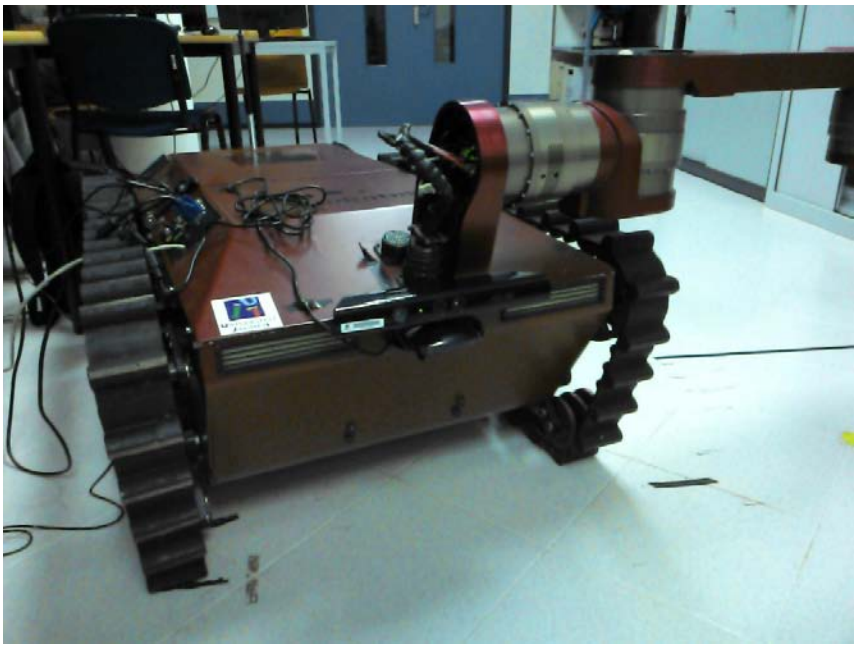


Figura 3.6.: Instalación de Kinect

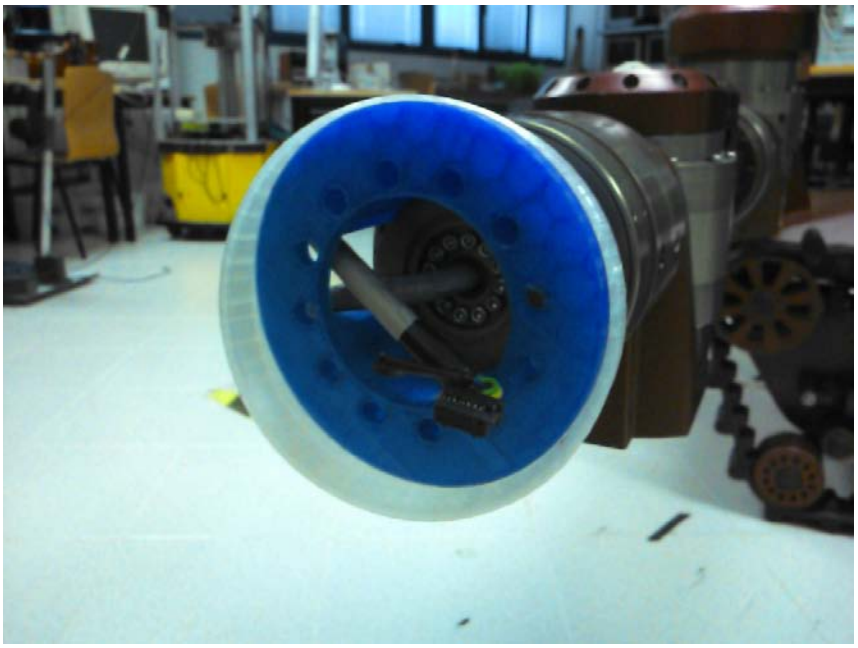


Figura 3.7.: Conector para la Barret Hand y pieza de acople entre brazo/mano

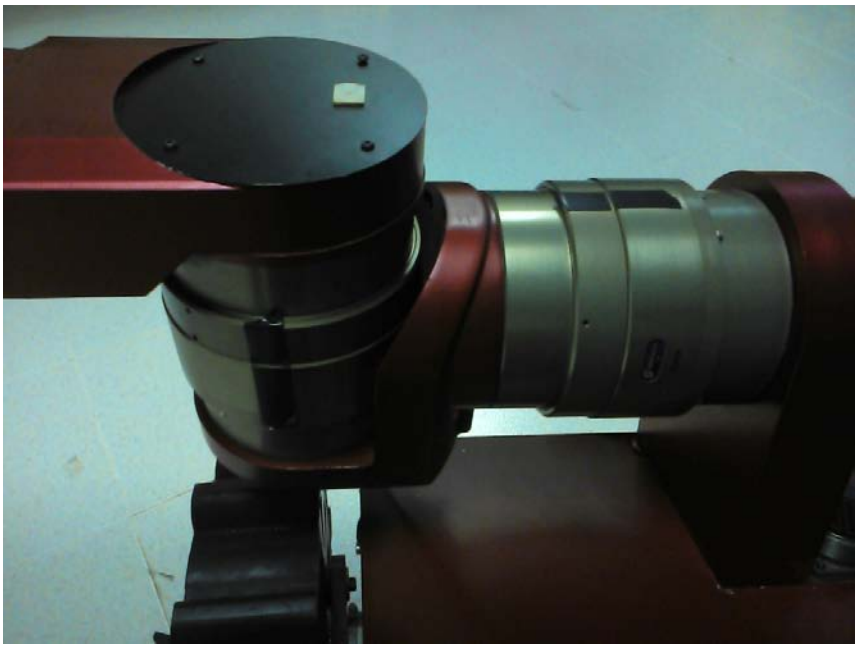


Figura 3.8.: Marcas rústicas de la posición cero de los módulos

4. Descripción del robot

El paquete *rescuer_description* (ver Apéndice C).

Para poder utilizar *MoveIt!* es necesario disponer de una descripción del robot en Unified Robot Description Format (en adelante URDF).

En [Leó15] ya se había creado el paquete de catkin *rescuer_description,* que contenía una descripción del brazo solamente, para poder hacer funcionar el controlador del brazo robótico. Ahora se hace necesario disponer de la descripción completa del robot. En este capítulo se documenta la construcción de esa descripción, que básicamente explica el contenido del paquete *rescuer_description.*

En la descripción del robot se incluye:

- La base móvil con las cadenas, llamado en conjunto *rover.*
- El brazo robótico, llamado también *modular.*
- Una mano de Barret.
- Una Kinect.

4.1. Modelos 3D

En el directorio *meshes* se encuentran los modelos de mallas de los componentes de *RESCUER,* estos modelos están en formato *STL* o *DAE.*

El contenido de *meshes* se resume a continuación, también se puede ver en la Figura 4.1:

- *barret*: modelos 3D de la mano de Barret.
- *kinect*: modelo 3D de Kinect.
- *modular*: modelos 3D del brazo robótico.
- *rover*: modelos 3D de la base móvil.
- *unused_sdh:* modelos 3D de un manipulador de Schunk (a eliminar).

Para visualizar los modelos se puede utilizar el programa *meshlab.*

Por ejemplo:

```
meshlab modular6.stl
```

El resultado puede observarse en la Figura 4.2.

4.2. Descripción URDF

El directorio urdf se utiliza para guardar la descripción del robot en formato *URDF*. Este formato se utiliza para describir un robot y cada una de sus partes. Para una introducción de cómo ROS utiliza URDF, ver [ROS14a] y para los tutoriales sobre URDF, ver [ROS14b].

Para facilitar el trabajo con la descripción en URDF se utiliza Xacro, que es un lenguaje de macros para XML usado en ROS. Para generar la descripción de *RESCUER* se utiliza el siguiente comando:

```
rosrun xacro xacro.py rescuer.urdf.xacro > rescuer.urdf
```

En la Figura 4.3 se observa la descripción del robot de forma gráfica. Las articulaciones (joints) y los elementos físicos del robot (links). La descripción completa en formato urdf puede consultarse en la Sección C.1.

De la misma forma, para generar la descripción del robot adecuada para el simulador:

```
rosrun xacro xacro.py rescuer.urdf.simulator.xacro >
   rescuer_simulator.urdf
```

Se puede ver en forma de árbol la descripción del robot en la la Figura 4.4. La descripción completa en formato urdf puede consultarse en la Sección C.2.

4.3. Carga de la descripción y visualización

En el directorio *launch* se encuentran los ficheros para cargar en ROS la configuración de la descripción del robot y para visualizar la descripción del mismo.

El contenido de *launch* se resume a continuación, también se puede ver en la Figura 4.5.

Ficheros marcados como unused

Se prevé borrarlos.

display.launch

Usado para visualizar la descripción del robot.

Carga la descripción del robot en ROS, lanza los nodos *joint_state_publisher*, *robot_state_publisher* y *rviz*.

```
<launch>
    <arg name="gui" default="False" />
    <param name="use_gui" value="$(arg gui)" />
    <param name="robot_description" command="$(find xacro)/xacro.py $(
        find rescuer_description)/urdf/rescuer.urdf.xacro" />

    <node name="rescuer_joint_state_publisher" pkg="
        joint_state_publisher" type="joint_state_publisher" />
    <node name="rescuer_robot_state_publisher" pkg="
        robot_state_publisher" type="robot_state_publisher" />

    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
        urdf_tutorial)/urdf.rviz" />
</launch>
```

Para ejecutarlo:

```
roslaunch rescuer_description display.launch
```

El resultado de ejecutar el comando puede verse en la Figura 4.6.

El grafo con los nodos de ROS tras lanzar este fichero puede verse en la Figura 4.7.

display_simulator.launch

Usado para visualizar la descripción del robot para el simulador. Es similar a *display.launch* pero la descripción cargada del robot es ligeramente diferente (por las ruedas invisibles).

Carga la descripción del robot en ROS, lanza los nodos *joint_state_publisher*, *robot_state_publisher* y *rviz.*

```
<launch>
    <arg name="gui" default="False" />
    <param name="use_gui" value="$(arg gui)" />
    <param name="robot_description" command="$(find xacro)/xacro.py $(
        find rescuer_description)/urdf/rescuer.urdf.simulator.xacro" />

    <node name="rescuer_joint_state_publisher" pkg="
        joint_state_publisher" type="joint_state_publisher" />
    <node name="rescuer_robot_state_publisher" pkg="
        robot_state_publisher" type="robot_state_publisher" />

    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
        urdf_tutorial)/urdf.rviz" />
</launch>
```

Para ejecutarlo:

```
roslaunch rescuer_description display_simulator.launch
```

El resultado de ejecutar el comando puede verse en la Figura 4.8.

El grafo con los nodos de ROS tras lanzar este fichero puede verse en la Figura 4.9.

load_description.launch

Solamente carga la descripción del robot.

```
<launch>
        <param name="robot_description" command="$(find xacro)/xacro.py
             $(find rescuer_description)/urdf/rescuer.urdf.xacro" />
</launch>
```

Para ejecutarlo:

```
roslaunch rescuer_description load_description.launch
```

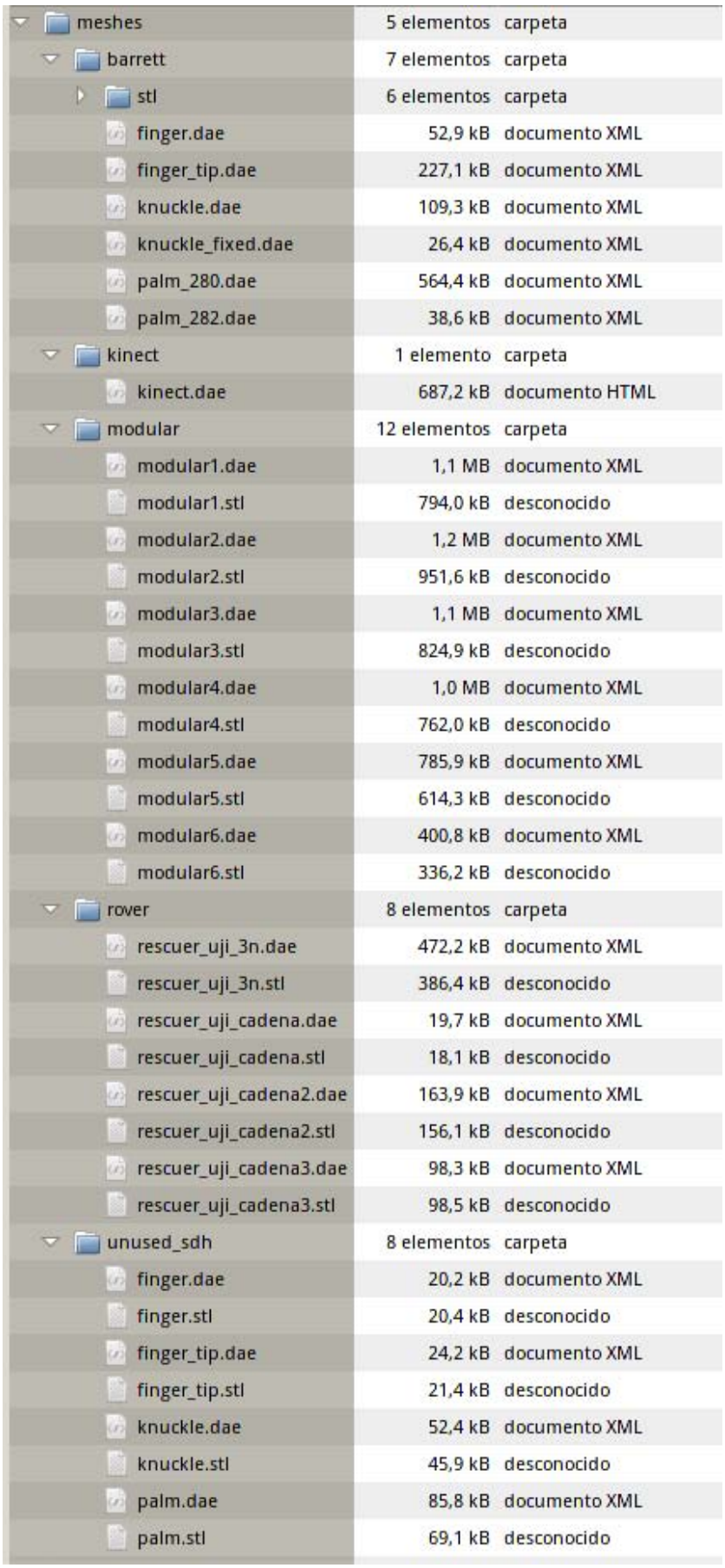
load_description_simulator.launch

Solamente carga la descripción del robot con la descripción ligeramente modificada.r

```
<launch>
        <param name="robot_description" command="$(find␣xacro)/xacro.py
            ␣$(find␣rescuer_description)/urdf/rescuer.urdf.xacro" />
</launch>
```

Para ejecutarlo:

```
roslaunch rescuer_description load_description_simulator.
   launch
```



meshes	5 elementos	carpeta
barrett	7 elementos	carpeta
stl	6 elementos	carpeta
finger.dae	52,9 kB	documento XML
finger_tip.dae	227,1 kB	documento XML
knuckle.dae	109,3 kB	documento XML
knuckle_fixed.dae	26,4 kB	documento XML
palm_280.dae	564,4 kB	documento XML
palm_282.dae	38,6 kB	documento XML
kinect	1 elemento	carpeta
kinect.dae	687,2 kB	documento HTML
modular	12 elementos	carpeta
modular1.dae	1,1 MB	documento XML
modular1.stl	794,0 kB	desconocido
modular2.dae	1,2 MB	documento XML
modular2.stl	951,6 kB	desconocido
modular3.dae	1,1 MB	documento XML
modular3.stl	824,9 kB	desconocido
modular4.dae	1,0 MB	documento XML
modular4.stl	762,0 kB	desconocido
modular5.dae	785,9 kB	documento XML
modular5.stl	614,3 kB	desconocido
modular6.dae	400,8 kB	documento XML
modular6.stl	336,2 kB	desconocido
rover	8 elementos	carpeta
rescuer_uji_3n.dae	472,2 kB	documento XML
rescuer_uji_3n.stl	386,4 kB	desconocido
rescuer_uji_cadena.dae	19,7 kB	documento XML
rescuer_uji_cadena.stl	18,1 kB	desconocido
rescuer_uji_cadena2.dae	163,9 kB	documento XML
rescuer_uji_cadena2.stl	156,1 kB	desconocido
rescuer_uji_cadena3.dae	98,3 kB	documento XML
rescuer_uji_cadena3.stl	98,5 kB	desconocido
unused_sdh	8 elementos	carpeta
finger.dae	20,2 kB	documento XML
finger.stl	20,4 kB	desconocido
finger_tip.dae	24,2 kB	documento XML
finger_tip.stl	21,4 kB	desconocido
knuckle.dae	52,4 kB	documento XML
knuckle.stl	45,9 kB	desconocido
palm.dae	85,8 kB	documento XML
palm.stl	69,1 kB	desconocido

Figura 4.1.: Contenido de rescuer_description/meshes

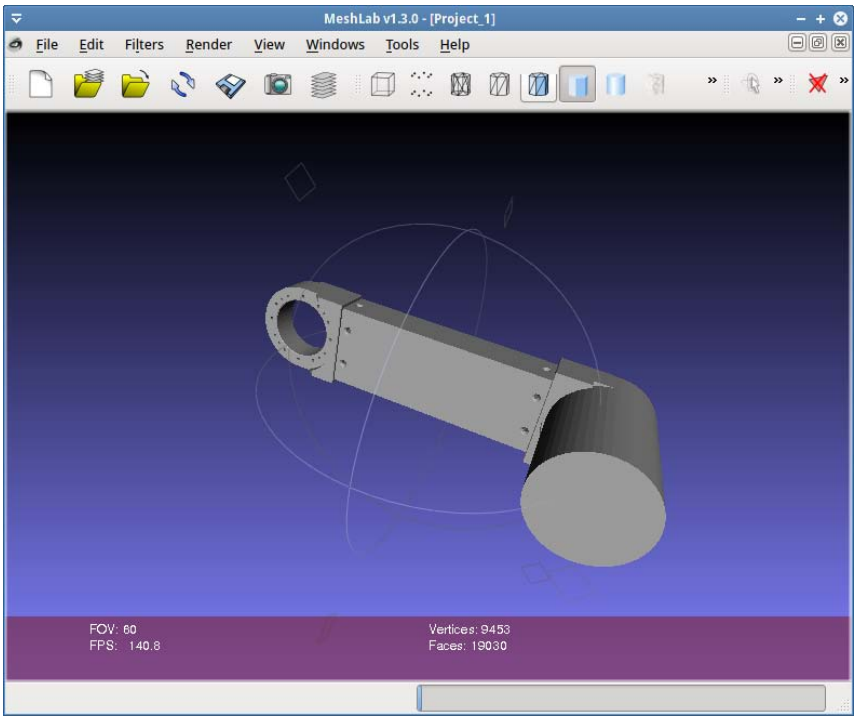


Figura 4.2.: Visualizar con meshlab

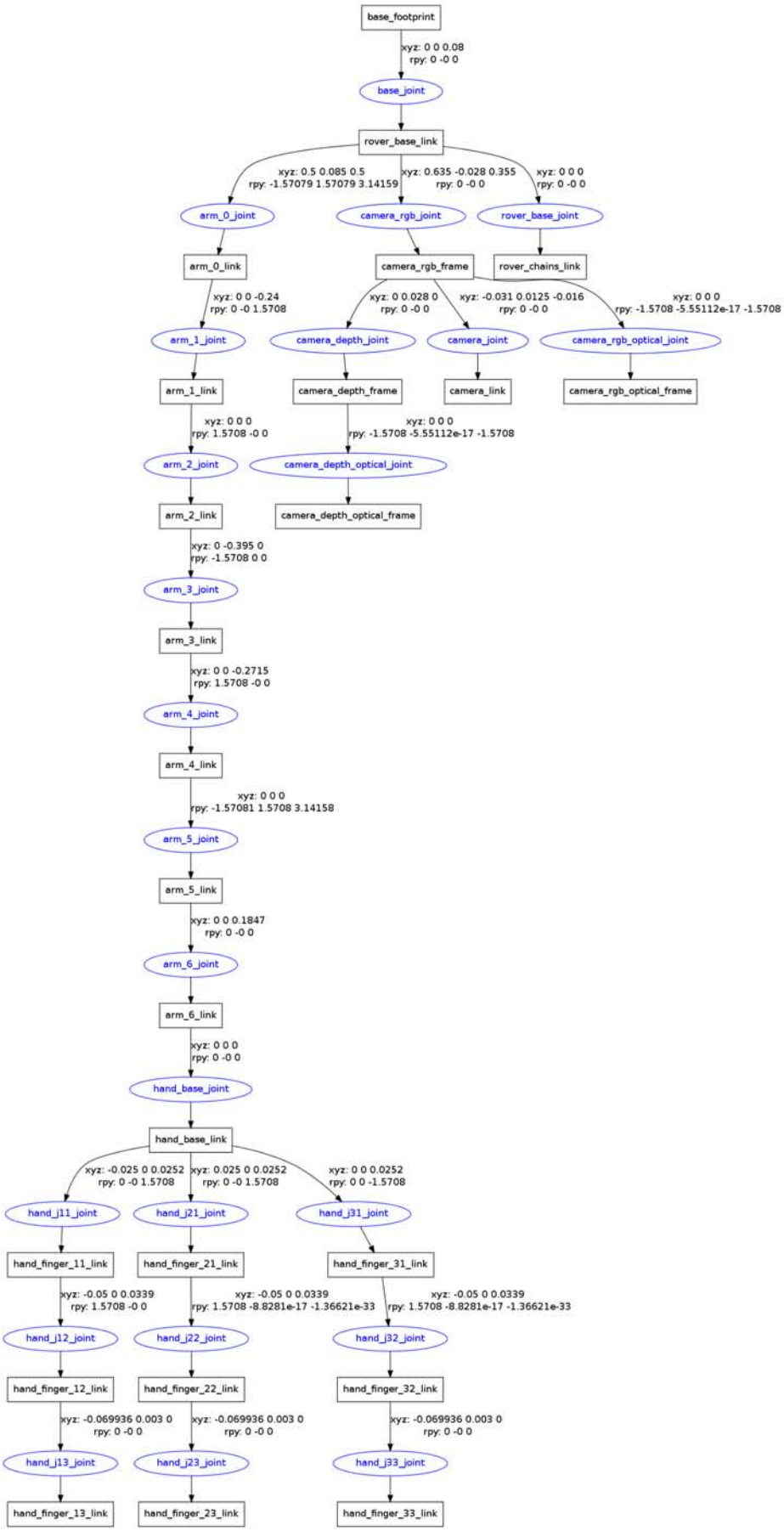


Figura 4.3.: rescuer.urdf en forma gráfica

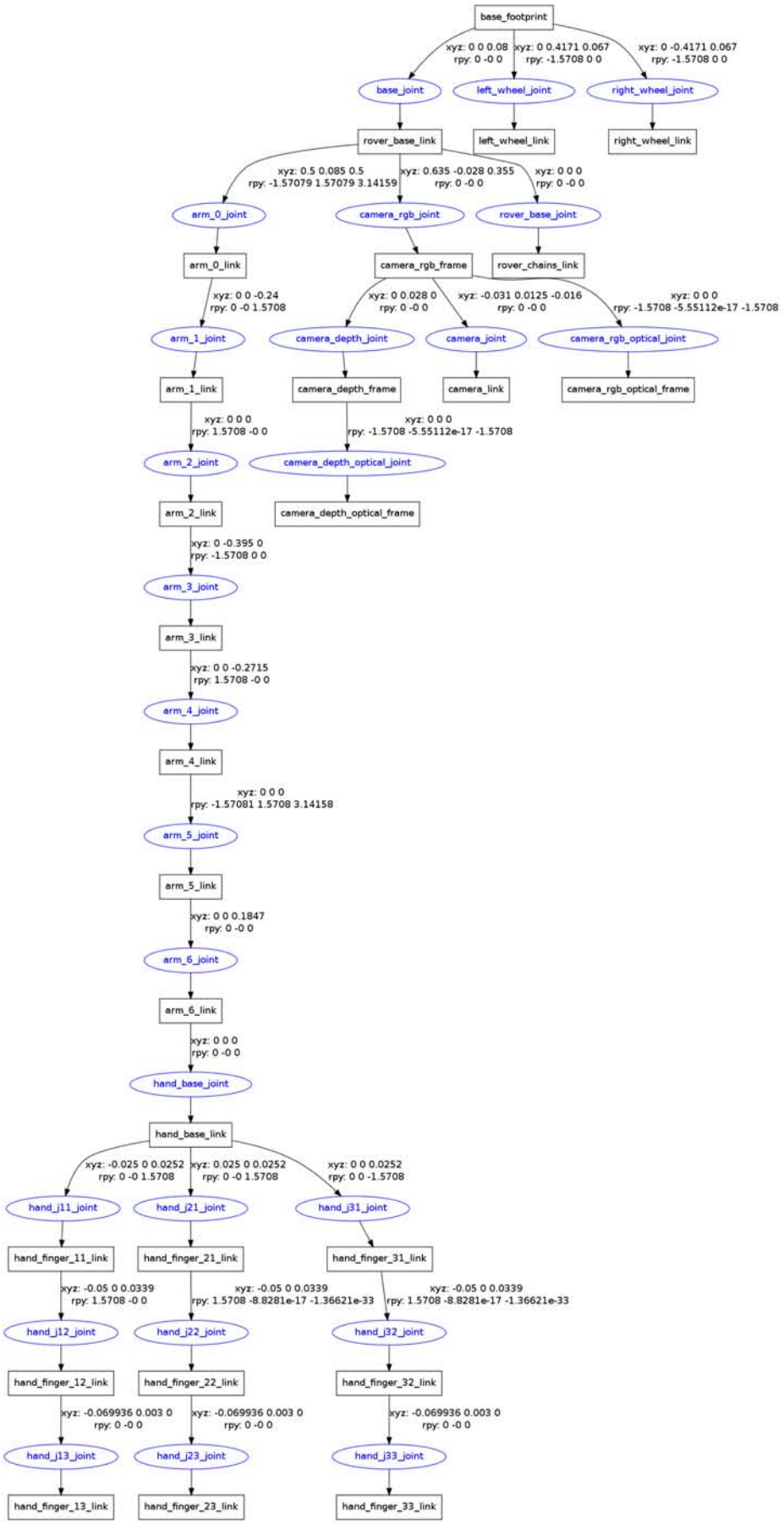


Figura 4.4.: rescuer_simulator.urdf en forma gráfica

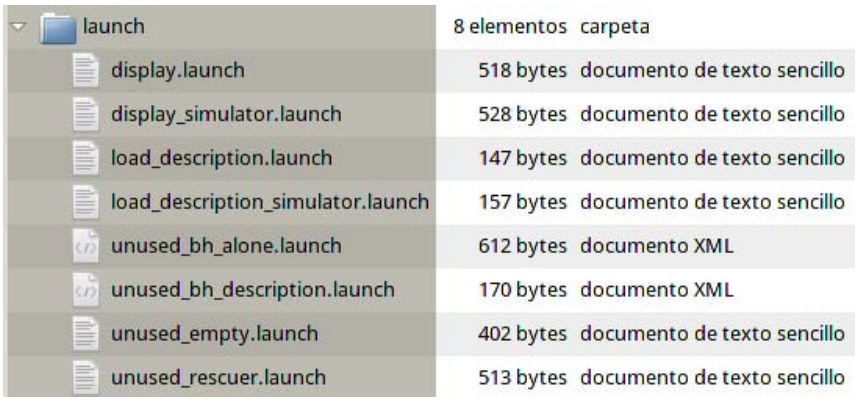


Figura 4.5.: Contenido de rescuer_description/launch

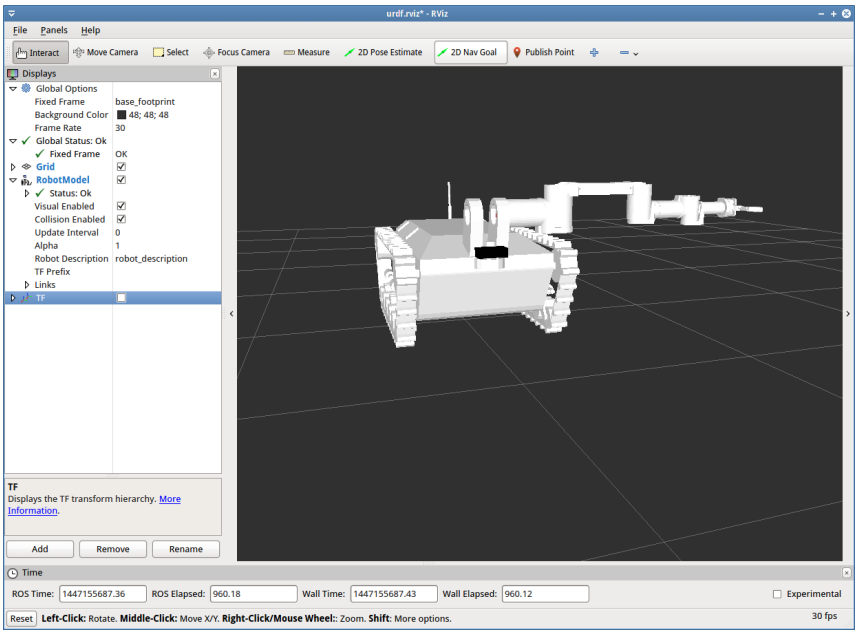


Figura 4.6.: Visualizando RESCUER

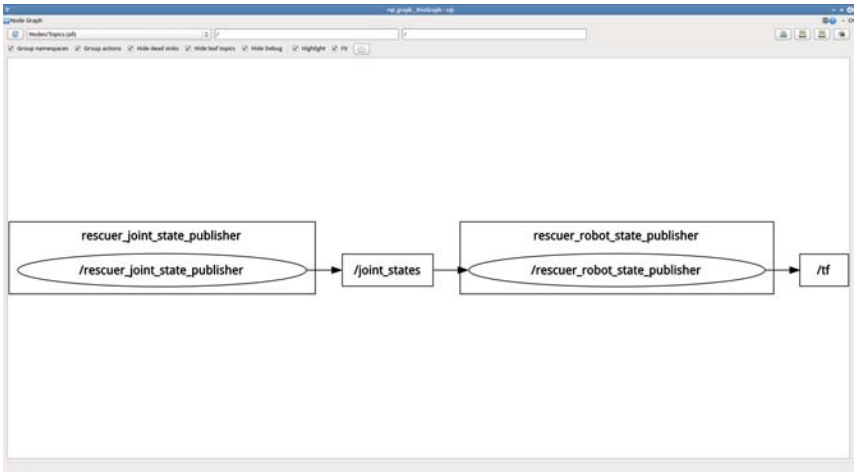


Figura 4.7.: Grafo de nodos y tópicos de ROS tras lanzar display.launch

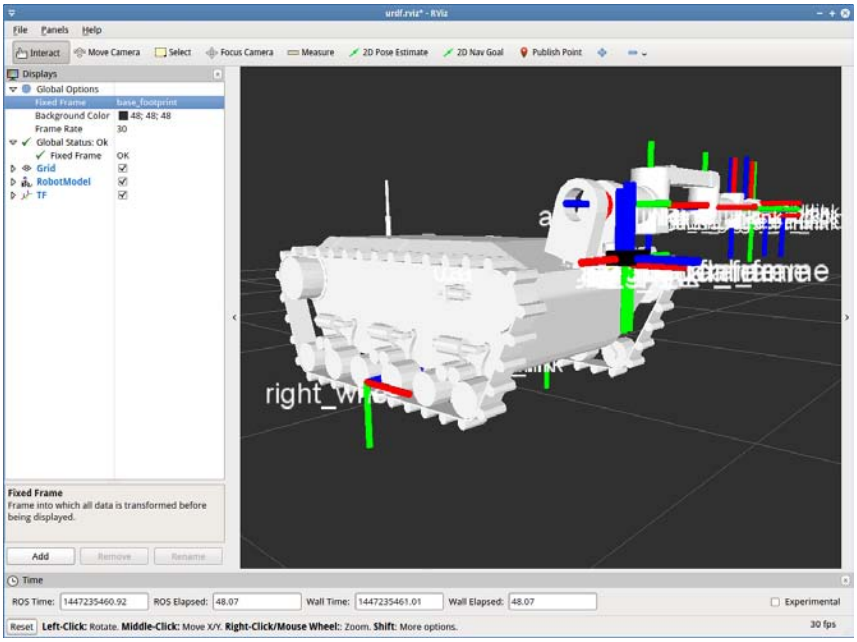


Figura 4.8.: Visualización de la descripción para el simulador

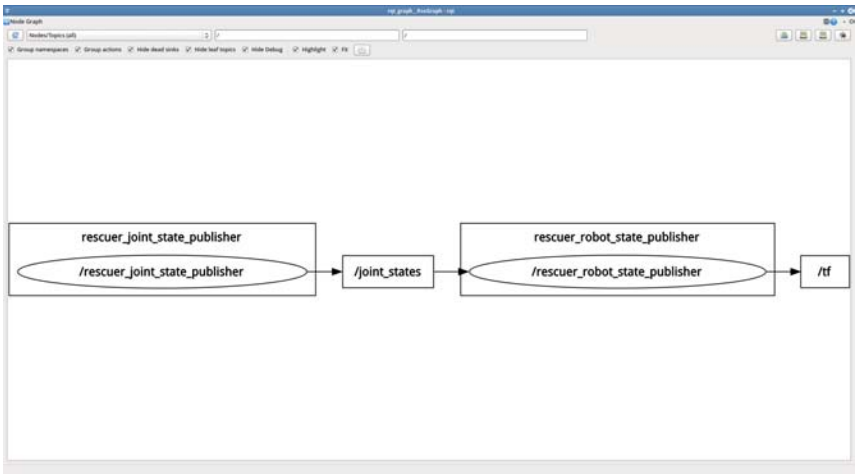


Figura 4.9.: Grafo de nodos y tópicos de ROS tras lanzar display_simulator.launch

5. Controladores

En [Leó15] ya había creado el paquete rescuer_controllers. Este paquete disponía de un controlador para la base móvil. Además se había modificado el paquete *schunk_modular_robotics* para habilitar y configurar el driver para el brazo robótico.

Se describen las diferencias con el proyecto anterior [Leó15].

5.1. Programas

Controlador de la base

Se modifica *src/base_controller.cpp (ver Apéndice A)* para que publique también información odométrica sobre el robot. Utiliza los mensajes recibidos para mover la base, para hacer un cálculo de la odometría y publicarlo en /odom.

Creación de controlador de trayectoria

Se crea un programa *src/joint_action_trajectory_controller.cpp* (ver Apéndice A). Este programa recibe una lista de puntos 3D para cada una de las articulaciones del brazo y envía las ordenes de velocidad adecuadas al controlador del brazo.

Creación de controlador de trayectoria para el simulador

Se crea un programa *src/joint_action_trajectory_controller_simulator.cpp* . Este programa recibe una lista de puntos 3D para cada una de las articulaciones del brazo y envía las ordenes de velocidad adecuadas al simulador.

Se descarta el programa y se propone para su eliminación puesto que se puede configurar *Gazebo* para que utilice un plugin que sustituye al controlador de trayectoria.

5.2. Configuración del brazo

En el directorio *params* del paquete existen ahora dos configuraciones para el controlador del brazo.

- powercube_chain_moveStep.yaml
- powercube_chain.yaml

La única diferencia entre ambas configuraciones es que en la queda como definitiva, *powercube_chain.yaml,* se fuerza el funcionamiento del brazo en modo comandos de velocidad y no modo paso a paso.

Las versión paso a paso se guarda para tareas de depuración y pruebas en el brazo.

¡Atención! Al utilizar el brazo en modo velocidad, este se sigue moviendo a la velocidad que le indica el último mensaje recibido. Hay que mandar un mensaje con velocidades cero o un mensaje para parar el brazo.

5.3. Lanzadores

En el directorio *launch* están ubicados los lanzadores para lanzar los controladores del robot de manera individual y de forma conjunta. Se detallan a continuación.

arm_joint_trajectory_controller.launch

Lanza el controlador de trayectoria del brazo.

```
<launch>
    <node ns="arm_controller" name="follow_joint_trajectory" pkg="
        rescuer_controllers" type="joint_trajectory_action_controller"
        output="screen"/>
</launch>
```

base_controller.launch

Lanza el controlador de la base móvil.

```
<launch>
    <node name="base_controller" pkg="rescuer_controllers" type="
        base_controller" output="screen" />
</launch>
```

gdb_arm_joint_trajectory_controller.launch

Lanza el controlador de trayectoria del brazo en modo depuración.

```
<launch>
    <node ns="arm_controller" name="follow_joint_trajectory" pkg="
        rescuer_controllers" type="joint_trajectory_action_controller"
        output="screen" launch-prefix="gdb -ex run --args" />
</launch>
```

kinect_controller.launch

Lanza el controlador de la Kinect, usando el ya existente en el paquete *freenect*.

```
<launch>
    <include file="$(find freenect_launch)/launch/freenect.launch">
                <arg name="publish_tf" value="false" />

        <!-- use device registration -->
        <arg name="depth_registration"              value="true" />
        <arg name="rgb_processing"                  value="true" />
        <arg name="ir_processing"                   value="false" />
        <arg name="depth_processing"                value="false" />
        <arg name="depth_registered_processing"     value="true" />
        <arg name="disparity_processing"            value="false" />
        <arg name="disparity_registered_processing" value="false" />
        <arg name="sw_registered_processing"        value="false" />
        <arg name="hw_registered_processing"        value="true" />
        </include>
</launch>
```

powercube_chain_controller.launch

Lanza el controlador del brazo mediante la carga de la configuración del robot en el espacio de nombres adecuado para el controlador de powercube, la carga de la configuración del brazo y la propia llamada al controlador schunk_powercube_chain.

```
<launch>
    <param name="powercube_chain/robot_description" command="$(find
        xacro)/xacro.py $(find rescuer_description)/urdf/rescuer.urdf.
        xacro" />
    <rosparam command="load" file="$(find rescuer_controllers)/params/
        powercube_chain.yaml" />
    <remap from="/joint_states" to="/powercube_chain/joint_states" />
    <node name="powercube_chain" pkg="schunk_powercube_chain" type="
        schunk_powercube_chain" output="screen" launch-prefix="$(find
        rescuer_controllers)/powercube_chain_init"/>
</launch>
```

La llamada al driver hace uso de un fichero que se encuentra en el directorio raíz de *rescuer_controllers*. Este fichero llama al servicio que inicializa el brazo.

```
#!/bin/bash
rosservice call --wait /powercube_chain/init &
exec "$@"
```

rescuer_controllers.launch

Lanza todos los controladores:

- Base móvil.
- Brazo robótico.
- Kinect.
- Trayectoria del brazo.

```
<launch>
        <include file="$(find rescuer_controllers)/launch/
            base_controller.launch" />
        <include file="$(find rescuer_controllers)/launch/
            powercube_chain_controller.launch" />
        <include file="$(find rescuer_controllers)/launch/
            kinect_controller.launch" />
        <include file="$(find rescuer_controllers)/launch/
            arm_joint_trajectory_controller.launch" />
</launch>
```

rescuer_controllers_moveStep.launch

Igual que *rescuer_controllers.launch* pero lanza el controlador del brazo con control paso a paso.

```
<launch>
        <param name="powercube_chain/robot_description" command="$(find
             xacro)/xacro.py $(find rescuer_description)/urdf/rescuer.
            urdf.xacro" />
    <rosparam command="load" file="$(find rescuer_controllers)/params/
        powercube_chain_moveStep.yaml" />

    <node name="base_controller" pkg="rescuer_controllers" type="
        base_controller" output="screen" />
    <node name="powercube_chain" pkg="schunk_powercube_chain" type="
        schunk_powercube_chain" output="screen" launch-prefix="$(find
        rescuer_controllers)/powercube_chain_init"/>
</launch>
```

rescuer_controllers_all_in_one.launch

No utilizado, propuesto para eliminar.

arm_joint_trajectory_controller_simulator.launch

Lanza el controlador de trayectoria del brazo para el simulador. No se utiliza y está propuesto para eliminación.

gdb_joint_trajectory_action_controler_simulator.launch

No utilizado, propuesto para eliminar.

6. Puesta en marcha

El paquete *rescuer_bringup.*

Este paquete sirve como interfaz para los controladores del robot, sirve también para cargar los parámetros del robot y en él se construyen algunos programas auxiliares para operar con éxito el robot.

6.1. Programas auxiliares

Puesta a cero del brazo

Se crea *src/ArmToZero.cpp que* es un programa para poner todos articulaciones del brazo en posición cero. Se recomienda poner el brazo en posición cero antes de apagar el robot.

Este programa no realiza ningún control de colisiones ni de trayectoria. Solamente es conveniente utilizarlo cuando el brazo se encuentra cercano a la posición cero antes de apagar el robot.

Publicar la odometría del robot

Se crea *src/odometry_publisher.cpp* para intentar publicar la odometría del robot en función de los comandos de velocidad recibidos. Se descarta. La publicación de la odometría se realiza en el controlador de la base móvil.

6.2. Lanzadores

En el directorio *launch* están ubicados los lanzadores para lanzar los controladores del robot sin usar directamente el paquete *rescuer_controllers.* Se recomienda utilizar este paquete como capa de abstracción sobre los controladores hardware.

arm_to_zero.launch

Pone todos los joints del brazo en posición cero.

Contenido:

```
<launch>
    <node ns="arm_to_zero" name="arm_to_zero" pkg="rescuer_bringup"
        type="ArmToZero" output="screen"/>
</launch>
```

Llamada:

```
roslaunch rescuer_bringup arm_to_zero.launch
```

gdb_arm_to_zero.launch

Similar al anterior para tareas de depuración usando gdb.

rescuer_bringup_arm_joint_trajectory.launch

Carga el controlador de trayectoria del brazo.

Contenido:

```
<launch>
    <include file="$(find␣rescuer_controllers)/launch/
        arm_joint_trajectory_controller.launch" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup
   rescuer_bringup_arm_joint_trajectory.launch
```

rescuer_bringup_base.launch

Carga el controlador de la base móvil.

Contenido:

```
<launch>
    <include file="$(find␣rescuer_controllers)/launch/base_controller.
        launch" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_bringup_base.launch
```

rescuer_bringup_kinect.launch

Carga el controlador de la Kinect.

Contenido:

```
<launch>
    <include file="$(find␣rescuer_controllers)/launch/kinect_controller
        .launch" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_bringup_kinect.launch
```

rescuer_bringup_powercube_chain.launch

Carga el controlador del brazo robótico.

Contenido:

```
<launch>
    <include file="$(find␣rescuer_controllers)/launch/
        powercube_chain_controller.launch" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_bringup_powercube_chain.
   launch
```

rescuer_joints_state_publisher.launch

Carga un nodo de tipo joint_state_publisher, con la particularidad de que redirige el tópico joint_states generado por el controlador del brazo.

Contenido:

```
<launch>
    <node name="joint_state_publisher" pkg="joint_state_publisher" type
        ="joint_state_publisher">
         <rosparam param="source_list">["/powercube_chain/joint_states"]
            </rosparam>
    </node>
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_joints_state_publisher.
   launch
```

rescuer_load_robot_description.launch

Carga la descripción del robot.

Contenido:

```
<launch>
    <param name="robot_description" command="$(find␣xacro)/xacro.py␣$(
        find␣rescuer_description)/urdf/rescuer.urdf.xacro" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_load_robot_description.
   launch
```

rescuer_robot_state_publisher.launch

Carga un nodo de tipo *robot_state_publisher*, que lee de *joint_states* y publica en *tf*.

Contenido:

```
<launch>
     <node name="robot_state_publisher" pkg="robot_state_publisher"
         type="robot_state_publisher" respawn="true" output="screen" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer_robot_state_publisher.
   launch
```

rescuer.launch

Es el punto de entrada para el robot real. Arranca los controladores hardware del robot. Debe ejecutarse en el robot real antes de empezar a trabajar con él.

Realiza las siguientes tareas:

- Carga la descripción de *RESCUER.*
- Carga el controlador de trayectoria del brazo.
- Carga el controlador de la base móvil.
- Carga el controlador de la Kinect.
- Carga el controlador del brazo.
- Arranca un nodo que publica las articulaciones del brazo.
- Arranca un nodo que publica las transformaciones entre las articulaciones del robot.

Contenido:

```
<launch>
    <include file="$(find rescuer_bringup)/launch/
        rescuer_load_robot_description.launch" />
    <include file="$(find rescuer_bringup)/launch/
        rescuer_bringup_arm_joint_trajectory.launch" />
    <include file="$(find rescuer_bringup)/launch/rescuer_bringup_base.
        launch" />
    <include file="$(find rescuer_bringup)/launch/
        rescuer_bringup_kinect.launch" />
    <include file="$(find rescuer_bringup)/launch/
        rescuer_bringup_powercube_chain.launch" />
    <include file="$(find rescuer_bringup)/launch/
        rescuer_joints_state_publisher.launch" />
    <include file="$(find rescuer_bringup)/launch/
        rescuer_robot_state_publisher.launch" />
</launch>
```

Llamada:

```
roslaunch rescuer_bringup rescuer.launch
```

6.3. Scripts

En el directorio bin del paquete se encuentran unos ejecutables que se pueden copiar a */usr/local/bin*. Estos ejecutables son:

Puesta en marcha de RESCUER. rescuer-startup.sh

Mediante este ejecutable se lanzan todos los controladores para el robot. Hace uso del programa *screen*. Proporciona una forma muy cómoda de lanzar los controladores de forma que se pueda consultar que es lo que hace cada controlador en un terminal individual.

Contenido:

```
#!/bin/bash
# file: rescuer-startup.sh

echo "Launch roscore and load rescuer description"
screen -dmS rescuer-roscore rescuer-bringup-load-description.sh
sleep 15
echo "Bringup base"
screen -dmS rescuer-base rescuer-bringup-base.sh
echo "Bringup powecube"
screen -dmS rescuer-powercube rescuer-bringup-powercube.sh
echo "Bringup follow joint trajectory controller"
screen -dmS rescuer-arm-joint rescuer-bringup-arm-joint-trajectory.sh
echo "Bringup kinect"
screen -dmS rescuer-kinect rescuer-bringup-kinect.sh
echo "Bringup joints state publisher"
screen -dmS rescuer-joints-state-publisher rescuer-joints-state-
    publisher.sh
echo "Bringup robot state publisher"
screen -dmS rescuer-robot-state-publisher rescuer-robot-state-publisher
    .sh
echo "done. view with screen -ls"
```

Llamada:

```
rescuer-startup.sh
```

Los terminales que se ponen en marcha se pueden consultar mediante:

```
screen -ls
```

Se puede acceder al terminal adecuado mediante:

```
screen -r id_terminal
```

Se puede salir del terminal:

```
Ctrl+A D
```

Controladores

Versiones ejecutables de los lanzadores vistos con anterioridad. Estos ejecutables son los que utiliza *rescuer-startup.sh*

- rescuer-bringup-arm-joint-trajectory.sh
- rescuer-bringup-base.sh
- rescuer-bringup-kinect.sh
- rescuer-bringup-load-description.sh
- rescuer-bringup-powercube.sh
- rescuer-joints-state-publisher.sh
- rescuer-robot-state-publisher.sh
- rescuer-roscore.sh

Proxy

Ejemplos de enrutamiento para proporcionar a RESCUER conexión a Internet.

- rescuerRoute2.sh
- rescuerRoute3.sh
- rescuerRoute.sh

.bashrc

Al fichero .bashrc del PC de control de RESCUER se le añaden las siguientes líneas.

```
export PATH=$PATH:/home/rescuer/development/ros/
   rescuer_catkin/src/rescuer_bringup/bin
source /opt/ros/hydro/setup.bash
source /home/rescuer/development/ros/rescuer_catkin/devel/
   setup.bash
export ROS_IP=192.168.1.100
```

.bashrc en PC remoto

Al fichero .bashrc de los equipos que quieran interactuar con RESCUER de forma remota deben añadirse las siguientes líneas.

```
source /opt/ros/hydro/setup.bash
source /home/robinlab/rescuer_catkin/rescuer_catkin/devel/
   setup.bash
export ROS_MASTER_URI=http://192.168.1.100:11311/
export ROS_IP=192.168.1.101
export LIBGL_ALWAYS_SOFTWARE=1
```

6.4. Conclusiones

Para habilitar todos los controladores en el robot hay que ejecutar una de estas dos opciones:

```
roslaunch rescuer_bringup rescuer.launch
```

o

```
rescuer-startup.sh
```

En la Figura 6.1 se pueden ver los nodos de ROS que generan los controladores de RESCUER.

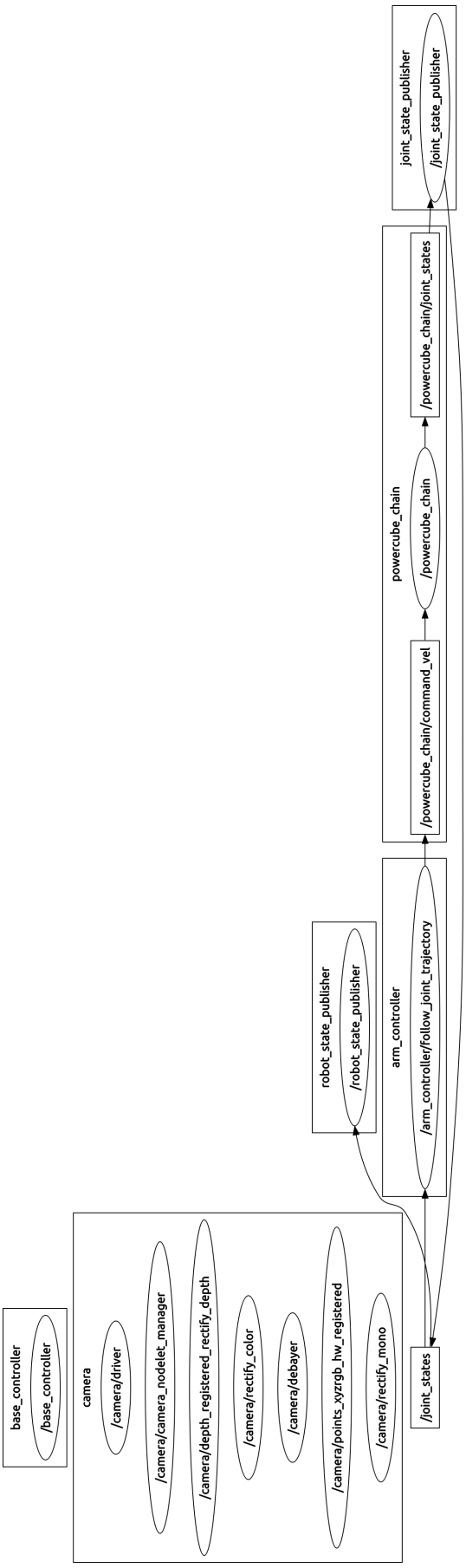


Figura 6.1.: Controladores de RESCUER

7. Planificación. MoveIt!.

Se utiliza como guía "Integrating a new robot with MoveIt!" de [ROS15b].

El primer paso es instalar MoveIt!

```
sudo apt-get install ros-hydro-moveit-full
```

7.1. Asistente MoveIt!

El asistente de *MoveIt!* se lanza mediante:

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

Inicio

En la primera pantalla (ver Figura 7.1) del asistente se puede crear una configuración para un robot nuevo o editar una ya existente. En este caso veremos la configuración que se ha creado para *RESCUER* en este proyecto.

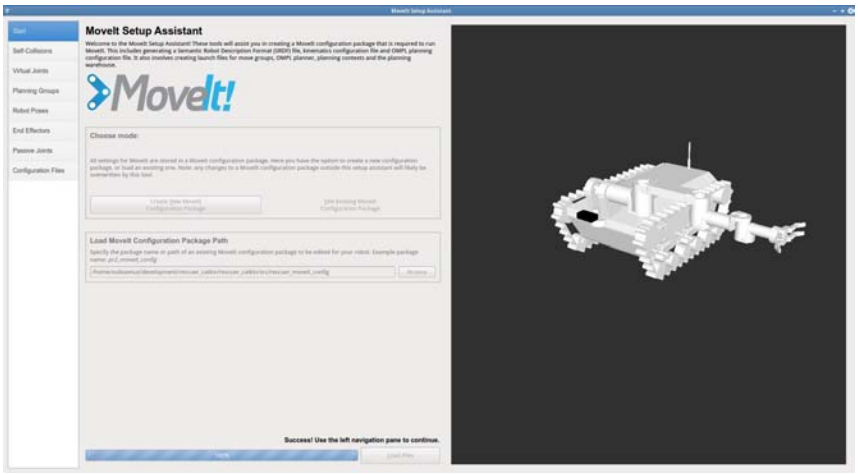
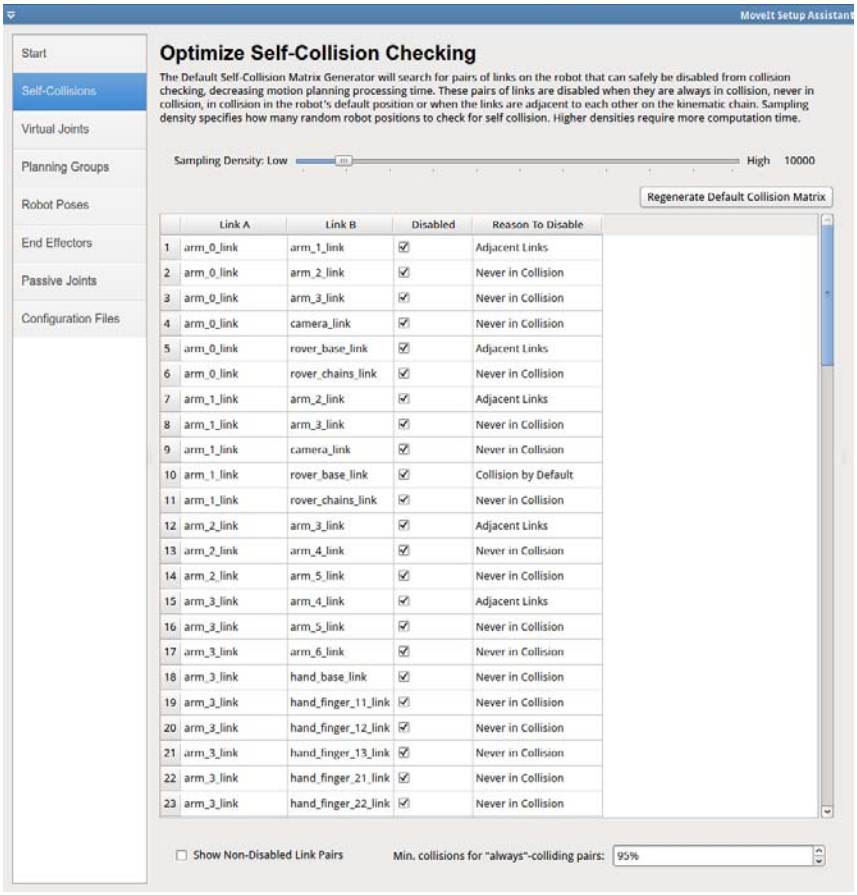


Figura 7.1.: MoveIt! Setup Assistant Start

Matriz de colisiones propias

En este paso (ver Figura 7.2) se realiza una búsqueda de colisiones entre las distintas partes que conforman el robot.



	Link A	Link B	Disabled	Reason To Disable
1	arm_0_link	arm_1_link	☑	Adjacent Links
2	arm_0_link	arm_2_link	☑	Never in Collision
3	arm_0_link	arm_3_link	☑	Never in Collision
4	arm_0_link	camera_link	☑	Never in Collision
5	arm_0_link	rover_base_link	☑	Adjacent Links
6	arm_0_link	rover_chains_link	☑	Never in Collision
7	arm_1_link	arm_2_link	☑	Adjacent Links
8	arm_1_link	arm_3_link	☑	Never in Collision
9	arm_1_link	camera_link	☑	Never in Collision
10	arm_1_link	rover_base_link	☑	Collision by Default
11	arm_1_link	rover_chains_link	☑	Never in Collision
12	arm_2_link	arm_3_link	☑	Adjacent Links
13	arm_2_link	arm_4_link	☑	Never in Collision
14	arm_2_link	arm_5_link	☑	Never in Collision
15	arm_3_link	arm_4_link	☑	Adjacent Links
16	arm_3_link	arm_5_link	☑	Never in Collision
17	arm_3_link	arm_6_link	☑	Never in Collision
18	arm_3_link	hand_base_link	☑	Never in Collision
19	arm_3_link	hand_finger_11_link	☑	Never in Collision
20	arm_3_link	hand_finger_12_link	☑	Never in Collision
21	arm_3_link	hand_finger_13_link	☑	Never in Collision
22	arm_3_link	hand_finger_21_link	☑	Never in Collision
23	arm_3_link	hand_finger_22_link	☑	Never in Collision

Figura 7.2.: MoveIt! Setup Assistant Self Collision Matrix

Articulaciones virtuales

Es necesario definir una articulación virtual para conseguir relacionar odom_combined con base_footprint (ver Figura 7.3). Esta relación se utilizará en la planificación y también en el simulador.

Grupos a planificar

En este paso (ver Figura 7.4) se definen los grupos de los que posteriormente se podrán realizar tareas de planificación automática.

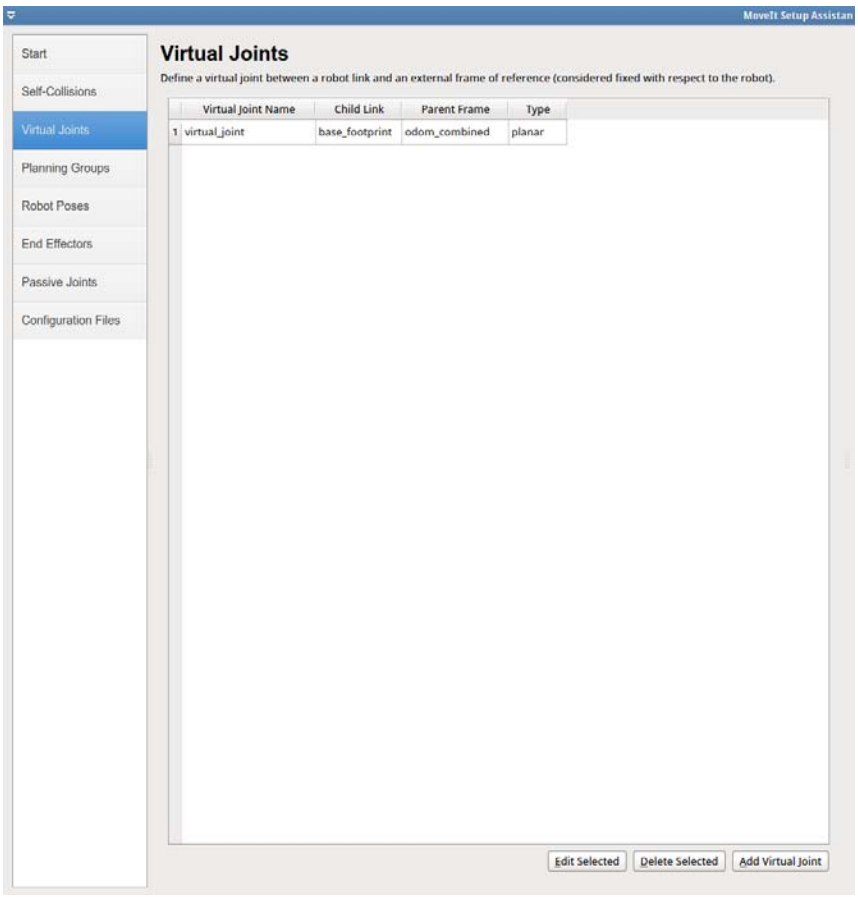


Figura 7.3.: MoveIt! Setup Assistant Virtual Joints

Para cada grupo que se añade hay que elegir que algoritmo cinemático se utilizará. En este caso se utiliza "kdl_kinematics_plugin/KDLKinematicsPlugin"

Se define un grupo para el brazo, otro grupo para cada uno de los dedos de la mano y otro grupo para la mano formado por los subgrupos de los dedos.

Poses

Se definen algunas poses interesantes para trabajar con el robot (ver Figura 7.5):

- Brazo con todas las articulaciones a cero (arm_zero, ver Figura 7.6).
- Brazo en posición de trabajo (arm_home, ver Figura 7.7).
- Brazo en posición de navegación (arm_navigation, ver Figura 7.8).
- Mano abierta (hand_open, ver Figura 7.9).
- Mano cerrada (hand_open, ver Figura 7.10).

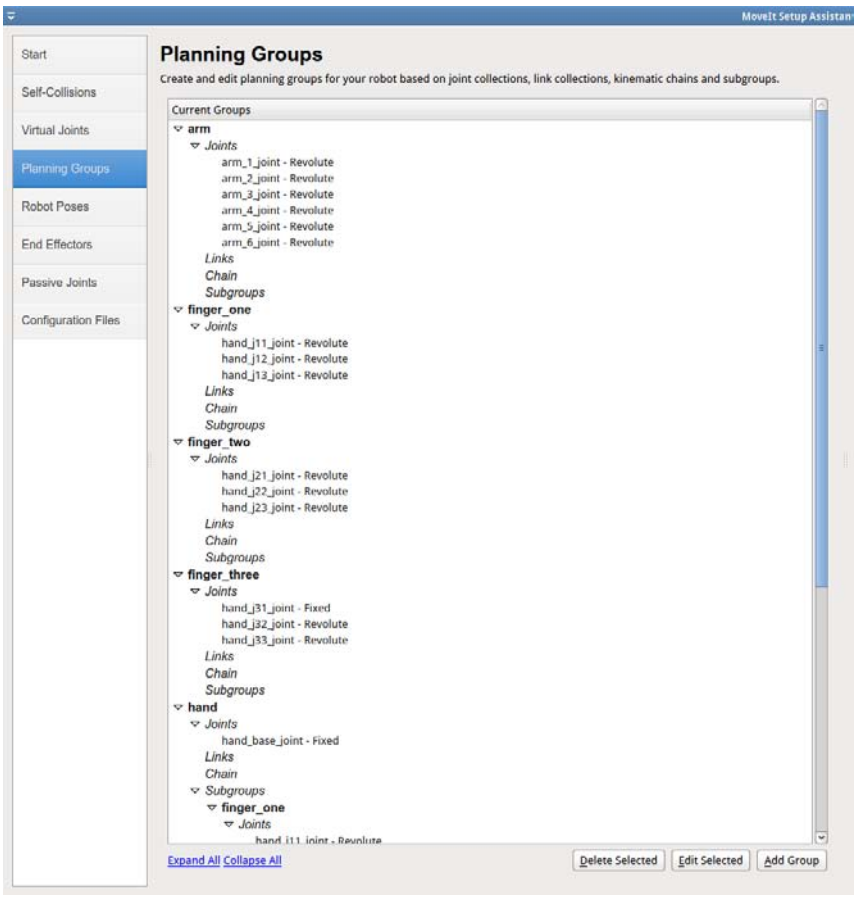


Figura 7.4.: MoveIt! Setup Assistant Planning Groups

Se puede aprovechar este paso para comprobar los límites de las articulaciones (ver Figura 7.11).

Intentar mover todas las articulaciones para ver si os límites de las mismas están bien.

Manipulador

Se define con el nombre *hand_end_effector* un manipulador perteneciente al grupo hand (ver Figura 7.12).

Articulaciones pasivas

No se define ninguna articulación pasiva para *RESCUER* (ver Figura 7.13).

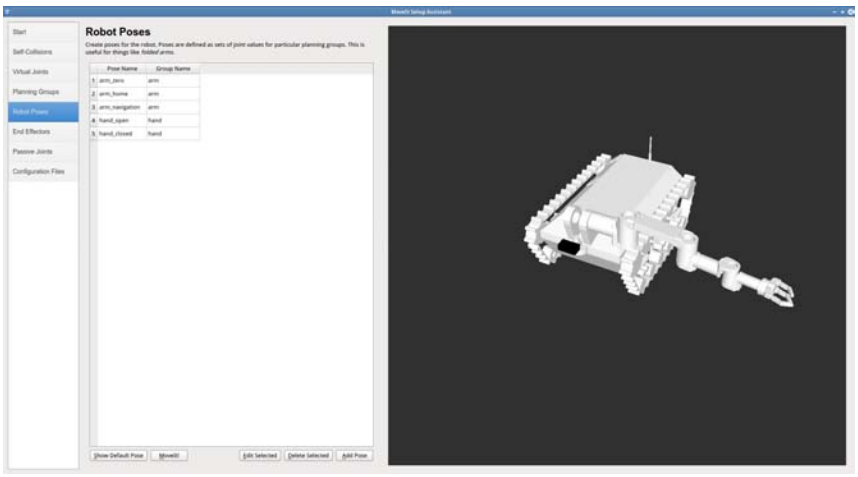


Figura 7.5.: MoveIt! Setup Assistant - Robot Poses

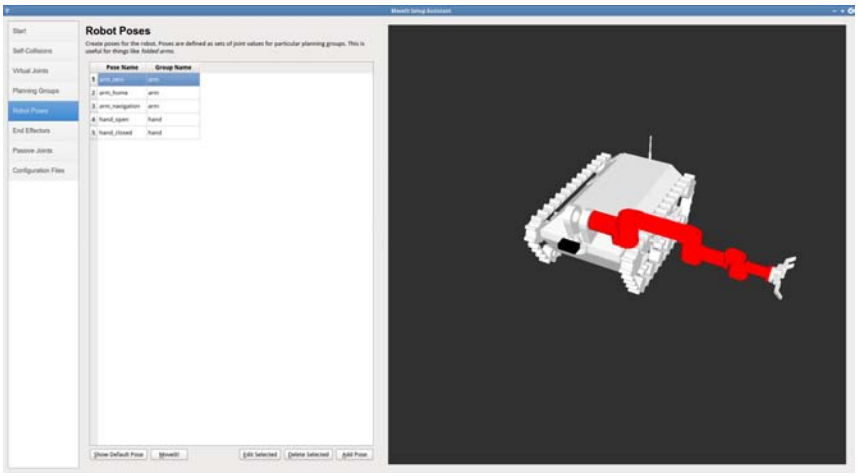


Figura 7.6.: MoveIt! Setup Assistant - Robot Poses - Arm Zero

Generación de la configuración

En este último paso (ver Figura 7.14) se genera el paquete de configuración de MoveIt! para *RESCUER*. En caso de ejecutar el asistente de MoveIt! sobre una configuración existente conviene hacer una copia del paquete antes de guardar los cambios. Posteriormente con una herramienta como *meld* se puede comprobar las diferencias entre la configuración nueva y la antigua.

7.2. Descripción del paquete

Una vez generado el paquete con el asistente de MoveIt! se describe la configuración modificada manualmente y los lanzadores creados para RESCUER.

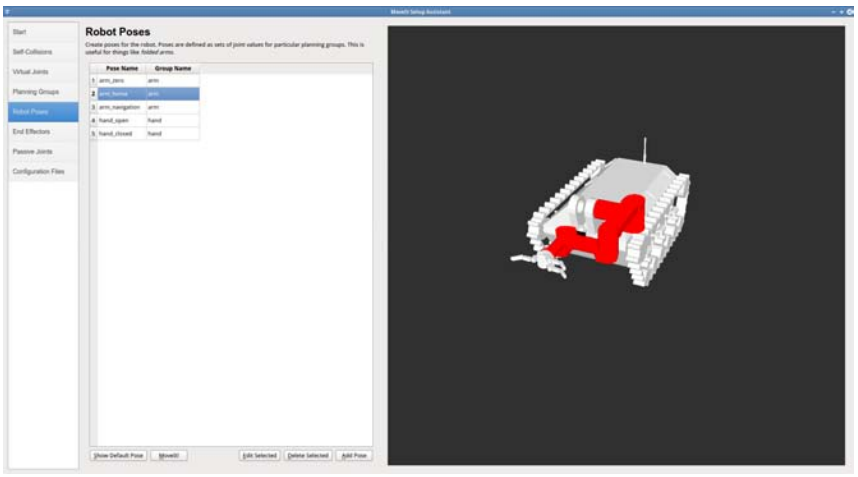


Figura 7.7.: MoveIt! Setup Assistant - Robot Poses - Arm Home

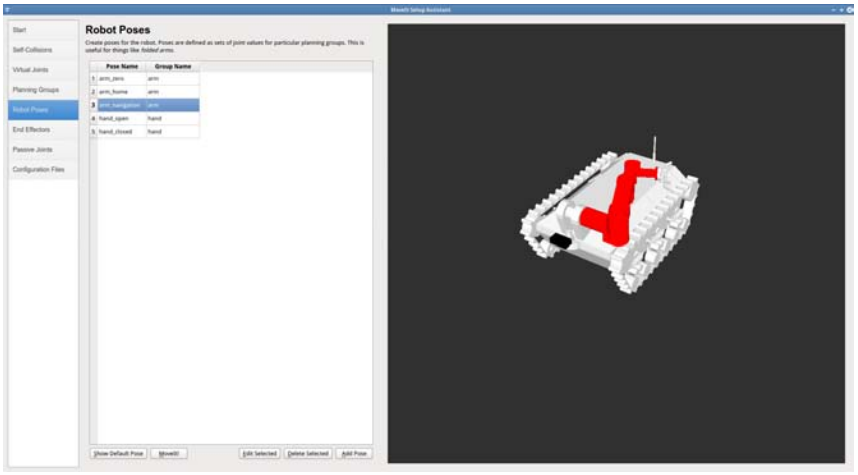


Figura 7.8.: MoveIt! Setup Assistant - Robot Poses - Arm Navigation

Lanzadores

rescuer.launch

- Carga *planning_context.launch* del mismo paquete, que a su vez:
  - Carga la configuración del robot creada por el asistente en formato *srdf*, el fichero cargado es *config/rescuer.srdf.*
    - Sobrescribe los límites de las articulaciones utilizando el fichero *config/joint_limits.yaml*
    - Carga la configuración por defecto para la resolución de problemas cinemáticos utilizando el fichero *config/kinematics.yaml.*
- Pone en marcha el nodo *move_group,* mediante el lanzador *launch/move_group.launch*

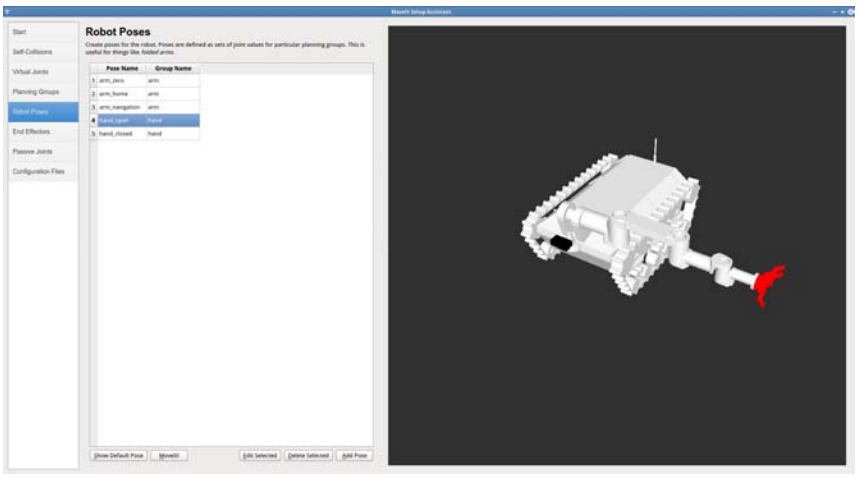


Figura 7.9.: MoveIt! Setup Assistant - Robot Poses - Hand Open

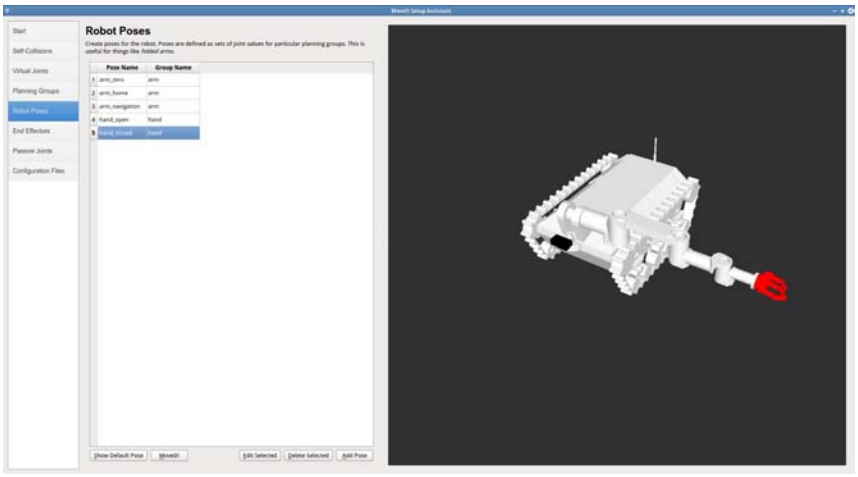


Figura 7.10.: MoveIt! Setup Assistant - Robot Poses - Hand Closed

- Pone en marcha la interfaz gráfica *RVIZ*

Llamada:

```
roslaunch rescuer_moveit_config rescuer.launch
```

rescuer_simulator.launch

- Realiza las mismas tareas que *rescuer.launch* pero en el espacio de nombres *rescuer*, esto es necesario para la correcta interacción con el simulador Gazebo.
- Utiliza el lanzador *launch/move_group_simulator.launch.*

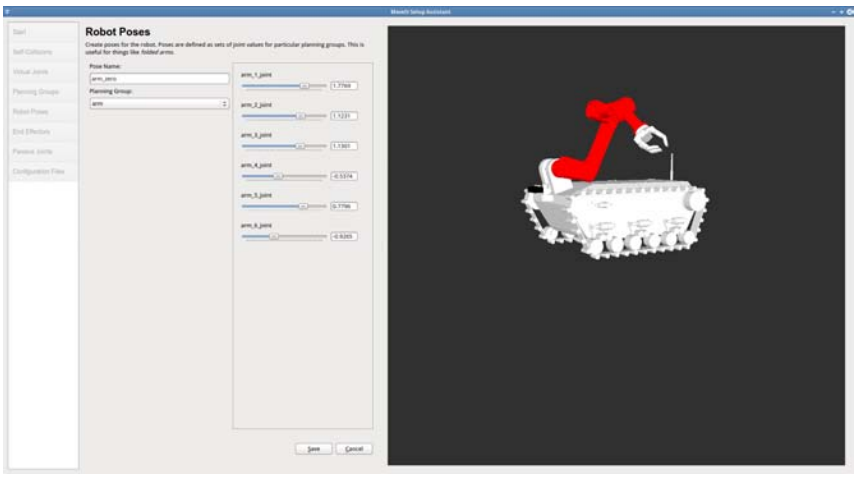


Figura 7.11.: MoveIt! Setup Assistant - Robot Poses - Comprobar límites de las articulaciones

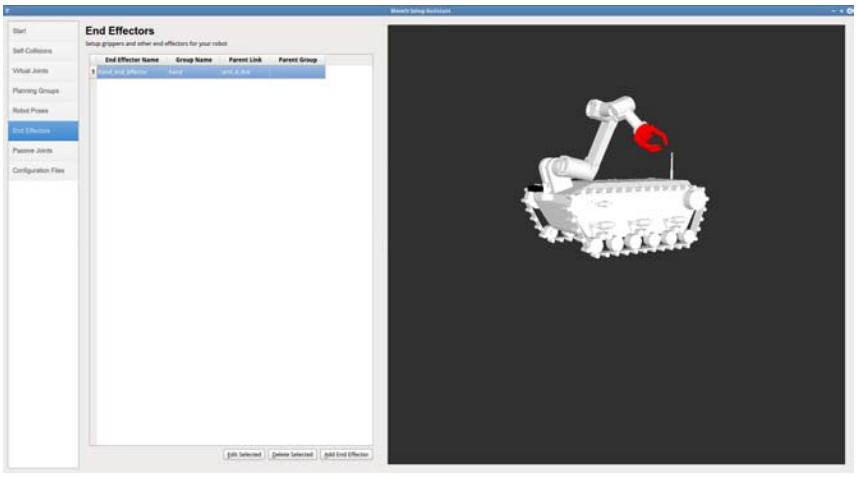


Figura 7.12.: MoveIt! Setup Assistant - End Effectors

Llamada:

```
roslaunch rescuer_moveit_config rescuer_simulator.launch
```

Otros lanzadores

- rescuer_moveit_controller_manager.launch.xml
  - Carga la configuración existente en *config/controllers.yaml*
- rescuer_moveit_sensor_manager.launch.xml

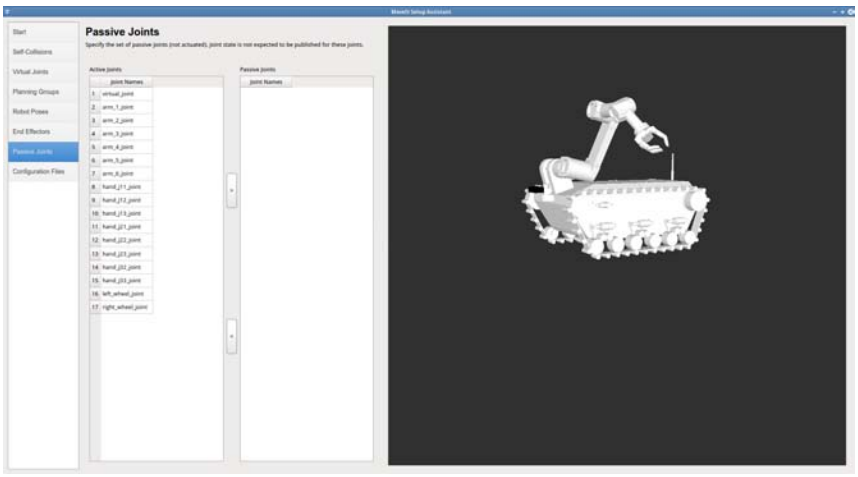


Figura 7.13.: MoveIt! Setup Assistant - Passive Joints

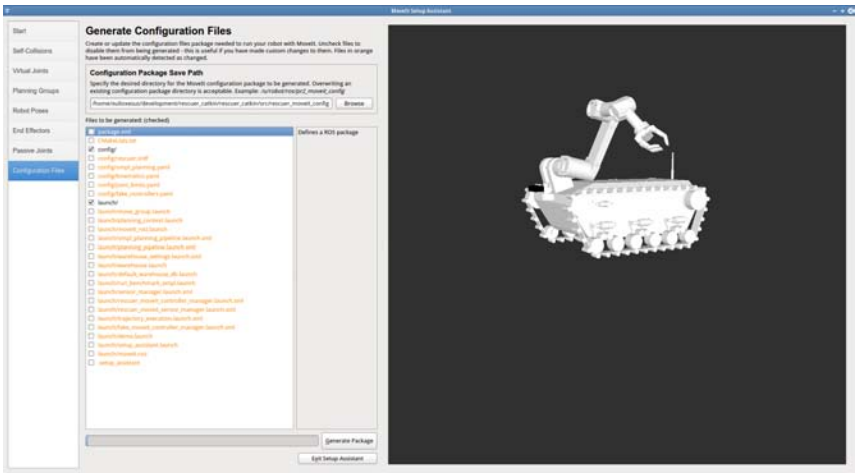


Figura 7.14.: MoveIt! Setup Assistant - Configuration Files

- Carga la configuración respecto a Kinect.

```
<launch>
    <rosparam command="load" file="$(find rescuer_moveit_config)/config
        /sensors_kinect.yaml" />
        <param name="octomap_frame" type="string" value="odom_combined"
            />
    <param name="octomap_resolution" type="double" value="0.05" />
    <param name="max_range" type="double" value="5.0" />
</launch>
```

Configuración

La configuración específica para *RESCUER,* que se utiliza en *MoveIt!*, se define en los siguientes archivos:

controllers.yaml

Se definen los controladores de trayectoria para el brazo y para la mano. El controlador del brazo existe para el robot real, pero para la mano no.

```
controller_list:
 - name: arm_controller
   action_ns: follow_joint_trajectory
   type: FollowJointTrajectory
   default: true
   joints:
     - arm_1_joint
     - arm_2_joint
     - arm_3_joint
     - arm_4_joint
     - arm_5_joint
     - arm_6_joint
 - name: hand_controller
   action_ns: follow_joint_trajectory
   type: FollowJointTrajectory
   default: true
   joints:
     - hand_j11_joint
     - hand_j12_joint
     - hand_j13_joint
     - hand_j21_joint
     - hand_j22_joint
     - hand_j23_joint
     - hand_j23_joint
     - hand_j32_joint
     - hand_j33_joint
```

joint_limits.yaml

Se definen los límites de velocidades y aceleraciones de las articulaciones. Estos valores sobreescriben los descritos en la descripción *urdf* del robot.

```
joint_limits:
  arm_6_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.01
  arm_4_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.1
  arm_1_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.01
  arm_2_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.01
  arm_5_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.01
  arm_3_joint:
    has_velocity_limits: true
    max_velocity: 0.49
    has_acceleration_limits: true
    max_acceleration: 0.01
...
```

kinematics.yaml

La configuración cinemática para resolver problemas de planificación se mantiene tal y como la genera el asistente de *MoveIt!*

sensors_kinect.yaml

La configuración de la Kinect. El tópico point_cloud_topic se tiene que modificar dependiendo del espacio de nombres en el que se lance *move_group.*

```
sensors:
  - sensor_plugin: occupancy_map_monitor/PointCloudOctomapUpdater
    point_cloud_topic: /rescuer/camera/depth_registered/points
    max_range: 5.0
    point_subsample: 1
    padding_offset: 0.1
    padding_scale: 1.0
    filtered_cloud_topic: filtered_cloud
```

7.3. Planificando con el robot real

Los componentes que se establecieron como necesarios para realizar tareas de planificación con MoveiIt! en el Capítulo 2 eran:

- ☑ Controlador de la base
  - Proporcionado por *rescuer_co*ntrollers.
- ☑ Controlador del brazo robótico.
  - Proporcionado por *rescuer_co*ntrollers.
- ☑ Un controlador capaz de seguir trayectorias para el brazo.
  - Proporcionado por *rescuer_co*ntrollers.
- ☑ Un sensor de puntos 3D.
  - Instalada Kinect
  - Controlador proporcionado por *rescuer_controllers.*
- ☑ Publicar el estado de las articulaciones del robot.
  - Proporcionado por *rescuer_bringup*
- ☑ Publicar el estado del robot y las transformaciones entre las distintas articulaciones del robot.
  - Proporcionado por *rescuer_bringup*

Ahora que se cumplen todos los requisitos se pueden realizar tareas de planificación.

Para hacer esto se tienen que lanzar los controladores y nodos necesarios en el robot real.

```
roslaunch rescuer_bringup rescuer.launch
```

Se lanza MoveIt! con la configuración de *RESCUER.*

```
roslaunch rescuer_moveit_config rescuer.launch
```

¡Ahora ya se pueden realizar tareas de planificación con el brazo!

7.4. Un experimento

Para ejemplificar los posibles usos de *RESCUER,* se diseña el siguiente experimento.

1. Detectar automáticamente un cilindro situado frente al robot.
2. Calcular el centroide del cilindro.
3. Mediante *MoveIt!* colocar el brazo en la posición del cilindro.

rescuer_pcl

Se crea el paquete *rescuer_pcl*.

Dentro de este paquete se crea un programa *src/detectCylinder.cpp* que procesa la nube de puntos obtenida de la Kinect y filtra un objeto cilíndrico.

El programa se lanza mediante el comando:

```
roslaunch rescuer_pcl rescuer.launch
```

rescuer_moveit_tutorials

Se crea el paquete *rescuer_moveit_tutorials.*

Dentro del paquete se crea un programa *src/move_group_tutorial.cpp* que planifica de forma automática los movimientos necesarios del brazo para conseguir la pose adecuada para ir hasta el centroide obtenido por detectCylinder del apartado anterior.

El programa se lanza mediante el comando:

```
roslaunch rescuer_moveit_tutorials planning_tutorial.launch
```

En el siguiente enlace puede verse un vídeo con la ejecución de este experimento.

```
https://drive.google.com/file/d/0B-5jNBSVP_EWNG5GaFdBZGZFd2c/
   view?usp=sharing
```

8. Simulación. Gazebo

8.1. Introducción

El punto de partida se realiza a partir del tutorial sobre ROS control, ver [ROS15f].

Para instalar *Gazebo* se selecciona la versión compatible con ROS Hydro (ver [OSF15]). La versión seleccionada es la 1.9.x que se instala utilizando los repositorios *packages.ros.org* y no *packages.osrfoundation.org*

Para utilizar *Gazebo* y *ROS* se tiene que hacer uso del metapaquete *gazebo_ros_pkgs,* en el tutorial [OSR15b] se indica el proceso de instalación de dicho paquete.

```
sudo apt-get install ros-hydro-gazebo-ros-pkgs ros-hydro-
   gazebo-ros-control
```

8.2. Control

Se crea el paquete *rescuer_control,* necesario para el simulador *Gazebo.*

config/rescuer_control.yaml

En el directorio *config* del paquete se encuentra la configuración de *rescuer_control.* En este directorio hay un fichero *rescuer_control.yaml* cuyo contenido se puede ver en la Figura 8.1. En este fichero se definen los siguientes controladores:

- joint_state_controller - Para publicar el estado de las articulaciones del robot.
- hand_controller - Se trata de un controlador de trayectoria que se encarga de las articulaciones de la mano. En el se definen también las ganancias del controlador *pid* de cada articulación.
- arm_controller - Se trata de un controlador de trayectoria que se encarga de las articulaciones del brazo. En el se definen también las ganancias del controlador *pid* de cada articulación.
- base_controller - Se trata de un controlador diferencial para la base del robot. Es el resultado de intentar mover la base en el simulador, no funciona en este momento y por ello se encuentra comentado.

launch/rescuercontrol.launch

Para lanzar los controladores definidos en *config/rescuer_control.yaml se utiliza launch/rescuercontrol.launch* que puede verse en la Figura 8.2. Aunque este fichero launch no se lanza directamente, se podría hacer de la forma siguiente:

```
roslaunch rescuer_control rescuercontrol.launch
```

Las tareas que realiza este fichero:

- Cargar la configuración de rescuer_control.
- Cargar los controladores.
  - mano.
  - brazo.
  - estado de las articulaciones
- Publicar el estado de las articulaciones del robot, con sus respectivas transformadas.

La llamada a este lanzador se hace desde el paquete *rescuer_simulator*, no siendo necesario utilizarlo de forma directa.

8.3. Descripción

Para poder mover la base en el simulador utilizando el controlador diferencial que proporciona *Gazebo*, se ha modificado la descripción del robot añadiendo dos ruedas con sus respectivas articulaciones. Aunque se intenta que no aparezcan visibles en el simulador, no se ha conseguido. La modificación sólo se realiza para la descripción que utiliza el simulador, no afectando al resto de componentes del proyecto.

Plugins

En la descripción también se añaden los elementos necesarios para que *Gazebo* pueda funcionar. Los fragmentos de la descripción más relevantes para el funcionamiento del simulador, se describen a continuación:

- Plugin para utilizar ros_control (Figura 8.3).
- Plugin para utilizar los controladores de trayectoria para la mano y el brazo (Figura 8.3).
- Plugin para la Kinect (Figura 8.4).
- Plugin para el controlador diferencial para la base móvil (Figura 8.5).

Transmisiones

Otra información relevante para *Gazebo* de la descripción del robot, es la que se refiere a las transmisiones para las articulaciones. A modo de ejemplo se muestra en la Figura 8.6.

Inercias

Tras poner en marcha el simulador el brazo se mueve compulsivamente sin poder ejercer ningún tipo de control sobre las articulaciones del mismo. Tras investigar las causas se procede a editar la descripción del robot y corregir las masas y matrices de inercia de las articulaciones del brazo.

Para calcular las matrices de inercia se utiliza el tutorial [OSR15a], utilizando *meshlab y octave.* Los cálculos realizados pueden verse en el Apéndice B.

8.4. Simulador

Se crea el paquete *rescuer_simulator.*

De los ficheros que componen el paquete, sólo tiene relevancia el lanzador del simulador *launch/gazebo.launch.*

- *config/gazebo_ros_control.yaml*
  - Intento de configuración de las ganancias del controlador de la base móvil. Se utiliza en su lugar el plugin del controlador diferencial visto en la *Sección 8.3.*
- *src/state_publisher.cpp*
  - Ejecutable para publicar las transformadas entre las articulaciones del robot. No se utiliza. En su lugar es *Gazebo* quien publica las transformadas mediante *rescuer_control.*

launch/gazebo.launch

Las tareas realizadas por *gazebo.launch* son las siguientes:

- Carga la descripción del robot simulado en el parámetro */robot_description*
- Lanza *Gazebo* con un mundo vacío.
- Carga los controladores descritos en el apartado anterior de Control a través del *rescuer_control/launch/rescuercontrol.launch*
- Carga en *Gazebo* el modelo RESCUER, además se cargan los plugins escritos en la descripción del robot.

El contenido del archivo puede verse en la Figura 8.7.

8.5. Funcionamiento

Llegado a este punto se cumplen los requisitos que se listaban en la arquitectura necesaria para tener un robot simulado en *Gazebo.*

- ☑ Tener la descripción *URDF* completa del robot.
- ☑ Describir las transmisiones de las articulaciones en la descripción *URDF.*
- ☑ Definir los controladores a utilizar por el robot simulado, utilizando *gazebo_ros_control.*

Para poner en marcha el simulador en *Gazebo* se debe ejecutar el siguiente comando:

```
roslaunch rescuer_simulator gazebo.launch
```

Esto arranca el simulador como se puede ver en Figura 8.8.

Se pueden ver los nodos creados por Gazebo en laFigura 8.9.

8.6. Cálculo de pids

Relacionado con el comportamiento errático de las articulaciones del brazo cuando se arranca el simulador Gazebo. Se procede a realizar el cálculo de las ganancias pid de los controladores, siguiendo el siguiente procedimiento.

1. Poner en marcha RESCUER en Gazebo.
2. Utilizar el plugin de ROS para reconfiguración dinámica.
3. Modificar los valores p, i, d y i_clamp hasta obtener valores de respuesta adecuados para los comandos de velocidad enviados al controlador correspondiente.

En la Figura 8.10 se observa el procedimiento. Se están enviando velocidades -2 y +2 para observar el comportamiento del controlador.

8.7. Planificando con el robot simulado

Los componentes que se establecieron como necesarios para realizar tareas de planificación con MoveiIt! en el Capítulo 7 eran:

- ☑ Controlador de la base
  - Proporcionado por *rescuer_simulator (plugin base en u*rdf).
- ☑ Controladores capaces de seguir trayectorias para el brazo y la mano.

    - Proporcionados por *rescuer_control*.
- ☑ Un sensor de puntos 3D.
    - Controlador proporcionado por *rescuer_control*.
- ☑ Publicar el estado de las articulaciones del robot.
    - Proporcionado por *rescuer_control*.
- ☑ Publicar el estado del robot y las transformaciones entre las distintas articulaciones del robot.
    - Proporcionado por *rescuer_control*.

Ahora que se cumplen todos los requisitos se pueden realizar tareas de planificación con el simulador.

Se lanza *Gazebo*.

```
roslaunch rescuer_simulator gazebo.launch
```

Se lanza MoveIt! con la configuración de *RESCUER* simulado.

```
roslaunch rescuer_moveit_config rescuer_simulator.launch
```

¡Ahora ya se pueden realizar tareas de planificación con el brazo y la mano!

En la Figura 8.11 se visualiza *RESCUER* conectado a Gazebo *y* en la Figura 8.12 puede observarse la situación del robot tras haber ejecutado una tarea de planificación para el brazo.

Los nodos de ROS resultantes de ejecutar el simulador y el planificador pueden verse en la Figura 8.13.

```
rescuer:
  # Publish all joint states ----------------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  hand_controller:
    type: "effort_controllers/JointTrajectoryController"
    joints:
      - hand_j11_joint
      - hand_j12_joint
      - hand_j13_joint
      - hand_j21_joint
      - hand_j22_joint
      - hand_j23_joint
      - hand_j32_joint
      - hand_j33_joint

    gains: # Required because we're controlling a velocity interface
      hand_j11_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j12_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j13_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j21_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j22_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j23_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j32_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      hand_j33_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}

  arm_controller:
    type: "effort_controllers/JointTrajectoryController"
    joints:
      - arm_1_joint
      - arm_2_joint
      - arm_3_joint
      - arm_4_joint
      - arm_5_joint
      - arm_6_joint

    gains: # Required because we're controlling a velocity interface
      arm_1_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      arm_2_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      arm_3_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      arm_4_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      arm_5_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
      arm_6_joint:  {p: 1000.0, i: 10.0, d: 0.0, i_clamp: 100.0}
```

Figura 8.1.: config/rescuer_control.yaml de rescuer_control

```
<launch>

  <!-- Load joint controller configurations from YAML file to parameter
       server -->
  <rosparam file="$(find rescuer_control)/config/rescuer_control.yaml"
      command="load"/>

  <!-- load the controllers -->
  <node name="controller_spawner" pkg="controller_manager" type="
      spawner" respawn="true" ns="/rescuer" output="screen"
     args="  hand_controller
          arm_controller
          joint_state_controller"/>


  <!-- convert joint states to TF transforms for rviz, etc-->
  <node ns="rescuer" name="robot_state_publisher" pkg="
      robot_state_publisher" type="robot_state_publisher"
     respawn="false" output="screen">
     <param name="tf_prefix" value="/rescuer" />
     <remap from="/joint_states" to="/rescuer/joint_states" />
  </node>

</launch>
```

Figura 8.2.: launch/rescuercontro.launch de rescuer_control

```
<gazebo>
    <plugin filename="libgazebo_ros_control.so" name="
        gazebo_ros_control">
        <robotNamespace>/rescuer</robotNamespace>
        <robotSimType>gazebo_ros_control/DefaultRobotHWSim</
            robotSimType>
    </plugin>
    <plugin filename="libgazebo_ros_joint_pose_trajectory.so" name="
        gazebo_ros_joint_pose_trajectory">
    </plugin>
</gazebo>
```

Figura 8.3.: Gazebo, ros_control y joint_pose_trajectory

```
<gazebo reference="${name}_link">

    <material value="Gazebo/Blue" />
    <turnGravityOff>false</turnGravityOff>
    <sensor type="depth" name="openni_camera_camera">
        <always_on>1</always_on>
        <visualize>true</visualize>
        <camera>
            <horizontal_fov>1.047</horizontal_fov>
            <image>
                <width>640</width>
                <height>480</height>
                <format>R8G8B8</format>
            </image>
            <depth_camera>
            </depth_camera>
            <clip>
                <near>0.1</near>
                <far>100</far>
            </clip>
        </camera>

        <plugin name="${name}_link_controller" filename="
            libgazebo_ros_openni_kinect.so">
            <!--robotNamespace>/rescuer</robotNamespace-->
            <baseline>0.2</baseline>
            <alwaysOn>true</alwaysOn>
            <updateRate>1.0</updateRate>
            <cameraName>rescuer/${name}</cameraName>
            <imageTopicName>depth_registered/image_raw</imageTopicName>
            <cameraInfoTopicName>depth_registered/camera_info</
                cameraInfoTopicName>
            <depthImageTopicName>depth_registered/image_raw</
                depthImageTopicName>
            <depthImageInfoTopicName>depth_registered/camera_info</
                depthImageInfoTopicName>
            <pointCloudTopicName>depth_registered/points</
                pointCloudTopicName>
            <frameName>camera_depth_optical_frame</frameName>
            <pointCloudCutoff>0.5</pointCloudCutoff>
            <distortionK1>0</distortionK1>
            <distortionK2>0</distortionK2>
            <distortionK3>0</distortionK3>
            <distortionT1>0</distortionT1>
            <distortionT2>0</distortionT2>
            <CxPrime>0</CxPrime>
            <Cx>0</Cx>
            <Cy>0</Cy>
            <focalLength>0</focalLength>
            <hackBaseline>0</hackBaseline>
            <depthImageCameraInfoTopicName>depthcamerainfo</
                depthImageCameraInfoTopicName>
        </plugin>
    </sensor>

</gazebo>
```


Figura 8.4.: Gazebo y Kinect

```
<gazebo>
    <plugin name="differential_drive_controller" filename="
        libgazebo_ros_diff_drive.so">
        <!--robotNamespace>/rescuer</robotNamespace-->
        <alwaysOn>true</alwaysOn>
        <updateRate>50</updateRate>
        <leftJoint>left_wheel_joint</leftJoint>
        <rightJoint>right_wheel_joint</rightJoint>
        <wheelSeparation>0.77</wheelSeparation>
        <wheelDiameter>0.26</wheelDiameter>
        <torque>20</torque>
        <commandTopic>cmd_vel</commandTopic>
        <odometryTopic>/rescuer/odom_combined</odometryTopic>
        <odometryFrame>/rescuer/odom_combined</odometryFrame>
        <robotBaseFrame>/rescuer/base_footprint</robotBaseFrame>
        <publishWheelTF>true</publishWheelTF>
        <publishWheelJointState>true</publishWheelJointState>
        <rosDebugLevel>Info</rosDebugLevel>
        <wheelAcceleration>1</wheelAcceleration>
        <wheelTorque>20</wheelTorque>
        <odometrySource>world</odometrySource>
        <publishTf>1</publishTf>
    </plugin>
</gazebo>
```

Figura 8.5.: Gazebo y el controlador diferencial de la base móvil

```
<transmission name="${name}_1_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${name}_1_joint"/>
        <actuator name="${name}_1_motor">
            <hardwareInterface>EffortJointInterface</hardwareInterface>
            <mechanicalReduction>1</mechanicalReduction>
        </actuator>
</transmission>
```

Figura 8.6.: Gazebo y las transmisiones

```
<?xml version="1.0"?>
<launch>
    <param name="/use_sim_time" value="true" />
    <param name="robot_description" command="$(find xacro)/xacro.py $(
        find rescuer_description)/urdf/rescuer.urdf.simulator.xacro" />
    <param name="/rescuer/robot_description" command="$(find xacro)/
        xacro.py $(find rescuer_description)/urdf/rescuer.urdf.
        simulator.xacro" />
    <!--rosparam command="load" ns="/rescuer" file="$(find
        rescuer_simulator)/config/gazebo_ros_control.yaml"/-->
    <include file="$(find gazebo_ros)/launch/empty_world.launch" >
        <!--arg name="world_name" value="$(find rescuer_simulator)/
            worlds/rescuerSampleObject.world"/-->
    </include>

    <!--node name="joint_state_publisher" pkg="joint_state_publisher"
        type="joint_state_publisher"></node-->
    <!--node name="robot_state_publisher" pkg="robot_state_publisher"
        type="robot_state_publisher" output="screen" ></node-->

    <include file="$(find rescuer_control)/launch/rescuercontrol.launch
        " />
    <node name="spawn_robot"  pkg="gazebo_ros" type="spawn_model" args=
        "-urdf -param robot_description -z 0.1 -model robot_model"
        respawn="false" output="screen" />
</launch>
```

Figura 8.7.: rescuer_simulator/launch/gazebo.launch

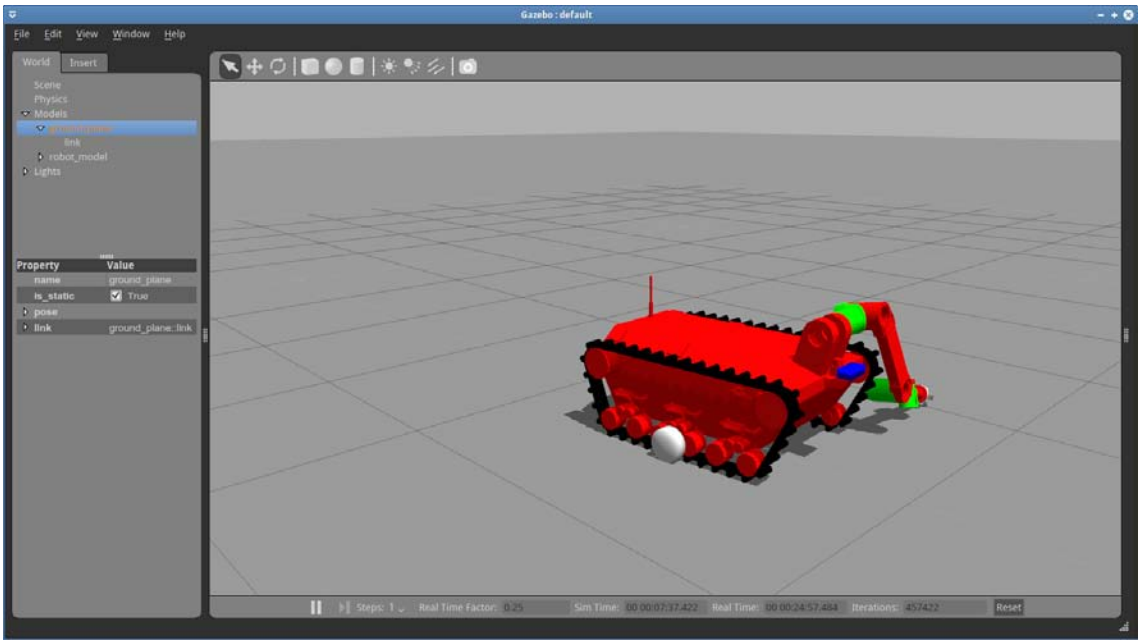


Figura 8.8.: Rescuer en el simulador gazebo

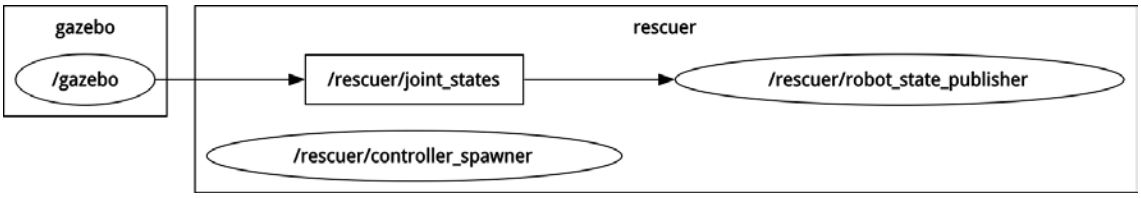


Figura 8.9.: Nodos y tópicos después de arrancar *Gazebo*

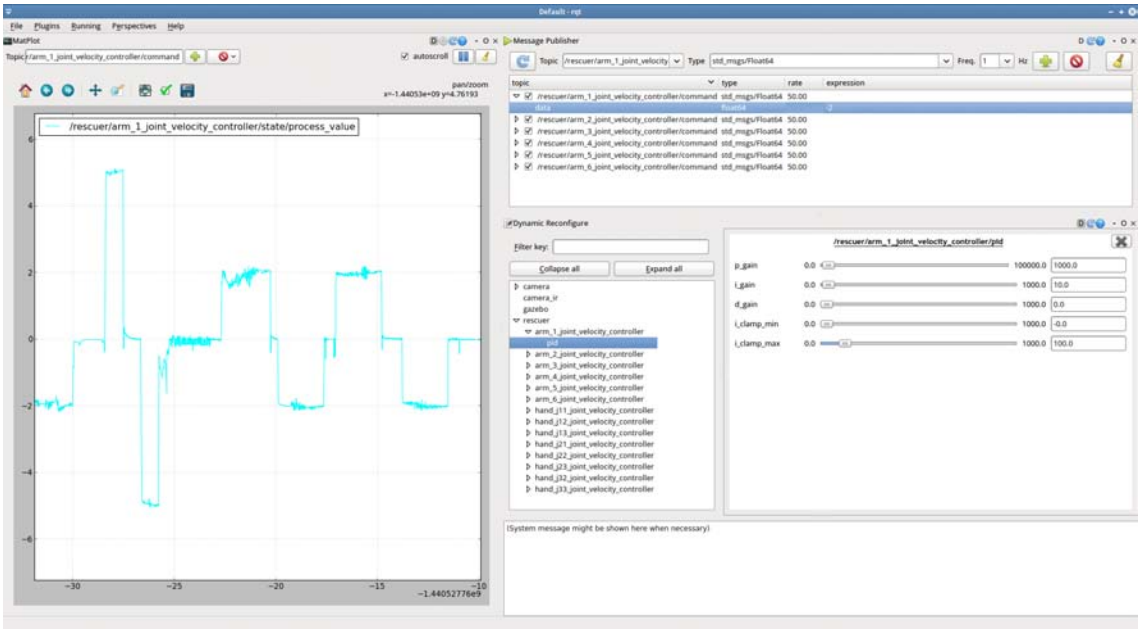


Figura 8.10.: Cálculo pid

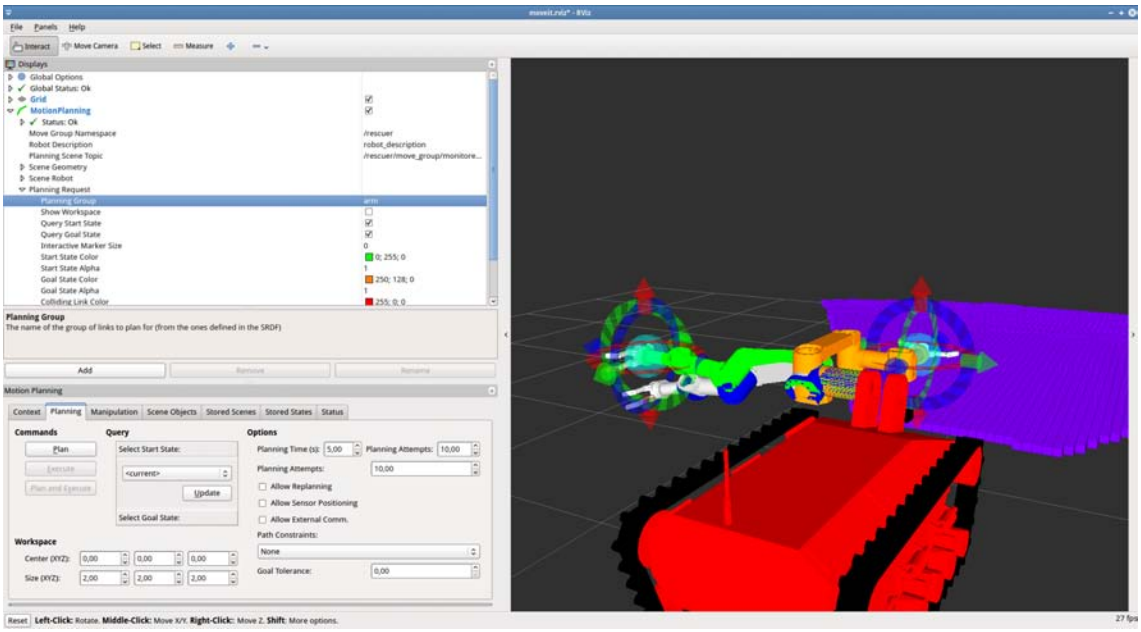


Figura 8.11.: MoveIt! con RESCUER

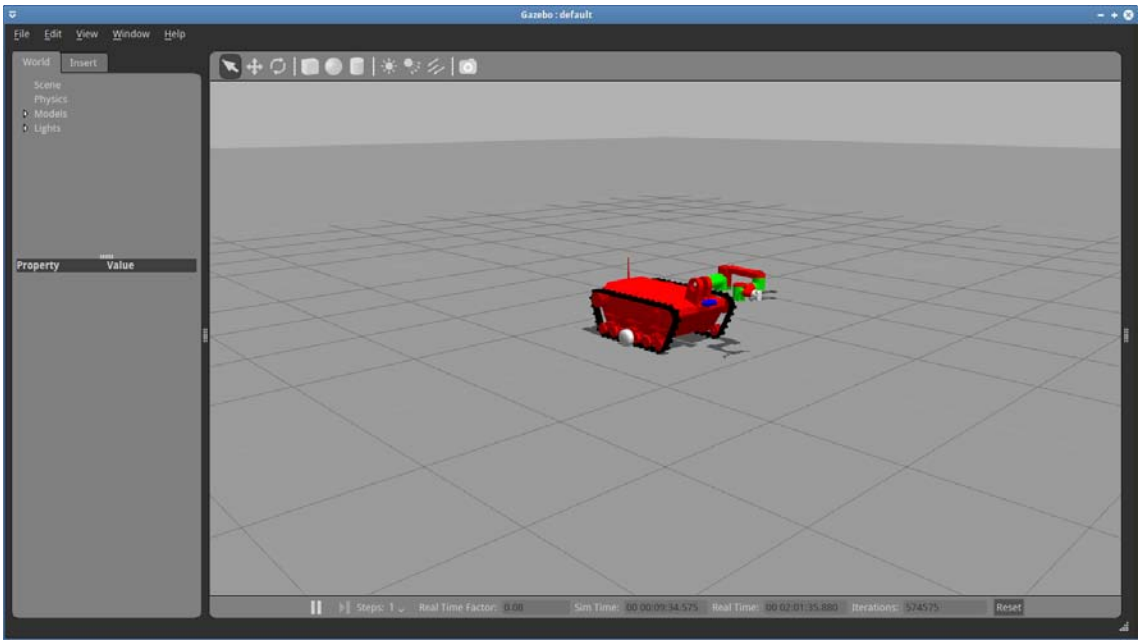


Figura 8.12.: Robot simulado tras ejecutar planificación

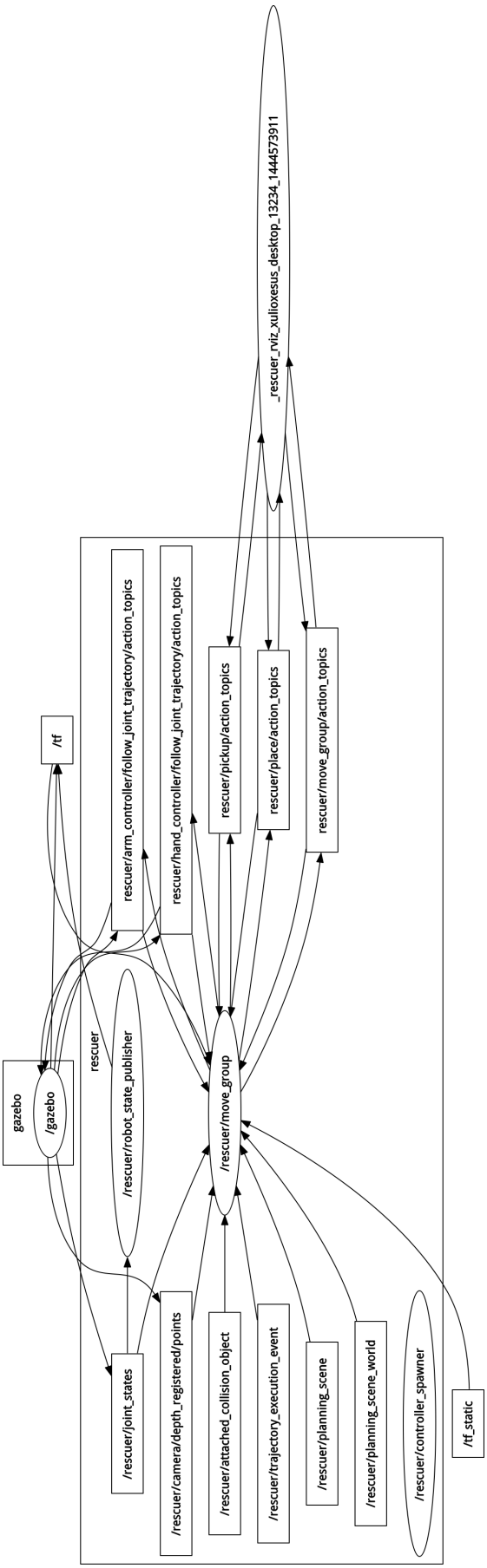


Figura 8.13.: Rosgraph tras lanzar Rescuer y MoveIt!

9. Navegación

9.1. Introducción

La pila de navegación de ROS utiliza la información odométrica y los datos obtenidos de los sensores para enviar comandos de velocidad a una base móvil.

Tal y como se refleja en la documentación de ROS, para poder usar la pila de navegación con un robot, RESCUER debe cumplir una serie de requisitos.

Los esfuerzos realizados para conseguir integrar la navegación con *RESCUER,* se han centrado en el robot simulado. La configuración confeccionada no se ha llegado a probar en el robot real. En el robot simulado ante la dificultad de poder mover la base mediante el controlador lanzado por *rescuer_control,* tampoco se ha conseguido probar la configuración.

9.2. Requisitos

Para utilizar la pila de navegación, con el robot real y el simulado, se deben cumplir los prerrequisitos:

- ☑ El robot tiene que estar utilizando ROS. Ver [Leó15].
- ☑ Poseer un árbol de transformadas tf.
- ☑ Publicar los datos de los sensores utilizando los tipos de mensajes de ROS.
- ☒ Configurar la pila de navegación con la forma y la dinámica del robot a utilizar.

Se seguirá el procedimiento de la documentación de la pila de navegación, ver [ROS15d], para cumplir con los prerrequisitos anteriores.

La pila de navegación a pesar de ser de propósito general tiene tres requisitos hardware que restringen su uso:

- ☑ El robot ha de tener una base diferencial u holonímica con ruedas. Se tienen que poder enviar comandos de velocidad con la forma, velocidad en x, y y theta.
    - RESCUER es un robot con cadenas, la base móvil acepta comandos como si de un modelo diferencial se tratara.

- ☑ Requiere un láser montado en la base que se utiliza para la construcción del mapa y para la localización.
  - Usa kinect que realizará las funciones del laser.
- ☑ Se requiere que el robot sea cuadrado o circular.
  - RESCUER es rectangular si se coloca el brazo plegado encima de la base.

Según el esquema de la pila de navegación (ver Figura 9.1) los componentes en azul son los que debe proveer cada robot. El procedimiento para configurar la pila de navegación en *RESCUER* se puede consultar en [ROS15c].

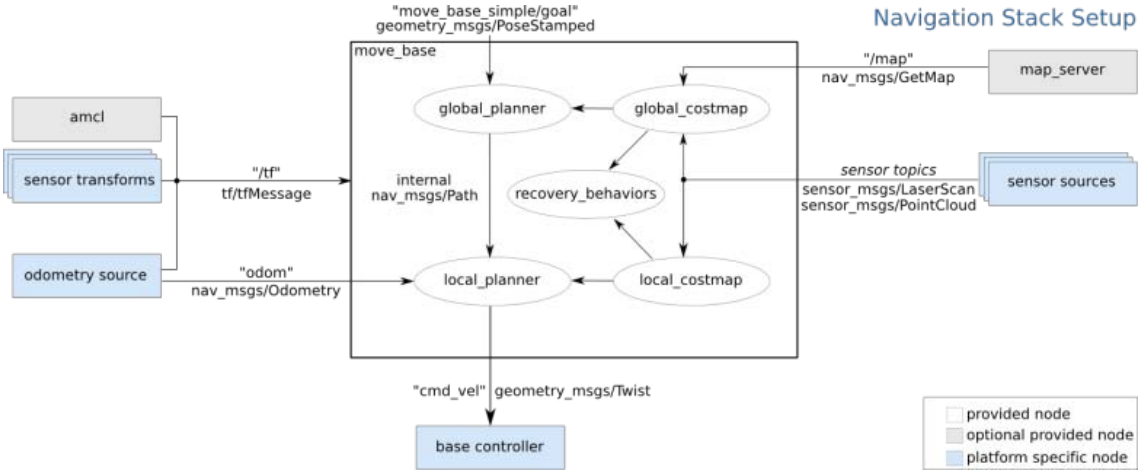


Figura 9.1.: Esquema de la pila de navegación

- sensor sources:
  - Se utiliza PointCloud provisto por Kinect.
- base controller:
  - base_controller proporciona el controlador para poder enviar los comandos de velocidad.
- Información odómetrica y de transformadas necesaria
  - Gazebo en el robot simulado.
  - base_controller y robot_state_publisher en el robot real.

9.3. Adaptación a RESCUER

9.3.1. rescuer_map

Se crea el paquete *rescuer_map.*

En este paquete se guarda una imagen con un mapa para pruebas en *config/map.pgm.*

Este mapa se corresponde con Willow Garage para poder utilizar en el entorno simulado un mapa conocido mientras que con el robot real no se ha llegado a construir ningún mapa.

9.3.2. rescuer_navigation

Se crea el paquete *rescuer_navigation.*

Mover la base

Mediante el lanzador *launch/move_base.launch.*

- Carga un nodo que transforma los datos leídos por la Kinect al formato de lectura láser adecuado para la pila de navegación.
- Lanza un servidor de mapas con los datos de *rescuer_map.*
- Lanza el nodo *amcl.*
- Lana el nodo *move_base* cargando la configuración de *rescuer_navigation/config*

```
<launch>
  <master auto="start"/>

  <!--include file="$(find rescuer_navigation)/launch/
      pointcloud_to_laserscan.launch" /-->
  <include file="$(find rescuer_navigation)/launch/
      depthimage_to_laserscan.launch" />
  <!-- Run the map server -->
  <node name="map_server" pkg="map_server" type="map_server" args="$(
      find rescuer_map)/config/map.pgm 0.005"/>

  <!--- Run AMCL -->

  <include file="$(find rescuer_navigation)/launch/amcl_diff.launch" />

  <node pkg="move_base" type="move_base" respawn="false" name="
      move_base" output="screen">
    <rosparam file="$(find rescuer_navigation)/config/
        costmap_common_params.yaml" command="load" ns="global_costmap"
        />
    <rosparam file="$(find rescuer_navigation)/config/
        costmap_common_params.yaml" command="load" ns="local_costmap" /
        >
    <rosparam file="$(find rescuer_navigation)/config/
        local_costmap_params.yaml" command="load" />
    <rosparam file="$(find rescuer_navigation)/config/
        global_costmap_params.yaml" command="load" />
    <rosparam file="$(find rescuer_navigation)/config/
        base_local_planner_params.yaml" command="load" />
  </node>
</launch>
```

Configuración

La configuración de la pila de navegación se encuentra en *rescuer_navigation/config* en los siguientes ficheros:

base_local_planner_params.yaml

```
TrajectoryPlannerROS:
  max_vel_x: 0.45
  min_vel_x: 0.1
  max_vel_theta: 1.0
  min_in_place_vel_theta: 0.4

  acc_lim_theta: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 2.5

  holonomic_robot: false
```

costmap_common_params.yaml

```
obstacle_range: 2.5
raytrace_range: 3.0
footprint: [[-0.4, -0.6], [-0.4, 0.6], [0.4,0.6], [0.4,-0.6]]
#robot_radius: ir_of_robot
inflation_radius: 0.55

#observation_sources: laser_scan_sensor point_cloud_sensor
observation_sources: point_cloud_sensor

#laser_scan_sensor: {sensor_frame: frame_name, data_type: LaserScan, topic:
    topic_name, marking: true, clearing: true}

point_cloud_sensor: {sensor_frame: rescuer/camera/camera_depth_optical_frame,
    data_type: PointCloud2, topic: /rescuer/camera/depth_registered/points, marking
    : true, clearing: true}
```

global_costmap_params.yaml

```
global_costmap:
  global_frame: map
  robot_base_frame: rescuer/base_footprint
  update_frequency: 5.0
  static_map: true
```

local_costmap_params.yaml

```
local_costmap:
  global_frame: rescuer/odom_combined
  robot_base_frame: rescuer/base_footprint
  update_frequency: 5.0
  publish_frequency: 2.0
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.05
```

9.4. Conclusiones.

La configuración de la pila de navegación queda preparada hasta poder hacer funcionar la base en el simulador. Una vez probado en el simulador se podría pasar al robot real, empezando por la construcción de un mapa de laboratorio o de los pasillos circundantes.

La forma de lanzar la pila de navegación sería:

```
roslaunch rescuer_navigation move_base.launch
```

10. Problemas

Algunos de los problemas encontrados a lo largo del proyecto se detallan a continuación.

10.1. Articulaciones del brazo desfasadas

A veces sucede que con el brazo en posición cero, la información de las articulaciones es incorrecta. Sin encontrar la causa del problema, la forma de solucionarlo es mediante el programa PowerCube.

- Utilizar el software PowerCube desde Windows XP, (ver anexos de [Leó15])
- Enviar corriente cero al motor desfasado.
- Girar mecánicamente el motor hasta la marca cero.
- Reset del módulo.
- Marcar la posición cero como la actual.
- Guardar los cambios en la EEPROM.

10.2. Error al intentar visualizar la descripción del robot. Ficheros STL mal formados

Al intentar visualizar en rviz la descripción del robot se observa un error que dice que los ficheros stl están mal formados.

```
roslaunch rescuer_description display.launch
```

El mensaje de error se puede ver en la Figura 10.1.

Esto se debe a un bug de rviz documentado en [Rvi15], donde también se pude ver como solucionarlo.

```
[ERROR] [1447149561.795927655]: The STL file 'package://rescuer_description/meshes/
    modular/modular1.stl' is malformed. According to the binary STL header it
    should have '15878' triangles, but it has too much data for that to be the case
    .
[ERROR] [1447149561.797005166]: Failed to load file [package://rescuer_description/
    meshes/modular/modular1.stl]
[ERROR] [1447149561.852990494]: Could not load model 'package://rescuer_description
    /meshes/modular/modular1.stl' for link 'arm_1_link': OGRE EXCEPTION(6:
    FileNotFoundException): Cannot locate resource package://rescuer_description/
    meshes/modular/modular1.stl in resource group Autodetect or any other group. in
     ResourceGroupManager::openResource at /build/buildd/ogre-1.7.4/OgreMain/src/
    OgreResourceGroupManager.cpp (line 753)
[ERROR] [1447149561.853438875]: Could not load model 'package://rescuer_description
    /meshes/modular/modular1.stl' for link 'arm_1_link': OGRE EXCEPTION(6:
    FileNotFoundException): Cannot locate resource package://rescuer_description/
    meshes/modular/modular1.stl in resource group Autodetect or any other group. in
     ResourceGroupManager::openResource at /build/buildd/ogre-1.7.4/OgreMain/src/
    OgreResourceGroupManager.cpp (line 753)
[ERROR] [1447149561.853712094]: Could not load model 'package://rescuer_description
    /meshes/modular/modular1.stl' for link 'arm_1_link': OGRE EXCEPTION(6:
    FileNotFoundException): Cannot locate resource package://rescuer_description/
    meshes/modular/modular1.stl in resource group Autodetect or any other group. in
     ResourceGroupManager::openResource at /build/buildd/ogre-1.7.4/OgreMain/src/
    OgreResourceGroupManager.cpp (line 753)
[ERROR] [1447149561.853982240]: Could not load model 'package://rescuer_description
    /meshes/modular/modular1.stl' for link 'arm_1_link': OGRE EXCEPTION(6:
    FileNotFoundException): Cannot locate resource package://rescuer_description/
    meshes/modular/modular1.stl in resource group Autodetect or any other group. in
     ResourceGroupManager::openResource at /build/buildd/ogre-1.7.4/OgreMain/src/
    OgreResourceGroupManager.cpp (line 753)
```

Figura 10.1.: Error ficheros stl

10.2.1. Primera solución

Básicamente se trata de versionar rviz par parchear el bug.

```
sudo apt-get install ros-hydro-cmake-modules
git clone https://github.com/ros-visualization/rviz.git -b
   hydro-devel
mkdir -p rviz/build
cd rviz/build
source /opt/ros/hydro/setup.bash
cmake .. -DCMAKE_INSTALL_PREFIX=./install
make install
source ~/rviz/build/install/setup.bash
catkin_make # en el espacio de trabajo de catkin
source ~/rescuer_catkin/devel/setup.bash
roslaunch rescuer_description display.launch
```

10.2.2. Segunda solución

Se cambian los ficheros stl por dae utilizando meshlab y se modifican las descripciones URDF de RESCUER para utilizar los nuevos ficheros dae. Esta es la solución adoptada con la desventaja de que se pierden los colores en la representación gráfica de RESCUER, ver la Figura 10.2.

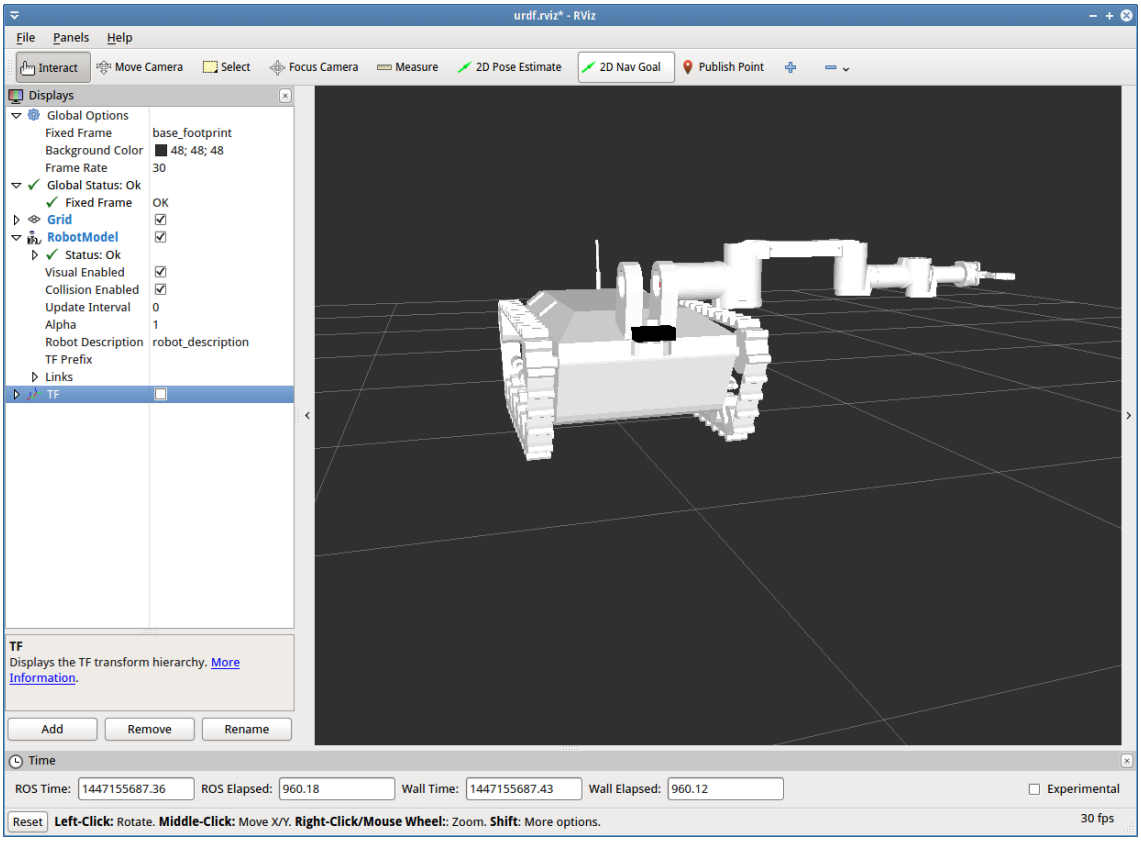


Figura 10.2.: Visualizando RESCUER con ficheros dae

10.3. RVIZ problema de visualización

Si no se visualiza RVIZ en el sistema, se puede solucionar estableciendo la variable de entorno adecuada.

```
export LIBGL_ALWAYS_SOFTWARE=1
```

10.4. Transmisiones de la mano simulada

Si no se definen las transmisiones correctamente en la descripción urdf del robot, se puede recibir un mensaje como el siguiente.

```
[ERROR] [1439979211.494525336, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j32_joint
[ERROR] [1439979211.498489526, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j33_joint
[ERROR] [1439979211.503478987, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j11_joint
[ERROR] [1439979211.508803128, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j12_joint
[ERROR] [1439979211.513200614, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j13_joint
[ERROR] [1439979211.517155307, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j21_joint
[ERROR] [1439979211.528619892, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j22_joint
[ERROR] [1439979211.533392007, 4.939000000]: No p gain specified for pid.
    Namespace: /gazebo_ros_control/pid_gains/hand_j23_joint
```

Solucionado cambiando en el urdf:

```
<transmission name="${name}_j32_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${name}_j32_joint"/>
    <actuator name="${name}_j32">
        <hardwareInterface>PositionJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
        <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
</transmission>
```

por:

```
<transmission name="${name}_j32_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${name}_j32_joint"/>
    <actuator name="${name}_j32">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
        <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
</transmission>
```

para todos los joints del manipulador.

Fichero: rescuer_description/urdf/barrett/bh282.urdf.xacro

10.5. Fallo de configuración local en rescuer.srdf

El asistente de *MoveIt!* genera el fichero *rescuer_moveit_config/config/rescuer.srdf* basándose en la configuración del idioma del sistema. Debido a un bug cuando se arrranca *RVIZ* con el plugin de planificación, esa configuración puede dar error al detectar que las poses contienen números con coma en lugar de punto.

La solución consiste en editar el fichero y sustituir las comas por puntos.

10.6. La plataforma móvil no se mueve en el simulador

Ha resultado imposible lograr mover la base móvil en el simulador. Se cree que la razón se encuentra en la descripción *URDF* del robot, en lo que se refiere a la masas de la base móvil y de las ruedas así como de las matrices de inercia de las mismas.

Una posible solución (de mayor coste) sería eliminar completamente las ruedas e intentar diseñar para *Gazebo* el plugin con las cadenas reales, ver [ROS15e].

11. Guía rápida

11.1. Recomendaciones

- No mover el brazo con la batería cargada por encima de 25'5 V, se pueden llegar a fundir los fusibles.
- Operar siempre cerca de la seta de emergencia.
- Devolver el brazo a la posición cero antes de apagar el robot.

11.2. Visualización

Visualizar RESCUER

```
roslaunch rescuer_description display.launch
```

Visualizar RESCUER simulado

```
roslaunch rescuer_description display_simulator.launch
```

11.3. Robot real

Puesta en marcha

```
roslaunch rescuer_bringup rescuer.launch
```

o

```
rescuer-startup.sh
```

Poner el brazo en posición cero

```
roslaunch rescuer_bringup arm_to_zero.launch
```

MoveIt!

Para realizar tareas de planificación desde MoveIt! ejecutar (con *RESCUER* simulado ya en marcha):

```
roslaunch rescuer_moveit_config rescuer.launch
```

11.4. Gazebo

Para trabajar con el robot simulado, se debe ejecutar siempre en primer lugar:

```
roslaunch rescuer_simulator gazebo.launch
```

MoveIt!

Para realizar tareas de planificación desde MoveIt! ejecutar (con *RESCUER* simulado ya en marcha):

```
roslaunch rescuer_moveit_config rescuer_simulator.launch
```

12. Conclusiones

En lo que se refiere a la integración de RESCUER con la plataforma *MoveIt!* se considera que se ha llegado a cumplir los objetivos pretendidos, puesto que se pueden realizar tareas de planificación con el brazo. Sin embargo la imposibilidad de disponer de una mano de Barret operativa, ha provocado que no se pueda considerar el trabajo finalizado a este respecto.

En cuanto al robot simulado, se considera que queda bastante trabajo por realizar, puesto que el comportamiento de los controladores del brazo y de la base móvil provocan que el robot se mueva de forma errática en *Gazebo.*

La navegación se considera el punto débil de este proyecto. Aunque se ha construido la arquitectura necesaria para integrar el robot con la pila de navegación de ROS, no se ha podido experimentar con su funcionalidad. El problema radica en el controlador de la base móvil del simulador, que no consigue mover la base. Tampoco se ha llegado a probar la arquitectura de navegación en el robot real, porque se ha invertido demasiado tiempo en intentar hacer funcionar en el simulador.

Pese a todos los inconvenientes, el entorno de trabajo creado para *RESCUER* se considera exitoso. Se ha dotado de una infraestructura que sigue los estándares de ROS, con lo que el trabajo se podría continuar de una forma más sencilla que en los inicios de este proyecto.

El número de horas planificadas se ha excedido, debido fundamentalmente a que los objetivos del proyecto quizá hayan sido demasiado optimistas. Se ha requerido mucho tiempo para describir el robot adecuadamente y algunas de las piezas de software creadas también han entrañado cierta dificultad.

Se valora positivamente la creación de esta propia documentación, para servir como guía a los próximos investigadores *RESCUER,* o para aquellos que quieran integrar su robot con ROS, partiendo básicamente de cero.

Personalmente la posibilidad de trabajar con un robot real ha resultado una absoluta satisfacción.

13. Trabajo futuro

Descripción	Paquete	Fichero	Prioridad
Proporcionar color en la representación de los ficheros dae para una mejor visualización	rescuer_description	meshes	Leve
Eliminar dependencias por no tener ejecutables	rescuer_description	package.xml	Leve
Utilizar dae para describir la kinect	rescuer_description	urdf	Leve
Eliminar ficheros marcados con unused	rescuer_description	launch	Leve
Eliminar ficheros marcados con unused	rescuer_description	meshes	Leve
Eliminar rescuer_odom	rescuer_odom		Leve
Eliminar rescuer_moveit_config_wrong	rescuer_moveit_config_wrong		Leve
Eliminar rescuer_moveit_config_wheels	rescuer_moveit_config_wheels		Leve
Invisibilidad ruedas simulador	rescuer_description		Leve
Eliminar odometry_publisher	rescuer_bringup		Leve
Sustituir rescuer_bringup por ros_control	rescuer_bringup		Leve
Añadir dependencias ros_control ros_controllers	rescuer_control	CMakeLists.txt	Leve
Añadir dependencias ros_control ros_controllers	rescuer_control	package.xml	Leve
Revisar dependencias	Todos		Leve
Unificar lanzadores MoveIt!	rescuer_moveit_config	rescuer*.launch	Media
Dotar a rescuer de odometría (GPS)	rescuer_odom		Media
Utilizar IKFast en MoveIt! (ver [ROS15a])	rescuer_moveit_config		Media
Eliminar alimentación externa Kinect	hardware		Media
Mover base en el simulador	rescuer_simulator		Alta
Instalar mano de Barret	hardware		Alta
Crear controlador mano de Barret	rescuer_controllers		Alta
Crear controlador de trayectoria Barret	rescuer_controllers		Alta
Calibración de Kinect	rescuer_controllers		Alta
Instalación de láser	hardware		Alta
Crear controlador para laser	rescuer_controllers		Alta
Revisión medidas meshes	rescuer_description		Muy Alta

Cuadro 13.1.: Trabajo futuro

Agradecimientos

A mi familia y amigos. A Marta por soportar tantas horas de trabajo.

A mi tutor Enric por permitirme tener la experiencia de realizar este trabajo.

A. rescuer_controllers

La estructura del paquete *rescuer_controllers* puede verse en la Figura A.1. El fichero *CMakeLists.txt* (Figura A.2) se usa para la construcción del paquete. En él se puede ver cómo se construyen las librerías y ejecutables a partir del código fuente existente en la antigua instalación de RESCUER. Por otra parte, en *package.xml* (Figura A.3) se señalan las dependencias del paquete con otros paquetes de ROS, tanto en tiempo de compilación como en tiempo de ejecución.

Los ficheros terminados en ".unset" se utilizan para desactivar temporalmente el paquete y no compilarlo con *catkin_make*. La técnica para desactivar consiste en borrar *CMakesLists.txt* y *package.xml*, de manera que el paquete *rescuer_controllers* sea "invisible" para *catkin_make*. Los archivos con el sufijo ".unset" sirven de copia para restaurar el paquete.

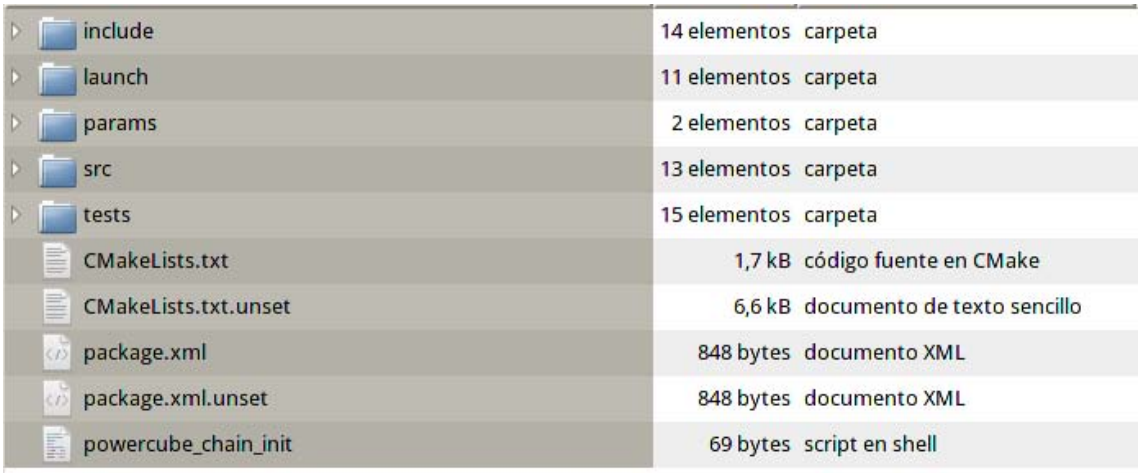


Figura A.1.: Estructura del paquete rescuer_controllers

Directorio src

En la Figura A.4 se puede observar el contenido del directorio *src* del paquete.

Directorio include

Como se observa en la Figura A.5, en este directorio se encuentran las cabeceras de código fuente de los fichero del directorio *src* del paquete.

```
cmake_minimum_required(VERSION 2.8.3)
project(rescuer_controllers)

find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs genmsg)
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  message_generation
  tf
  actionlib
  actionlib_msgs
)

catkin_package()

include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(base_controller src/base_controller.cpp)

add_library(logLib src/logLib.cc)
add_library(Component src/Component.cc)
add_library(dx60co8c src/dx60co8c.c)
add_library(dx60co8_esd src/dx60co8_esd.cc)
add_library(MotorDrive src/MotorDrive.cc)
add_library(PrimitiveDriver src/PrimitiveDriver.cc)

target_link_libraries(dx60co8_esd ntcan)
target_link_libraries(PrimitiveDriver logLib MotorDrive dx60co8_esd
    Component)
target_link_libraries(base_controller ${catkin_LIBRARIES}
    PrimitiveDriver)
```

Figura A.2.: CMakesLists.txt de rescuer_controllers

Directorio launch

En este directorio se sitúan los archivos que sirven para poner en marcha nodos de ROS, cargar parámetros, etc.

Directorio params

En el directorio *params* del paquete se encuentra la configuración del driver que controla el brazo visto con anterioridad

```
<?xml version="1.0"?>
<package>
  <name>rescuer_controllers</name>
  <version>0.0.0</version>
  <description>The rescuer_controllers package</description>

  <maintainer email="xulioxesus@gmail.com">xulioxesus</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>actionlib</build_depend>
  <build_depend>actionlib_msgs</build_depend>
  <build_depend>message_generation</build_depend>
  <build_depend>tf</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>actionlib</run_depend>
  <run_depend>actionlib_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
  <run_depend>tf</run_depend>

</package>
```

Figura A.3.: package.xml de rescuer_controllers

Directorio tests

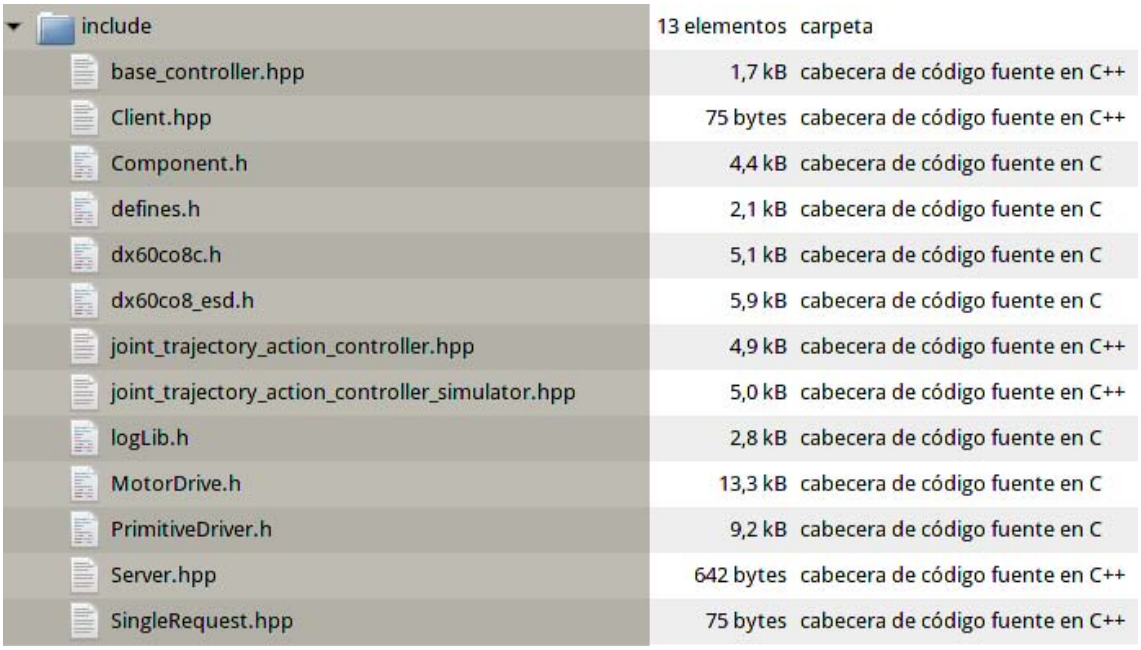
El directorio tests no sufre cambios respecto al proyecto [Leó15].

base_controller.cpp

```
#include <base_controller.hpp>

using namespace std;


PrimitiveDriver base_controller::driver = *(new PrimitiveDriver(
    DEFAULT_RESCUER_PORT));
double base_controller::x = 0.0;
double base_controller::y = 0.0;
double base_controller::theta = 0.0;
double base_controller::vx = 0.0;
double base_controller::vth = 0.0;
ros::Time base_controller::last_time;
geometry_msgs::TransformStamped base_controller::odom_trans;
nav_msgs::Odometry base_controller::odom;
```

Figura A.4.: Directorio src de rescuer_controllers

```
//=============================================================================
// Constructor and Destructor =================================================
//=============================================================================

base_controller::base_controller()
{
}

base_controller::~base_controller()
{
}

//=============================================================================
// Setters and getters ========================================================
//=============================================================================

void base_controller::SetX(double pos)
{
    base_controller::x = pos;
}

double base_controller::GetX()
{
    return base_controller::x;
}

void base_controller::SetY(double pos)
{
    base_controller::y = pos;
}

double base_controller::GetY()
{
    return base_controller::y;
}
```



include	13 elementos	carpeta
base_controller.hpp	1,7 kB	cabecera de código fuente en C++
Client.hpp	75 bytes	cabecera de código fuente en C++
Component.h	4,4 kB	cabecera de código fuente en C
defines.h	2,1 kB	cabecera de código fuente en C
dx60co8c.h	5,1 kB	cabecera de código fuente en C
dx60co8_esd.h	5,9 kB	cabecera de código fuente en C
joint_trajectory_action_controller.hpp	4,9 kB	cabecera de código fuente en C++
joint_trajectory_action_controller_simulator.hpp	5,0 kB	cabecera de código fuente en C++
logLib.h	2,8 kB	cabecera de código fuente en C
MotorDrive.h	13,3 kB	cabecera de código fuente en C
PrimitiveDriver.h	9,2 kB	cabecera de código fuente en C
Server.hpp	642 bytes	cabecera de código fuente en C++
SingleRequest.hpp	75 bytes	cabecera de código fuente en C++

Figura A.5.: Directorio include de rescuer_controllers

```
void base_controller::SetTheta(double pos)
{
    base_controller::theta = pos;
}

double base_controller::GetTheta()
{
    return base_controller::theta;
}


void base_controller::SetXVelocity(double vel)
{
    base_controller::vx = vel;
}

double base_controller::GetXVelocity()
{
    return base_controller::vx;
}


void base_controller::SetThetaVelocity(double vel)
{
    base_controller::vth = vel;
}

double base_controller::GetThetaVelocity()
{
    return base_controller::vth;
}

//=============================================================================
// Callback to /cmd_vel =======================================================
```

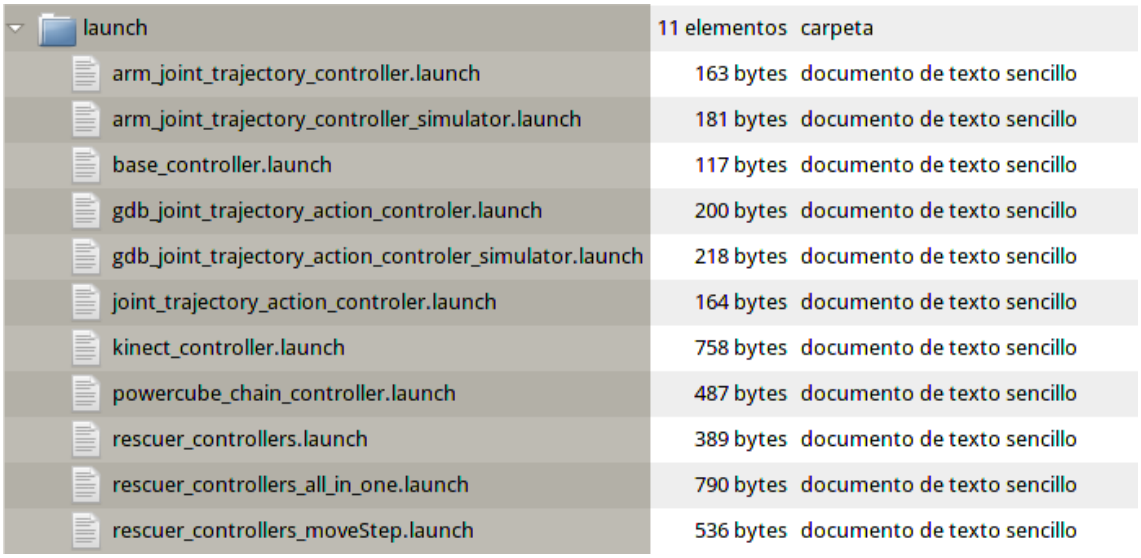


Figura A.6.: Directorio launch de rescuer_controllers

Figura A.7.: params de rescuer_controllers

```
//=============================================================================

void base_controller::cmd_velCallback(const geometry_msgs::Twist &twist_original)
{
    geometry_msgs::Twist twist = twist_original;
    double velocity_x = twist_original.linear.x;
    double velocity_th = twist_original.angular.z;

    ROS_INFO("request: linear=%f, angular=%f", velocity_x, velocity_th);

    base_controller::SetXVelocity(velocity_x);
    base_controller::SetThetaVelocity(velocity_th);

        float propulsiveLinearEffortXPercent=0.0,
            propulsiveRotationalEffortZPercent=0.0;

        //Converts velocity values into proportional percentage
        //propulsiveLinearEffortXPercent = velocity_x*100;
        //propulsiveRotationalEffortZPercent = velocity_th*100;
        propulsiveLinearEffortXPercent = velocity_x*LINEAR_SPEED_TO_EFFORT_FACTOR
            *100;
        propulsiveRotationalEffortZPercent = velocity_th*
            ANGULAR_SPEED_TO_EFFORT_FACTOR*100;

        /*if(propulsiveLinearEffortXPercent > 100.0)
                propulsiveLinearEffortXPercent = 100.0;
        else if(propulsiveLinearEffortXPercent < -100.0)
                propulsiveLinearEffortXPercent = -100.0;
```

```
        if(propulsiveRotationalEffortZPercent > 100.0)
                propulsiveRotationalEffortZPercent = 100.0;
        else if(propulsiveRotationalEffortZPercent < -100.0)
                propulsiveRotationalEffortZPercent = -100.0;
    */


        int result = driver.Move(propulsiveLinearEffortXPercent,
            propulsiveRotationalEffortZPercent);

        ROS_INFO("sending back response: [%ld]", (long int) result);

        //return true;
}

//=============================================================
// Calculate the odometry and publishing onto /odom topic ======
//=============================================================

void base_controller::CalculateOdometry()
{

    ros::Time current_time = ros::Time::now();
    double dt = (current_time - base_controller::last_time).toSec();;

    double vx = base_controller::GetXVelocity();
    double vy = 0;
    //double vth = base_controller::GetThetaVelocity();
    double vth = - base_controller::GetThetaVelocity();
    double th = base_controller::GetTheta();

    double dx = (vx * cos(th) - vy * sin(th)) * dt;
    double dy = (vx * sin(th) + vy * cos(th)) * dt;
    double dth = vth * dt;


    base_controller::SetX(base_controller::GetX()+dx);
    base_controller::SetY(base_controller::GetY()+dy);
    base_controller::SetTheta(base_controller::GetTheta()+dth);

    geometry_msgs::Quaternion odom_quat;
    odom_quat = tf::createQuaternionMsgFromRollPitchYaw(0,0,base_controller::
        GetTheta());

    //base_controller::odom_trans.header.frame_id = "odom";
    base_controller::odom_trans.header.frame_id = "odom_combined";
    //base_controller::odom_trans.child_frame_id = "base_link";
    base_controller::odom_trans.child_frame_id = "base_footprint";
    base_controller::odom_trans.header.stamp = current_time;
    base_controller::odom_trans.transform.translation.x = base_controller::GetX();
    base_controller::odom_trans.transform.translation.y = 0.0;
    base_controller::odom_trans.transform.translation.z = 0.0;
    base_controller::odom_trans.transform.rotation = tf::createQuaternionMsgFromYaw
        (base_controller::GetTheta());

    base_controller::odom.header.stamp = current_time;
    base_controller::odom.header.frame_id = "odom";
    base_controller::odom.child_frame_id = "base_link";

    // position
    base_controller::odom.pose.pose.position.x = base_controller::GetX();
    base_controller::odom.pose.pose.position.y = base_controller::GetY();
    base_controller::odom.pose.pose.position.z = 0.0;
    base_controller::odom.pose.pose.orientation = odom_quat;
    // velocity
    base_controller::odom.twist.twist.linear.x = vx;
```

```
    base_controller::odom.twist.twist.linear.y = vy;
    base_controller::odom.twist.twist.linear.z = 0.0;
    base_controller::odom.twist.twist.angular.x = 0.0;
    base_controller::odom.twist.twist.angular.y = 0.0;
    base_controller::odom.twist.twist.angular.z = vth;

    last_time = current_time;

}


void base_controller::StartUp()
{
        driver.SetMaxSpeed(MOTOR_DEF_MAX_SPEED,MOTOR_DEF_MAX_TURNSPEED);

        if(driver.StartUp() == ERROR)
        {
                exit(-1);
        }
        else
        {
                ROS_INFO("Starting rover rescuer motors...");
        }
}

void base_controller::Sleep(unsigned int mseconds)
{
    clock_t goal = mseconds + clock();
    while (goal > clock());
}

int main(int argc, char** argv)
{
    ros::init(argc, argv, "base_controller");
    ros::NodeHandle node;
    base_controller::StartUp();
    ros::Subscriber cmd_vel_subscriber = node.subscribe("cmd_vel", 50,
        base_controller::cmd_velCallback);
    ros::Publisher odom_pub = node.advertise<nav_msgs::Odometry>("odom", 50);
    tf::TransformBroadcaster broadcaster;
    ros::Rate loop_rate(50);

    while(ros::ok())
    {

        ros::spinOnce();

        base_controller::CalculateOdometry();

        broadcaster.sendTransform(base_controller::odom_trans);
        odom_pub.publish(base_controller::odom);

        loop_rate.sleep();
    }
}
```

joint_trajectory_action_controller.cpp

```
/*
 * Copyright (c) 2008, Willow Garage, Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
```

```
 *     * Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *     * Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *     * Neither the name of the Willow Garage, Inc. nor tnhe names of its
 *       contributors may be used to endorse or promote products derived from
 *       this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

/*
 * Author: Stuart Glaser
 * Modified by: Julio Jesús León Pérez
 */

#include "joint_trajectory_action_controller.hpp"

//PLUGINLIB_DECLARE_CLASS(JointTrajectoryActionController, controller::
    JointTrajectoryActionController, controller::MultiJointController)

static const double ACCEPTABLE_BOUND = 0.05; // amount two positions can vary
    without being considered different positions.

/**
*   Constructor:
*   - the server must not be terminated
*   - initialize the number of joints
*   - initialize the joint names
*/
JointTrajectoryActionController::JointTrajectoryActionController()
{
    terminate_ = false;
    num_joints_ = 6;
    joint_names_.push_back("arm_1_joint");
    joint_names_.push_back("arm_2_joint");
    joint_names_.push_back("arm_3_joint");
    joint_names_.push_back("arm_4_joint");
    joint_names_.push_back("arm_5_joint");
    joint_names_.push_back("arm_6_joint");

    setIsAction(false);
}

/**
*   Destructor:
*   Empty by default.
*/
JointTrajectoryActionController::~JointTrajectoryActionController()
{
}

/**
*   Getters and Setters:
*/
```

```
void JointTrajectoryActionController::setTrajectoryUpdateRate(int value)
{
    trajectory_update_rate_ = value;
}

int JointTrajectoryActionController::getTrajectoryUpdateRate()
{
    return trajectory_update_rate_;
}

void JointTrajectoryActionController::setStateUpdateRate(int value)
{
    state_update_rate_ = value;
}

int JointTrajectoryActionController::getStateUpdateRate()
{
    return state_update_rate_;
}

void JointTrajectoryActionController::setIsAction(bool value)
{
    is_action = value;
}

bool JointTrajectoryActionController::isAction()
{
    return is_action;
}

/**
*   Initialize:
*   - Update rate to processTrajectory set to 1000
*   - Update rate to updateState set to 50
*   - Fill node parameters
*   - Resize feedback_msg_
*/
bool JointTrajectoryActionController::initialize(std::string prefix)
{
    setTrajectoryUpdateRate(70);
    setStateUpdateRate(50);

    node_.param<double>(prefix + "goal_time", goal_time_constraint_, 0.0);
    node_.param<double>(prefix + "stopped_velocity_tolerance",
        stopped_velocity_tolerance_, 0.01);
    node_.param<double>("joint_trajectory_action_node/min_velocity", min_velocity_,
         0.1);
    node_.param<double>("joint_trajectory_action_node/max_velocity", max_velocity_,
         0.5);

    goal_constraints_.resize(num_joints_);
    trajectory_constraints_.resize(num_joints_);

    for (size_t i = 0; i < num_joints_; ++i)
    {
        node_.param<double>(prefix + joint_names_[i] + "/goal", goal_constraints_[i
            ], -1.0);
        node_.param<double>(prefix + joint_names_[i] + "/trajectory",
            trajectory_constraints_[i], -1.0);
    }

    // Setup/resize feedback message
    feedback_msg_.joint_names = joint_names_;
    feedback_msg_.desired.positions.resize(num_joints_);
    feedback_msg_.desired.velocities.resize(num_joints_);
    feedback_msg_.desired.accelerations.resize(num_joints_);
```

```
    feedback_msg_.actual.positions.resize(num_joints_);
    feedback_msg_.actual.velocities.resize(num_joints_);
    feedback_msg_.actual.accelerations.resize(num_joints_);
    feedback_msg_.error.positions.resize(num_joints_);
    feedback_msg_.error.velocities.resize(num_joints_);
    feedback_msg_.error.accelerations.resize(num_joints_);


    return true;
}

/**
*   - ros::Subscriber command_sub_ is subscribed to command topic. Requests are
    processed by JointTrajectoryActionController::processCommand.
*   - ros::Subscriber joint_state_sub_ is subscribed to joint_states topic.
    Requests are processed by JointTrajectoryActionController::getJointStates.
*
*   - ros::Publisher state_pub___ publishes onto state topic. Topic published is
    control_msgs::FollowJointTrajectoryFeedback type.
*
*   - action_server_ pointer is reset.
*       + See: typedef actionlib::SimpleActionServer<control_msgs::
    FollowJointTrajectoryAction> FJTAS;
*       + New FJTAS (simple action server) is created:
*           - SimpleActionServer (ros::NodeHandle n, std::string name,
    ExecuteCallback execute_cb, bool auto_start), the callback is processed by
    JointTrajectoryActionController::processFollowTrajectory.
*   - sction_server_ (simple action server) is started
*
*   - Se crea un thread para JointTrajectoryActionController::updateState
*/

void JointTrajectoryActionController::start()
{

    JointTrajectoryActionController::initialize("joint_trajectory_action_node/
        constraints/");

    command_sub_ = node_.subscribe("command", 50, &JointTrajectoryActionController
        ::processCommand, this);
    joint_state_sub_ = node_.subscribe("/joint_states", 100, &
        JointTrajectoryActionController::getJointStates, this);
    state_pub_ = node_.advertise<control_msgs::FollowJointTrajectoryFeedback>("
        state", 50);
    velocity_pub_ = node_.advertise<brics_actuator::JointVelocities>("/
        powercube_chain/command_vel", 70);

    action_server_.reset(new FJTAS(node_, "follow_joint_trajectory",
                                   boost::bind(&JointTrajectoryActionController::
                                       processFollowTrajectory, this, _1),
                                   false));
    action_server_->start();
    feedback_thread_ = new boost::thread(boost::bind(&
        JointTrajectoryActionController::updateState, this));

    //JointTrajectoryActionController::initState();
}

/**
*   Stops the controller
*/

void JointTrajectoryActionController::stop()
{
    {
        boost::mutex::scoped_lock terminate_lock(terminate_mutex_);
```

```
        terminate_ = true;
    }

    feedback_thread_->join();
    delete feedback_thread_;

    command_sub_.shutdown();
    state_pub_.shutdown();
    velocity_pub_.shutdown();
    joint_state_sub_.shutdown();
    action_server_->shutdown();
}

/**
*   Process command request:
*       - Wait to server availability to process trajectory commands
*       - Call processTrajectory
*/

void JointTrajectoryActionController::processCommand(const trajectory_msgs::
    JointTrajectoryConstPtr& msg)
{
    if (action_server_->isActive())
    {
        action_server_->setPreempted();
    }

    while (action_server_->isActive())
    {
        ros::Duration(0.01).sleep();
    }

    setIsAction(false);
    processTrajectory(*msg);
}

//=============================================================================
// Process Follow Trajectory                                          =
// This is what MoveIt is sending out                                 =
//=============================================================================


void JointTrajectoryActionController::processFollowTrajectory(const control_msgs::
    FollowJointTrajectoryGoalConstPtr& goal)
{
    setIsAction(true);
    processTrajectory(goal->trajectory);
}

/**
*   Init State - TODO: not used
*/

void JointTrajectoryActionController::initState()
{

    for (size_t j = 0; j < joint_names_.size(); ++j)
        {
            joint_states_.position[j] = 0;
            joint_states_.velocity[j] = 0;
        }

}

/**
*   Update state.
```

```
*       - Fills the feedback_msg_
*       - Publish the feedback_msg_
*/

void JointTrajectoryActionController::updateState()
{
    ros::Rate rate(getStateUpdateRate());

    while (node_.ok())
    {
        {
            boost::mutex::scoped_lock terminate_lock(terminate_mutex_);
            if (terminate_) {
                break;
            }
        }


        feedback_msg_.header.stamp = ros::Time::now();

    if(sizeof(joint_states_.position) == 0)
    {
        for (size_t j = 0; j < joint_names_.size(); ++j)
        {
            feedback_msg_.desired.positions[j] = joint_states_.position[j];
            feedback_msg_.desired.velocities[j] = std::abs(joint_states_.velocity[j
                ]);
            feedback_msg_.actual.positions[j] = joint_states_.position[j];
            feedback_msg_.actual.velocities[j] = std::abs(joint_states_.velocity[j
                ]);
            feedback_msg_.error.positions[j] = feedback_msg_.actual.positions[j] -
                feedback_msg_.desired.positions[j];
            feedback_msg_.error.velocities[j] = feedback_msg_.actual.velocities[j]
                - feedback_msg_.desired.velocities[j];
        }
    }

        state_pub_.publish(feedback_msg_);

        rate.sleep();
    }
}

/**
*   Check Controller Joints
*       - Check that all the joints in the trajectory exist in this multiDOF
    controller
*/

bool JointTrajectoryActionController::checkControllerJoints(const trajectory_msgs::
    JointTrajectory& traj_msg, std::vector<int> &lookup)
{
    // Maps from an index in joint_names_ to an index in the JointTrajectory msg "
        traj_msg"

    for (size_t j = 0; j < num_joints_; ++j)
    {
        for (size_t k = 0; k < traj_msg.joint_names.size(); ++k)
        {
            if (traj_msg.joint_names[k] == joint_names_[j])
            {
                lookup[j] = k;
                break;
            }
        }
```

```
        if (lookup[j] == -1)
        {
        generateErrorNotLocatedJoint(j);
            return false;
        }
    }

    return true;
}

/**
*   Generates a error with unable to locate joint
*/

void JointTrajectoryActionController::generateErrorNotLocatedJoint(const size_t
    index)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        INVALID_JOINTS;
    error_msg = "Unable to locate joint " + joint_names_[index] + " in the
        commanded trajectory";
    ROS_ERROR("%s", error_msg.c_str());

    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg.c_str());
    }
}

/**
*   Validate initial positions
*/

bool JointTrajectoryActionController::validateInitialPositions(const
    trajectory_msgs::JointTrajectory& traj_msg)
{

    // Check for initial position
    if (traj_msg.points[0].positions.empty())
    {
    generateErrorFirstPointNoPositions();
        return false;
    }

    return true;
}

/**
*   Generates a error: First point of trajectory has no positions
*/

void JointTrajectoryActionController::generateErrorFirstPointNoPositions()
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

        trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
            INVALID_GOAL;
        error_msg = "First point of trajectory has no positions";
        ROS_ERROR("%s", error_msg.c_str());
        if (isAction())
        {
            action_server_->setAborted(trajectory_result, error_msg);
```

```
        }
}


/**
*   Find out segments duration
*/

void JointTrajectoryActionController::fillSegmentsDuration(const trajectory_msgs::
    JointTrajectory& traj_msg, std::vector<double> &durations, double &
    trajectory_duration)
{
    int num_points = traj_msg.points.size();

    durations[0] = traj_msg.points[0].time_from_start.toSec();
    trajectory_duration = durations[0];

    for (int i = 1; i < num_points; ++i)
    {
        durations[i] = (traj_msg.points[i].time_from_start - traj_msg.points[i-1].
            time_from_start).toSec();
        trajectory_duration += durations[i];
        ROS_DEBUG("tpi: %f, tpi-1: %f", traj_msg.points[i].time_from_start.toSec(),
             traj_msg.points[i-1].time_from_start.toSec());
        ROS_DEBUG("i: %d, duration: %f, total: %f", i, durations[i],
            trajectory_duration);
    }
}

/*
*   Checks that the incoming segment has the right number of velocity elements
*/

bool JointTrajectoryActionController::checkSegmentVelocityElements(const
    trajectory_msgs::JointTrajectoryPoint point,const int position)
{


        if (!point.velocities.empty() && point.velocities.size() != num_joints_)
        {
        generateErrorUnexpectedVelocities(point.velocities.size(),position);
        return false;
        }

    return true;
}

/**
*   Generates a error: a command point has not the right number of velocities.
*/

void JointTrajectoryActionController::generateErrorUnexpectedVelocities(const
    size_t size, const int position)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        INVALID_GOAL;
    error_msg =     "Command point " + boost::lexical_cast<std::string>(position) +
         " has " +
            boost::lexical_cast<std::string>(size) +
            " elements for the velocities, expecting " + boost::lexical_cast<std::
                string>(num_joints_);

    ROS_ERROR("%s", error_msg.c_str());
```

```
    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }
}


/*
*   Checks that the point has a not empty number of velocities
*/

bool JointTrajectoryActionController::checkPointWithoutVelocity(const
    trajectory_msgs::JointTrajectoryPoint point)
{

        if(point.velocities.size() == 0)
        {
         generateErrorNoVelocities();
            return false;
        }

    return true;
}

/**
*   Generates a error: No velocities included in the trajectory method.
*/

void JointTrajectoryActionController::generateErrorNoVelocities()
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    error_msg = "No velocities included in the trajectory method!";
    ROS_ERROR("%s", error_msg.c_str());

    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }

}



/*
*   Checks that the incoming segment has the right number of position elements
*/

bool JointTrajectoryActionController::checkSegmentPositionElements(const
    trajectory_msgs::JointTrajectoryPoint point,const int position)
{

        if (!point.positions.empty() && point.positions.size() != num_joints_)
        {
        generateErrorUnexpectedPositions(point.positions.size(),position);
            return false;
        }

    return true;
}

/**
*   Generates a error: a command point has not the right number of positions.
*/
```

```
void JointTrajectoryActionController::generateErrorUnexpectedPositions(const size_t
     size, const int position)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        INVALID_GOAL;
    error_msg = "Command point " + boost::lexical_cast<std::string>(position) + "
        has " +
            boost::lexical_cast<std::string>(size) +
            " elements for the positions, expecting " + boost::lexical_cast<std::
                string>(num_joints_);

    ROS_ERROR("%s", error_msg.c_str());

    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }
}


/*
*   Create Segments
*       - Decide segments start time
*/

bool JointTrajectoryActionController::createSegments(const trajectory_msgs::
    JointTrajectory& traj_msg, const ros::Time time, const std::vector<double>
    durations , const std::vector<int> lookup, std::vector<Segment> &trajectory)
{
    int num_points = traj_msg.points.size();

    for (int i = 0; i < num_points; ++i)
    {
        const trajectory_msgs::JointTrajectoryPoint point = traj_msg.points[i];
        Segment seg;

        bool csve = checkSegmentVelocityElements(point,i);
    if(!csve){ return false;}

    bool cspe = checkSegmentPositionElements(point,i);
    if(!cspe){ return false;}

    bool cpwv = checkPointWithoutVelocity(point);
    if(!cpwv){ return false;}

    // Decide segments start time
        if (traj_msg.header.stamp == ros::Time(0.0)) {
            seg.start_time = (time + point.time_from_start).toSec() - durations[i];
        }
        else {
            seg.start_time = (traj_msg.header.stamp + point.time_from_start).toSec
                () - durations[i];
        }

        seg.duration = durations[i];
        seg.velocities.resize(num_joints_);
        seg.positions.resize(num_joints_);

        for (size_t j = 0; j < num_joints_; ++j)
        {
            seg.velocities[j] = point.velocities[lookup[j]];
            seg.positions[j] = point.positions[lookup[j]];
```

```
        }

        trajectory.push_back(seg);
    }

    return true;
}

/**
*   Process Trajectory
*   Error check the trajectory then send it to the controllers
*/

bool JointTrajectoryActionController::checkOnGoal(const std::vector<Segment>
    trajectory, const std::vector<int> lookup)
{

    // Check if this trajectory goal is already fullfilled by robot's current
        position
    bool outside_bounds = false; // flag for remembing if a different position was
        found
    const Segment* last_segment = &trajectory[trajectory.size()-1];

    for( std::size_t i = 0; i < last_segment->positions.size(); ++i)
    {

        std::string joint_name = joint_names_[i];
        int joint_index = lookup[i];

        ROS_DEBUG_STREAM("Checking for similarity on joint " << joint_name << "
            with real position " << joint_states_.position[joint_index]);
        ROS_DEBUG_STREAM("    Iterator id = " << i << " size " << last_segment->
            positions.size() << "    Goal position: " << last_segment->positions[i]
             );

        // Test if outside acceptable bounds, meaning we should continue trajectory
            as normal
        if( last_segment->positions[i] > (joint_states_.position[joint_index] +
            ACCEPTABLE_BOUND) ||
                last_segment->positions[i] < (joint_states_.position[joint_index] -
                    ACCEPTABLE_BOUND) )
        {
            outside_bounds = true;
            break;
        }
    }
    // Check if all the states were inside the position bounds
    if( !outside_bounds )
    {
    generateSuccesfulTrajectorySkipped();
        return true;
    }

    return false;
}

/**
*   Generates a message: Trajectory execution skipped because goal is same as
    current state.
*/

void JointTrajectoryActionController::generateSuccesfulTrajectorySkipped()
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;
```

```
        // We can exit trajectory
        trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
            SUCCESSFUL;
        error_msg = "Trajectory execution skipped because goal is same as current
            state";
        ROS_INFO("%s", error_msg.c_str());
        action_server_->setSucceeded(trajectory_result, error_msg);
}

/**
*   Check motors are within their goal constraints
*/

bool JointTrajectoryActionController::checkGoalConstraints()
{

    for (size_t i = 0; i < num_joints_; ++i)
    {
        if (goal_constraints_[i] > 0 && std::abs(feedback_msg_.error.positions[i])
            > goal_constraints_[i])
        {
        generateErrorGoalConstraints(i);
            return false;
        }
    }

    return true;
}

/**
*   Generates a error: a command point has not the right number of positions.
*/

void JointTrajectoryActionController::generateErrorGoalConstraints(const size_t
    index)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;
    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        GOAL_TOLERANCE_VIOLATED;
                error_msg = "Aborting at end because " + joint_names_[index] +
                " joint wound up outside the goal constraints. The position error "
                    +
                boost::lexical_cast<std::string>(fabs(feedback_msg_.error.positions
                    [index])) +
                " is larger than the goal constraints " + boost::lexical_cast<std::
                    string>(goal_constraints_[index]);
    ROS_ERROR("%s", error_msg.c_str());

    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }
}


/**
*   Verify trajectory constraints
*/

bool JointTrajectoryActionController::validateTrajectoryConstraints(int
    segment_index)
{

    for (size_t j = 0; j < joint_names_.size(); ++j)
    {
```

```
        if (trajectory_constraints_[j] > 0.0 && feedback_msg_.error.positions[j] >
            trajectory_constraints_[j])
        {
           generateErrorTrajectoryConstraints(segment_index, j);
                  return false;
        }
    }

    return true;
}

/**
*   Generates a error: a command point has not the right number of positions.
*/

void JointTrajectoryActionController::generateErrorTrajectoryConstraints(const int
    segment_index, const size_t index)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        PATH_TOLERANCE_VIOLATED;
                error_msg = "Unsatisfied position constraint for " + joint_names_[
                    index] +
                " trajectory point " + boost::lexical_cast<std::string>(
                    segment_index) +
                ", " + boost::lexical_cast<std::string>(feedback_msg_.error.
                    positions[index]) +
                " is larger than " + boost::lexical_cast<std::string>(
                    trajectory_constraints_[index]);

    ROS_ERROR("%s", error_msg.c_str());

    if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }

}


/**
*   Get Joint States
*/

void JointTrajectoryActionController::getJointStates(const sensor_msgs::
    JointStateConstPtr &joint_state)
{
    joint_states_ = *joint_state;
    return;
}

/**
*   Process Trajectory
*   Error check the trajectory then send it to the controllers
*/

void JointTrajectoryActionController::processTrajectory(const trajectory_msgs::
    JointTrajectory& traj_msg)
{
    ROS_INFO_STREAM("\n inside process Trajectory with msg:\n" << traj_msg);

    int num_points = traj_msg.points.size();
    std::vector<int> lookup(num_joints_, -1);
    double trajectory_duration = 0.0;
```

```
std::vector<double> durations;
std::vector<Segment> trajectory;
ros::Time time;
ros::Rate rate(getTrajectoryUpdateRate());

durations.resize(num_points);

ROS_DEBUG("Received trajectory with %d points", num_points);

if(!checkControllerJoints(traj_msg, lookup)) return;
ROS_DEBUG("Checked controller joints");

if(!validateInitialPositions(traj_msg)) return;
ROS_DEBUG("Validated initial positions");

fillSegmentsDuration(traj_msg, durations,trajectory_duration);
ROS_DEBUG("Found out segments duration");

time = ros::Time::now() + ros::Duration(0.01);
if(!createSegments(traj_msg,time,durations,lookup,trajectory)) return;
ROS_DEBUG("Segments created");

if(checkOnGoal(trajectory, lookup)) return;
ROS_DEBUG("Not on goal... going on...");

ROS_INFO("Trajectory start requested at %.3lf, waiting...", traj_msg.header.
    stamp.toSec());
ros::Time::sleepUntil(traj_msg.header.stamp);

ros::Time end_time = traj_msg.header.stamp + ros::Duration(trajectory_duration)
    ;
std::vector<ros::Time> seg_end_times(num_points, ros::Time(0.0));

for (int i = 0; i < num_points; ++i)
{
    seg_end_times[i] = ros::Time(trajectory[i].start_time + durations[i]);
}

ROS_INFO("Trajectory start time is %.3lf, end time is %.3lf, total duration is
    %.3lf", time.toSec(), end_time.toSec(), trajectory_duration);

trajectory_ = trajectory;

//
    ------------------------------------------------------------------------------

// The main loop

for (int segment_index = 0; segment_index < num_points; ++segment_index)
{
    ROS_DEBUG("Processing segment %d
        ------------------------------------------------------", segment_index);

    // first point in trajectories calculated by OMPL is current position with
        duration of 0 seconds, skip it
    if (durations[segment_index] == 0.0)
    {
        ROS_DEBUG("Skipping segment %d because duration is 0", segment_index);
        continue;
    }

std::vector<double> desired_velocities(num_joints_,-1);
    // Loop through every joint
for(int joint_index = 0; joint_index < num_joints_ ; joint_index++)
    {
```

```
            // Get start position of this joint
            double start_position;

            if (segment_index != 0)
            {
                start_position = trajectory[segment_index-1].positions[
                    joint_index];
            }
            else
            {
                start_position = joint_states_.position[lookup[joint_index]];
            }

            // Calculate desired values
            double desired_position = trajectory[segment_index].positions[
                joint_index];
            //double desired_velocity = std::max<double>(min_velocity_,
            //                          std::abs(desired_position -
                start_position) /
            //                          durations[segment_index]);

    double desired_velocity = (desired_position - start_position) / durations[
        segment_index];

    /* TODO
    if(desired_velocity < 0 && std::abs(desired_velocity) < min_velocity_)
    {
        desired_velocity = -min_velocity_;
    }
    else if(desired_velocity > 0 && std::abs(desired_velocity) < min_velocity_)
    {
        desired_velocity = min_velocity_;
    }
    else
    {
        desired_velocity = 0;
    }
    */

    desired_velocities[joint_index] = desired_velocity;

            ROS_DEBUG("\tstart_position: %f, duration: %f", start_position,
                durations[segment_index]);

            if( std::abs(desired_velocity) > max_velocity_ )
            {
        generateErrorMaxVelocityExceeded(segment_index,joint_index,
            desired_velocity);
                return;
            }
    }

publishVelocitiesMessage(getVelocitiesMessage(desired_velocities));
rate.sleep();

    // Now wait for the next segment to be ready to go
    time = ros::Time::now(); //TODO maybe remove this

    while (time < seg_end_times[segment_index])
    {
        // check if new trajectory was received, if so abort old one
        // setting the desired position of all the servos to each of their
            current states, so as to pause everything
        if (isAction() && action_server_->isPreemptRequested())
        {
```

```
        for(int joint_index = 0; joint_index < num_joints_ ; joint_index++)
                {

                        double desired_position = joint_states_.position[lookup[
                            joint_index]];
                        double desired_velocity = joint_states_.velocity[lookup[
                            joint_index]];

            desired_velocities[joint_index] = desired_velocity;
                }

        publishVelocitiesMessage(getVelocitiesMessage(desired_velocities));
        generateSuccesfulTrajectoryPreemted();
        }


        publishVelocitiesMessage(getVelocitiesMessage(desired_velocities));
            rate.sleep(); //TODO maybe uncomment this
            time = ros::Time::now();
        }
    // Send zero velocity after each segment.
    publishVelocitiesMessage(getVelocitiesMessageZero());

        bool validTrajectoryConstraints = JointTrajectoryActionController::
            validateTrajectoryConstraints(segment_index);
        if (!validTrajectoryConstraints)
            return;


    }
    // end of the main loop

    // let motors roll for specified amount of time
    ros::Duration(goal_time_constraint_).sleep();

    bool validGoalConstraints = JointTrajectoryActionController::
        checkGoalConstraints();
    if( !validGoalConstraints )
        return;

    //
        ____________________________________________________________________________________________________

    // Finish up - Set result

    generateSuccesfulTrajectory();

}

/**
*   Generates a error: max velocity exceeded for a joint.
*/

void JointTrajectoryActionController::generateErrorMaxVelocityExceeded(const int
    segment_index, const int joint_index, const double desired_velocity)
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        PATH_TOLERANCE_VIOLATED;
    error_msg = "Invalid joint trajectory: max velocity exceeded for joint " +
        joint_names_[joint_index] +
                    " with a velocity of " + boost::lexical_cast<std::string>(
                        desired_velocity) +
```

```
                    " when the max velocity is set to " + boost::lexical_cast<std::
                        string>(max_velocity_) +
                    ". On trajectory step " + boost::lexical_cast<std::string>(
                        segment_index);

    ROS_ERROR("%s", error_msg.c_str());

        if (isAction())
    {
        action_server_->setAborted(trajectory_result, error_msg);
    }
}

/**
*   Generates a succesful trajectory: Canceled trajectory (preemted).
*/

void JointTrajectoryActionController::generateSuccesfulTrajectoryPreemted()
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        SUCCESSFUL;
        error_msg = "Trajectory controller recieved preempt request from action
            server. Canceling trajectory.";

    action_server_->setPreempted(trajectory_result, error_msg);
        ROS_WARN("%s", error_msg.c_str());
}

/**
*   Generates a succesful trajectory.
*/

void JointTrajectoryActionController::generateSuccesfulTrajectory()
{
    control_msgs::FollowJointTrajectoryResult trajectory_result;
    std::string error_msg;

    trajectory_result.error_code = control_msgs::FollowJointTrajectoryResult::
        SUCCESSFUL;
    error_msg = "Trajectory execution successfully completed";
    ROS_INFO("%s", error_msg.c_str());

    action_server_->setSucceeded(trajectory_result, error_msg);
}

/**
*   Generates velocity command to arm.
*/

brics_actuator::JointVelocities JointTrajectoryActionController::
    getVelocitiesMessage(const std::vector<double> velocities)
{
    std::string error_msg;
    error_msg = "Building brics_actuator/JointVelocities Message";
    ROS_INFO("%s", error_msg.c_str());

    brics_actuator::JointVelocities jointVelocitiesMsg;

    jointVelocitiesMsg.poisonStamp.originator = "";
    jointVelocitiesMsg.poisonStamp.description = "";
    jointVelocitiesMsg.poisonStamp.qos = 0.9;

    jointVelocitiesMsg.velocities.resize(num_joints_);
```

```
    for(size_t i=0; i < num_joints_; i++)
    {
        //jointVelocitiesMsg.velocities[i].timeStamp.secs = 0;
        //jointVelocitiesMsg.velocities[i].timeStamp.nsecs = 0;
        jointVelocitiesMsg.velocities[i].joint_uri = joint_names_[i];
        jointVelocitiesMsg.velocities[i].unit = "rad";
        jointVelocitiesMsg.velocities[i].value = velocities[i];
    }

    return jointVelocitiesMsg;
}

brics_actuator::JointVelocities JointTrajectoryActionController::
    getVelocitiesMessageZero()
{
    std::string error_msg;
    error_msg = "Building brics_actuator/JointVelocities Message";
    ROS_INFO("%s", error_msg.c_str());

    brics_actuator::JointVelocities jointVelocitiesMsg;

    jointVelocitiesMsg.poisonStamp.originator = "";
    jointVelocitiesMsg.poisonStamp.description = "";
    jointVelocitiesMsg.poisonStamp.qos = 0.9;

    jointVelocitiesMsg.velocities.resize(num_joints_);

    for(size_t i=0; i < num_joints_; i++)
    {
        //jointVelocitiesMsg.velocities[i].timeStamp.secs = 0;
        //jointVelocitiesMsg.velocities[i].timeStamp.nsecs = 0;
        jointVelocitiesMsg.velocities[i].joint_uri = joint_names_[i];
        jointVelocitiesMsg.velocities[i].unit = "rad";
        jointVelocitiesMsg.velocities[i].value = 0;
    }

    return jointVelocitiesMsg;
}
/**
*   Publish velocity command to arm.
*/

void JointTrajectoryActionController::publishVelocitiesMessage(const brics_actuator
    ::JointVelocities velocities)
{
    std::string error_msg;
    error_msg = "Sending brics_actuator/JointVelocities Message";
    ROS_INFO("%s", error_msg.c_str());

        velocity_pub_.publish(velocities);
}

//=============================================================================
// MAIN                                                                       =
//=============================================================================

int main(int argc, char **argv)
{
    ros::init(argc, argv, "rescuer_joint_action_controller");
    //ros::NodeHandle n;

    //ros::ServiceServer service = n.advertiseService("Server", Server::Move);

    JointTrajectoryActionController jtac;
    jtac.start();
```

```
    ROS_INFO("Ready to move rescuer arm.");
    ros::spin();

    return 0;
}
```

B. Cálculo de las matrices de inercia del brazo

B.1. Módulo 1

```
LOG: 0 Opened mesh modular1.stl in 283 msec
LOG: 0 All files opened in 1880 msec
LOG: 0 Started Mode Measuring Tool
LOG: 2 Mesh Bounding Box Size 0.130000 0.148000 0.307999
LOG: 2 Mesh Bounding Box Diag 0.365605
LOG: 2 Mesh Volume  is 0.002147
LOG: 2 Mesh Surface is 0.176083
LOG: 2 Thin shell barycenter   0.000001  -0.020811   0.100715
LOG: 2 Center of Mass  is 0.000003 -0.009599 0.141766
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.000012  -0.000000  -0.000000 |
LOG: 2     | -0.000000   0.000012  -0.000002 |
LOG: 2     | -0.000000  -0.000002   0.000005 |
LOG: 2 Principal axes are :
LOG: 2     |  0.999994  -0.003515   0.000028 |
LOG: 2     |  0.003336   0.951636   0.307208 |
LOG: 2     | -0.001107  -0.307206   0.951642 |
LOG: 2 axis momenta are :
LOG: 2     |  0.000012   0.000013   0.000005 |
LOG: 0 Applied filter Compute Geometric Measures in 40 msec

SCALE FACTOR: 1/Mesh Volume
    s^3 = 1/0.002147 = 465.76618537494176
    s = 7.751563668531336
    s Meshlab = 7.751563

Getting info again:

LOG: 2 Mesh Bounding Box Size 1.007703 1.147231 2.387471
LOG: 2 Mesh Bounding Box Diag 2.834012
LOG: 2 Mesh Volume  is 0.999817
LOG: 2 Mesh Surface is 10.579684
LOG: 2 Thin shell barycenter   0.000002  -0.080299   0.179806
LOG: 2 Center of Mass  is 0.000017 0.006611 0.498014
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.346973  -0.000016  -0.000001 |
LOG: 2     | -0.000016   0.330281  -0.064760 |
LOG: 2     | -0.000001  -0.064760   0.150580 |
LOG: 2 Principal axes are :
LOG: 2     |  0.999994  -0.003515   0.000028 |
LOG: 2     |  0.003336   0.951637   0.307208 |
LOG: 2     | -0.001107  -0.307206   0.951642 |
LOG: 2 axis momenta are :
LOG: 2     |  0.346973   0.351187   0.129674 |
LOG: 0 Applied filter Compute Geometric Measures in 41 msec

Multiplicando Inertia Tensor por 1/s^2
```

```
octave:1> A = [0.346973, -0.000016, -0.000001; -0.000016 ,0.330281 ,-0.064760;
    -0.000001, -0.064760, 0.150580]
A =

   3.4697e-01  -1.6000e-05  -1.0000e-06
  -1.6000e-05   3.3028e-01  -6.4760e-02
  -1.0000e-06  -6.4760e-02   1.5058e-01

octave:2> A*1/(7.751563^2)
ans =

   5.7745e-03  -2.6628e-07  -1.6643e-08
  -2.6628e-07   5.4967e-03  -1.0778e-03
  -1.6643e-08  -1.0778e-03   2.5060e-03


<xacro:property name="inertia_1_link">
    <inertia        ixx="5.7745e-03" ixy="-2.6628e-07" ixz="-1.6643e-08"
                                     iyy="5.4967e-03"  iyz="-1.0778e-03"
                                                       izz="2.5060e-03"/>
```

B.2. Módulo 2

```
=================================================================
Primera computación de la matriz de inercia desde meshlab:
=================================================================

LOG: 2 Mesh Bounding Box Size 0.124000 0.511919 0.189000
LOG: 2 Mesh Bounding Box Diag 0.559606
LOG: 2 Mesh Volume  is 0.002741
LOG: 2 Mesh Surface is 0.351974
LOG: 2 Thin shell barycenter  -0.000000  -0.157323  -0.079652
LOG: 2 Center of Mass  is -0.000001 -0.071960 -0.043237
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.000056   0.000000   0.000000 |
LOG: 2     |  0.000000   0.000011  -0.000011 |
LOG: 2     |  0.000000  -0.000011   0.000051 |
LOG: 2 Principal axes are :
LOG: 2     |  1.000000  -0.000001  -0.000014 |
LOG: 2     | -0.000002   0.969260  -0.246039 |
LOG: 2     |  0.000013   0.246039   0.969260 |
LOG: 2 axis momenta are :
LOG: 2     |  0.000056   0.000008   0.000054 |


=================================================================
Factor de escala: 1/Mesh Volume
=================================================================

    s^3 = 1/0.002741 = 364.83
    s = 7.1455
    s Meshlab = 7.1455

=================================================================
Segunda computación de la matriz de inercia desde meshlab:
=================================================================

LOG: 2 Mesh Bounding Box Size 0.886042 3.657920 1.350500
LOG: 2 Mesh Bounding Box Diag 3.998662
LOG: 2 Mesh Volume  is 1.000142
LOG: 2 Mesh Surface is 17.971153
LOG: 2 Thin shell barycenter  -0.000000   0.067822  -0.363275
LOG: 2 Center of Mass  is -0.000004 0.677790 -0.103073
LOG: 2 Inertia Tensor is :
```

```
LOG: 2      |  1.051952   0.000001   0.000001 |
LOG: 2      |  0.000001   0.208077  -0.203099 |
LOG: 2      |  0.000001  -0.203099   0.956622 |
LOG: 2 Principal axes are :
LOG: 2      |  1.000000  -0.000001  -0.000014 |
LOG: 2      | -0.000002   0.969260  -0.246039 |
LOG: 2      |  0.000013   0.246039   0.969260 |
LOG: 2 axis momenta are :
LOG: 2      |  1.051952   0.156522   1.008177 |
```

Multiplicando Inertia Tensor por 1/s^2 en Octave

```
octave:8> A = [1.051952 ,  0.000001  , 0.000001; 0.000001  , 0.208077 ,
    -0.203099;0.000001,  -0.203099 ,  0.956622]
A =

   1.0520e+00   1.0000e-06   1.0000e-06
   1.0000e-06   2.0808e-01  -2.0310e-01
   1.0000e-06  -2.0310e-01   9.5662e-01

octave:9> A*(1/7.1455^2)
ans =

   2.0603e-02   1.9586e-08   1.9586e-08
   1.9586e-08   4.0753e-03  -3.9778e-03
   1.9586e-08  -3.9778e-03   1.8736e-02
```

URDF tags

```
<xacro:property name="inertia_2_link">
<inertia         ixx="2.0603e-02" ixy="1.9586e-08" ixz="1.9586e-08"
                                  iyy="4.0753e-03" iyz="-3.9778e-03"
                                                   izz="1.8736e-02"/>
</xacro:property>
```

B.3. Módulo 3

Primera computación de la matriz de inercia desde meshlab:

```
LOG: 2 Mesh Bounding Box Size 0.110949 0.141000 0.143001
LOG: 2 Mesh Bounding Box Diag 0.229434
LOG: 2 Mesh Volume  is 0.001382
LOG: 2 Mesh Surface is 0.115053
LOG: 2 Thin shell barycenter   0.000000   0.000733  -0.021030
LOG: 2 Center of Mass  is 0.000000 0.013305 -0.006775
LOG: 2 Inertia Tensor is :
LOG: 2      |  0.000003   0.000000   0.000000 |
LOG: 2      |  0.000000   0.000003  -0.000000 |
LOG: 2      |  0.000000  -0.000000   0.000003 |
LOG: 2 Principal axes are :
LOG: 2      |  1.000000  -0.000010   0.000010 |
LOG: 2      |  0.000013   0.923658  -0.383219 |
LOG: 2      | -0.000006   0.383219   0.923658 |
LOG: 2 axis momenta are :
LOG: 2      |  0.000003   0.000002   0.000003 |
```

```
Factor de escala: 1/Mesh Volume
```

```
    s^3 = 1/0.001382 = 723.59
    s = 8.9777
    s Meshlab = 8.9777
```

```
Segunda computación de la matriz de inercia desde meshlab:
```

```
LOG: 2 Mesh Bounding Box Size 0.996069 1.265856 1.283820
LOG: 2 Mesh Bounding Box Diag 2.059791
LOG: 2 Mesh Volume  is 1.000350
LOG: 2 Mesh Surface is 9.273326
LOG: 2 Thin shell barycenter   0.000000  -0.077184  -0.061167
LOG: 2 Center of Mass  is 0.000001 0.035681 0.066816
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.204017   0.000001   0.000000 |
LOG: 2     |  0.000001   0.151848  -0.014833 |
LOG: 2     |  0.000000  -0.014833   0.181446 |
LOG: 2 Principal axes are :
LOG: 2     |  1.000000  -0.000010   0.000010 |
LOG: 2     |  0.000013   0.923657  -0.383219 |
LOG: 2     | -0.000006   0.383219   0.923657 |
LOG: 2 axis momenta are :
LOG: 2     |  0.204017   0.145694   0.187601 |
```

```
Multiplicando Inertia Tensor por 1/s^2 en Octave
```

```
octave:12> A = [0.204017 ,  0.000001  , 0.000000; 0.000001 ,  0.151848,  -0.014833;
     0.000000 , -0.014833 ,  0.181446]
A =

   0.20402   0.00000   0.00000
   0.00000   0.15185  -0.01483
   0.00000  -0.01483   0.18145

octave:13> A*1/(8.9777^2)
ans =

   0.0025313   0.0000000   0.0000000
   0.0000000   0.0018840  -0.0001840
   0.0000000  -0.0001840   0.0022512
```

```
URDF tags
```

```
<xacro:property name="inertia_3_link">
    <inertia         ixx="0.0025313" ixy="0"           ixz="0"
                                     iyy="0.0018840"   iyz="-0.0001840"
                                                       izz="0.0022512"/>
        </xacro:property>
```

B.4. Módulo 4

```
Primera computación de la matriz de inercia desde meshlab:
```

```
LOG: 2 Mesh Bounding Box Size 0.100000 0.225498 0.102000
LOG: 2 Mesh Bounding Box Diag 0.266933
```

```
LOG: 2 Mesh Volume  is 0.000735
LOG: 2 Mesh Surface is 0.094663
LOG: 2 Thin shell barycenter   0.000014   0.075847  -0.014742
LOG: 2 Center of Mass  is 0.000015 0.106227 -0.008144
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.000002   0.000000   0.000000 |
LOG: 2     |  0.000000   0.000001  -0.000000 |
LOG: 2     |  0.000000  -0.000000   0.000002 |
LOG: 2 Principal axes are :
LOG: 2     |  0.999994  -0.000777   0.003344 |
LOG: 2     |  0.001744   0.954000  -0.299802 |
LOG: 2     | -0.002957   0.299807   0.953995 |
LOG: 2 axis momenta are :
LOG: 2     |  0.000002   0.000001   0.000002 |
```

```
Factor de escala: 1/Mesh Volume
```

```
    s^3 = 1/0.000735 = 1360.5
    s =
    s Meshlab = 11.081
```

```
Segunda computación de la matriz de inercia desde meshlab:
    Meshlab sólo permite escalar hasta 10
```

```
LOG: 2 Mesh Bounding Box Size 1.000000 2.254980 1.020000
LOG: 2 Mesh Bounding Box Diag 2.669333
LOG: 2 Mesh Volume  is 0.734756
LOG: 2 Mesh Surface is 9.466218
LOG: 2 Thin shell barycenter   0.000143   0.130708  -0.066421
LOG: 2 Center of Mass  is 0.000153 0.434507 -0.000439
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.241213   0.000115   0.000055 |
LOG: 2     |  0.000115   0.093452  -0.048112 |
LOG: 2     |  0.000055  -0.048112   0.231426 |
LOG: 2 Principal axes are :
LOG: 2     |  0.999994  -0.000777   0.003344 |
LOG: 2     |  0.001744   0.954000  -0.299802 |
LOG: 2     | -0.002957   0.299807   0.953995 |
LOG: 2 axis momenta are :
LOG: 2     |  0.241213   0.078332   0.246546 |
```

```
Factor de escala: 1/Mesh Volume
```

```
    s^3 = 1/0.734756 = 1.3610
    s = 1.1082
    s Meshlab = 1.1082
```

```
Tercera computación de la matriz de inercia desde meshlab:
```

```
LOG: 2 Mesh Bounding Box Size 1.108200 2.498969 1.130364
LOG: 2 Mesh Bounding Box Diag 2.958154
LOG: 2 Mesh Volume  is 0.999994
LOG: 2 Mesh Surface is 11.626400
LOG: 2 Thin shell barycenter   0.000158   0.137297  -0.072634
LOG: 2 Center of Mass  is 0.000169 0.473974 0.000488
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.403173   0.000193   0.000092 |
LOG: 2     |  0.000193   0.156200  -0.080416 |
```

```
LOG: 2      |   0.000092   -0.080416    0.386815 |
LOG: 2 Principal axes are :
LOG: 2      |   0.999994   -0.000777    0.003344 |
LOG: 2      |   0.001744    0.954000   -0.299802 |
LOG: 2      |  -0.002957    0.299807    0.953995 |
LOG: 2 axis momenta are :
LOG: 2      |   0.403173    0.130928    0.412087 |
```

Multiplicando Inertia Tensor por 1/s^2 en Octave

```
octave:21> A = [0.403173    0.000193    0.000092; 0.000193    0.156200   -0.080416;
    0.000092   -0.080416    0.386815]
A =

   4.0317e-01    1.9300e-04    9.2000e-05
   1.9300e-04    1.5620e-01   -8.0416e-02
   9.2000e-05   -8.0416e-02    3.8682e-01

octave:22> A*(1/(11.082^2))
ans =

   3.2829e-03    1.5715e-06    7.4912e-07
   1.5715e-06    1.2719e-03   -6.5480e-04
   7.4912e-07   -6.5480e-04    3.1497e-03
```

URDF tags

```
<xacro:property name="inertia_4_link">
    <inertia         ixx="3.2829e-03" ixy="1.5715e-06" ixz="7.4912e-07"
                                      iyy="1.2719e-03"  iyz="-6.5480e-04"
                                                        izz="3.1497e-03"/>
</xacro:property>
```

B.5. Módulo 5

Primera computación de la matriz de inercia desde meshlab:

```
LOG: 2 Mesh Bounding Box Size 0.114500 0.080000 0.113000
LOG: 2 Mesh Bounding Box Diag 0.179664
LOG: 2 Mesh Volume  is 0.000625
LOG: 2 Mesh Surface is 0.074786
LOG: 2 Thin shell barycenter    0.023939    0.000000    0.021592
LOG: 2 Center of Mass  is 0.016561 0.000000 0.008731
LOG: 2 Inertia Tensor is :
LOG: 2      |   0.000001    0.000000   -0.000000 |
LOG: 2      |   0.000000    0.000001   -0.000000 |
LOG: 2      |  -0.000000   -0.000000    0.000001 |
LOG: 2 Principal axes are :
LOG: 2      |   0.895225    0.000058   -0.445615 |
LOG: 2      |  -0.000017    1.000000    0.000096 |
LOG: 2      |   0.445615   -0.000078    0.895225 |
LOG: 2 axis momenta are :
LOG: 2      |   0.000001    0.000001    0.000001 |
```

Factor de escala: 1/Mesh Volume

```
s^3 = 1/0.000625 = 1600
s = 11.696
s Meshlab = 10
```

```
Segunda computación de la matriz de inercia desde meshlab:
```

```
LOG: 2 Mesh Bounding Box Size 1.145000 0.800000 1.130000
LOG: 2 Mesh Bounding Box Diag 1.796643
LOG: 2 Mesh Volume  is 0.624802
LOG: 2 Mesh Surface is 7.478780
LOG: 2 Thin shell barycenter   0.084133   0.000001   0.067421
LOG: 2 Center of Mass  is 0.010363 0.000001 -0.061189
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.073831   0.000001  -0.010412 |
LOG: 2     |  0.000001   0.108340  -0.000001 |
LOG: 2     | -0.010412  -0.000001   0.089565 |
LOG: 2 Principal axes are :
LOG: 2     |  0.895225   0.000058  -0.445615 |
LOG: 2     | -0.000017   1.000000   0.000096 |
LOG: 2     |  0.445615  -0.000078   0.895225 |
LOG: 2 axis momenta are :
LOG: 2     |  0.068649   0.108340   0.094748 |
```

```
Factor de escala: 1/Mesh Volume
```

```
s^3 = 1/0.624802 = 1.6005
s = 1.1697
s Meshlab = 1.1697
```

```
Tercera computación de la matriz de inercia desde meshlab:
```

```
LOG: 2 Mesh Bounding Box Size 1.339307 0.935760 1.321761
LOG: 2 Mesh Bounding Box Diag 2.101533
LOG: 2 Mesh Volume  is 0.999922
LOG: 2 Mesh Surface is 10.232059
LOG: 2 Thin shell barycenter   0.095487   0.000001   0.076068
LOG: 2 Center of Mass  is 0.009194 0.000002 -0.074373
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.161664   0.000003  -0.022798 |
LOG: 2     |  0.000003   0.237226  -0.000002 |
LOG: 2     | -0.022798  -0.000002   0.196115 |
LOG: 2 Principal axes are :
LOG: 2     |  0.895225   0.000058  -0.445615 |
LOG: 2     | -0.000017   1.000000   0.000096 |
LOG: 2     |  0.445615  -0.000078   0.895225 |
LOG: 2 axis momenta are :
LOG: 2     |  0.150316   0.237226   0.207463 |
```

```
Multiplicando Inertia Tensor por 1/s^2 en Octave
```

```
octave:27> A = [0.161664   0.000003  -0.022798; 0.000003   0.237226  -0.000002;
    -0.022798  -0.000002   0.196115]
A =

   1.6166e-01   3.0000e-06  -2.2798e-02
   3.0000e-06   2.3723e-01  -2.0000e-06
```

```
  -2.2798e-02  -2.0000e-06   1.9612e-01

octave:28> A*(1/(11.697^2))
ans =

   1.1816e-03   2.1927e-08  -1.6663e-04
   2.1927e-08   1.7339e-03  -1.4618e-08
  -1.6663e-04  -1.4618e-08   1.4334e-03
```

URDF tags

```
<xacro:property name="inertia_5_link">
    <inertia        ixx="1.1816e-03" ixy="2.1927e-08"  ixz="-1.6663e-04"
                                     iyy="1.7339e-03"  iyz="-1.4618e-08"
                                                       izz="1.4334e-03"/>
</xacro:property>
```

B.6. Módulo 6

Primera computación de la matriz de inercia desde meshlab:

```
LOG: 2 Mesh Bounding Box Size 0.092753 0.092753 0.116700
LOG: 2 Mesh Bounding Box Diag 0.175571
LOG: 2 Mesh Volume  is 0.000319
LOG: 2 Mesh Surface is 0.055319
LOG: 2 Thin shell barycenter   0.000000   0.000000  -0.030816
LOG: 2 Center of Mass  is -0.000000 -0.000000 -0.054637
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.000000  -0.000000   0.000000 |
LOG: 2     | -0.000000   0.000000   0.000000 |
LOG: 2     |  0.000000   0.000000   0.000000 |
LOG: 2 Principal axes are :
LOG: 2     |  0.937922  -0.346846  -0.000009 |
LOG: 2     |  0.346846   0.937922  -0.000011 |
LOG: 2     |  0.000012   0.000007   1.000000 |
LOG: 2 axis momenta are :
LOG: 2     |  0.000000   0.000000   0.000000 |
```

Factor de escala: 1/Mesh Volume

```
s^3 = 1/0.000319 = 3134.8
s = 14.635
s Meshlab = 10
```

Segunda computación de la matriz de inercia desde meshlab:

```
LOG: 2 Mesh Bounding Box Size 0.927528 0.927528 1.167000
LOG: 2 Mesh Bounding Box Diag 1.755707
LOG: 2 Mesh Volume  is 0.318863
LOG: 2 Mesh Surface is 5.531906
LOG: 2 Thin shell barycenter   0.000000   0.000000   0.171993
LOG: 2 Center of Mass  is -0.000002 -0.000002 -0.066217
LOG: 2 Inertia Tensor is :
LOG: 2     |  0.045067  -0.000000   0.000000 |
LOG: 2     | -0.000000   0.045067   0.000000 |
LOG: 2     |  0.000000   0.000000   0.019191 |
```

```
LOG: 2 Principal axes are :
LOG: 2      |  0.928192  -0.372103  -0.000009 |
LOG: 2      |  0.372103   0.928192  -0.000011 |
LOG: 2      |  0.000012   0.000006   1.000000 |
LOG: 2 axis momenta are :
LOG: 2      |  0.045067   0.045067   0.019191 |
```

Factor de escala: 1/Mesh Volume

```
s^3 = 1/0.318863 = 1.4637
s = 1.4637
s Meshlab = 1.4637
```

Tercera computación de la matriz de inercia desde meshlab:

```
LOG: 2 Mesh Bounding Box Size 1.357623 1.357623 1.708138
LOG: 2 Mesh Bounding Box Diag 2.569828
LOG: 2 Mesh Volume  is 0.999908
LOG: 2 Mesh Surface is 11.851395
LOG: 2 Thin shell barycenter   0.000001   0.000000   0.276473
LOG: 2 Center of Mass  is -0.000002 -0.000003 -0.072183
LOG: 2 Inertia Tensor is :
LOG: 2      |  0.302775  -0.000000   0.000002 |
LOG: 2      | -0.000000   0.302775   0.000002 |
LOG: 2      |  0.000002   0.000002   0.128934 |
LOG: 2 Principal axes are :
LOG: 2      |  0.928863  -0.370424  -0.000009 |
LOG: 2      |  0.370424   0.928863  -0.000011 |
LOG: 2      |  0.000012   0.000007   1.000000 |
LOG: 2 axis momenta are :
LOG: 2      |  0.302775   0.302775   0.128934 |
```

Multiplicando Inertia Tensor por 1/s^2 en Octave

```
A = [0.302775 , -0.000000 ,  0.000002; -0.000000,   0.302775  , 0.000002; 0.000002
    ,  0.000002 ,  0.128934]
A =

   0.30278  -0.00000   0.00000
  -0.00000   0.30278   0.00000
   0.00000   0.00000   0.12893

octave:34> A*(1/(14.637^2))
ans =

   0.0014132  -0.0000000   0.0000000
  -0.0000000   0.0014132   0.0000000
   0.0000000   0.0000000   0.0006018
```

URDF tags

```
<xacro:property name="inertia_6_link">
    <inertia         ixx="0.0014132" ixy="0"           ixz="0"
                                     iyy="0.0014132"   iyz="0"
                                                       izz="0.0006018"/>
</xacro:property>
```

C. rescuer_description

La estructura del paquete *rescuer_description* puede verse en la Figura C.1. La configuración del paquete se deja de la forma que puede verse en el fichero *CMakeLists.txt* (ver Figura C.2) y en el fichero *package.xml* (Figura C.3).

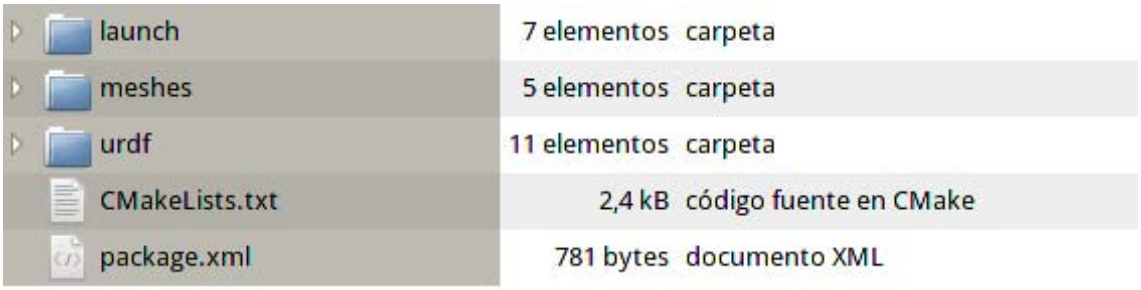


Figura C.1.: Estructura del paquete rescuer_description

```
cmake_minimum_required(VERSION 2.8.3)
project(rescuer_description)
```

Figura C.2.: CMakesLists.txt de rescuer_description

Las dependencias del fichero *package.xml,* no son necesarias por lo que convendría eliminarlas.

C.1. Descripción completa en urdf

```
<?xml version="1.0" ?>
<!--
  ================================================================================
  |    This document was autogenerated by xacro from rescuer.urdf.xacro
  |    EDITING THIS FILE BY HAND IS NOT RECOMMENDED
  ================================================================================
-->
<robot name="rescuer" xmlns:xacro="http://www.ros.org/wiki/xacro">
  <material name="Rover/Red">
```

```
<?xml version="1.0"?>
<package>
  <name>rescuer_description</name>
  <version>0.0.0</version>
  <description>The rescuer_description package</description>

  <maintainer email="xulioxesus@gmail.com">rescuer</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>actionlib</build_depend>
  <build_depend>actionlib_msgs</build_depend>
  <build_depend>message_generation</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>actionlib</run_depend>
  <run_depend>actionlib_msgs</run_depend>
  <run_depend>message_runtime</run_depend>

</package>
```

Figura C.3.: package.xml de rescuer_description

```
    <color rgba="1.0 0.0 0.0 1.0"/>
  </material>
  <material name="Rover/Black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/LightGrey">
    <color rgba="0.7 0.7 0.7 1.0"/>
  </material>
  <material name="Schunk/DarkGrey">
    <color rgba="0.4 0.4 0.4 1.0"/>
  </material>
  <material name="Schunk/Black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/DarkGolden">
    <color rgba="0.4 0.4 0.3 1.0"/>
  </material>
  <material name="Schunk/Blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="Schunk/Red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/Green">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="invisible">
    <color rgba="0.0 0.0 0.0 0.0"/>
```

```
</material>
<gazebo>
  <plugin filename="libgazebo_ros_control.so" name="gazebo_ros_control">
    <robotNamespace>/rescuer</robotNamespace>
    <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
  </plugin>
  <plugin filename="libgazebo_ros_joint_pose_trajectory.so" name="
      gazebo_ros_joint_pose_trajectory">
            </plugin>
</gazebo>
<!--xacro:include filename="$(find rescuer_description)/urdf/sdh/sdh.urdf.xacro"
    /-->
<!--xacro:include filename="$(find rescuer_description)/urdf/fake_gripper/
    fake_gripper.urdf.xacro" /-->
<link name="base_footprint">
  </link>
<joint name="base_joint" type="fixed">
  <origin rpy="0 0 0" xyz="0 0 0.08"/>
  <parent link="base_footprint"/>
  <child link="rover_base_link"/>
</joint>
<joint name="rover_base_joint" type="fixed">
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <parent link="rover_base_link"/>
  <child link="rover_chains_link"/>
</joint>
<link name="rover_base_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-0.002618 0.000006 0.204921"/>
    <!--mass value="0.001"/-->
    <mass value="200"/>
    <inertia ixx="2.7532e-02" ixy="1.5577e-06" ixz="-7.8932e-04" iyy="5.8836e-02"
        iyz="-8.9012e-07" izz="7.1819e-02"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/rover/rescuer_uji_3n.
          dae"/>
    </geometry>
    <material name="Rover/Red"/>
  </visual>
  <collision>
    <!--origin xyz="0 0 0.175" rpy="0 0 0" /-->
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <!--box size="1.20 0.78 0.50" /-->
      <mesh filename="package://rescuer_description/meshes/rover/rescuer_uji_3n.
          dae"/>
    </geometry>
  </collision>
</link>
<link name="rover_chains_link">
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <mass value="10"/>
    <inertia ixx="1.0" ixy="0" ixz="0" iyy="1.0" iyz="0" izz="1.0"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/rover/
          rescuer_uji_cadena3.dae"/>
    </geometry>
    <material name="Rover/Black"/>
  </visual>
```

```
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/rover/
            rescuer_uji_cadena3.dae"/>
        <!--box size="0.01 0.01 0.01" /-->
      </geometry>
    </collision>
  </link>
  <gazebo reference="rover_base_link">
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="rover_chains_link">
    <material value="Gazebo/Black"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <joint name="arm_0_joint" type="fixed">
    <origin rpy="1.5708 1.5708 0" xyz="0.5 0.085 0.5"/>
    <parent link="rover_base_link"/>
    <child link="arm_0_link"/>
  </joint>
  <link name="arm_0_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <mass value="2.0"/>
      <inertia ixx="10" ixy="0" ixz="0" iyy="10" iyz="0" izz="10"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <!--cylinder radius="0.045" length="0.18" /-->
        <cylinder length="0.066" radius="0.045"/>
      </geometry>
      <material name="Schunk/Red"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <!--box size="0.01 0.01 0.01" /-->
        <!--cylinder radius="0.045" length="0.18" /-->
        <cylinder length="0.066" radius="0.045"/>
      </geometry>
    </collision>
  </link>
  <joint name="arm_1_joint" type="revolute">
    <origin rpy="0 0 1.5707963268" xyz="0 0 -0.240"/>
    <parent link="arm_0_link"/>
    <child link="arm_1_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <!--dynamics damping="${damping}" friction="${friction}"/-->
    <dynamics damping="0"/>
    <limit effort="370" lower="-3.31" upper="3.31" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="50" soft_lower_limit="-3.3"
        soft_upper_limit="3.3"/>
  </joint>
  <link name="arm_1_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.000003 -0.009599 0.141766"/>
      <!--mass value="1.29364"/-->
      <mass value="4.89364"/>
      <!--inertia ixx="0.000012" ixy="0" ixz="0" iyy="0.000012" iyz="0.000002" izz=
          "0.000005" /-->
```

```
      <inertia ixx="5.7745e-03" ixy="-2.6628e-07" ixz="-1.6643e-08" iyy="5.4967e-03
          " iyz="-1.0778e-03" izz="2.5060e-03"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular1.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
    </visual>
    <collision>
      <!--origin xyz="0 0 0.15" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular1.dae"/
            >
        <!--cylinder radius="0.065" length="0.30" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_2_joint" type="revolute">
    <origin rpy="1.5707963268 0 0" xyz="0 0 0"/>
    <parent link="arm_1_link"/>
    <child link="arm_2_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="370" lower="-2.61" upper="2.61" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="50" soft_lower_limit="-2.6"
        soft_upper_limit="2.6"/>
  </joint>
  <link name="arm_2_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-0.000001 -0.071960 -0.043237"/>
      <!--mass value="1.68311"/-->
      <mass value="5.28311"/>
      <!--inertia ixx="0.00005" ixy="0" ixz="0" iyy="0.000011" iyz="-0.000011" izz=
          "0.00005" /-->
      <inertia ixx="2.0603e-02" ixy="1.9586e-08" ixz="1.9586e-08" iyy="4.0753e-03"
          iyz="-3.9778e-03" izz="1.8736e-02"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular2.dae"/
            >
      </geometry>
      <material name="Schunk/LightGrey"/>
    </visual>
    <collision>
      <!--origin xyz="0 -0.25 0" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular2.dae"/
            >
        <!--box size="0.12 0.50 0.18" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_3_joint" type="revolute">
    <origin rpy="-1.5707963268 0 0" xyz="0 -0.395 0"/>
    <parent link="arm_2_link"/>
    <child link="arm_3_link"/>
    <axis xyz="0 1 0"/>
    <calibration rising="0"/>
```

```
    <dynamics damping="0" friction="1.0"/>
    <limit effort="176" lower="-2.7" upper="2.7" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="-2.69"
        soft_upper_limit="2.69"/>
  </joint>
  <link name="arm_3_link">
    <inertial>
      <!--origin xyz="0 -0.395 0" rpy="-1.5708 0 0"/-->
      <origin rpy="0 0 0" xyz="0 -0.013305 -0.006775"/>
      <!--mass value="2.1"/-->
      <mass value="4.1"/>
      <!--inertia ixx="0.0000035" ixy="0" ixz="0" iyy="0.0000026" iyz="-0.00000025"
          izz="0.0000031" /-->
      <inertia ixx="0.0025313" ixy="0" ixz="0" iyy="0.0018840" iyz="-0.0001840" izz
          ="0.0022512"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular3.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular3.dae"/
            >
        <!--box size="0.11 0.14 0.14" /-->
        <!--box size="0.01 0.01 0.01" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_4_joint" type="revolute">
    <origin rpy="1.5707963268 0 0" xyz="0 0 -0.2715"/>
    <parent link="arm_3_link"/>
    <child link="arm_4_link"/>
    <axis xyz="0 1 0"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="176" lower="-3.14159265359" upper="3.14159265359" velocity="24.0
        "/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        -3.13159265359" soft_upper_limit="3.13159265359"/>
  </joint>
  <link name="arm_4_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.000015 0.106227 -0.008144"/>
      <!--mass value="1.68311"/-->
      <mass value="2.88311"/>
      <!--inertia ixx="0.0000024" ixy="0" ixz="0" iyy="0.00000093" iyz="-0.00000048
          " izz="0.0000023" /-->
      <inertia ixx="3.2829e-03" ixy="1.5715e-06" ixz="7.4912e-07" iyy="1.2719e-03"
          iyz="-6.5480e-04" izz="3.1497e-03"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular4.dae"/
            >
      </geometry>
      <material name="Schunk/LightGrey"/>
    </visual>
    <collision>
      <!--origin xyz="0 0.11 0" rpy="0 0 0" /-->
```

```
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular4.dae"/
            >
        <!--box size="0.10 0.22 0.10" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_5_joint" type="revolute">
    <!--origin xyz="0 0 0" rpy="${pi/2} ${-pi/2} 0" /-->
    <origin rpy="1.5707963268 1.5707963268 0" xyz="0 0 0"/>
    <parent link="arm_4_link"/>
    <child link="arm_5_link"/>
    <!--axis xyz="1 0 0" /-->
    <axis xyz="1 0 0"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <!--limit effort="41.6" velocity="0.4363" lower="-1.57079" upper="1.57079" /-->
    <!--limit effort="20" velocity="2" /-->
    <!--safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        ${-1.57079 + 0.01}" soft_upper_limit="${1.57079 - 0.01}" /-->
    <limit effort="176" lower="-1.56" upper="1.56" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="-1.55"
        soft_upper_limit="1.55"/>
  </joint>
  <link name="arm_5_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.016561 0.000000 0.008731"/>
      <!--mass value="0.807"/-->
      <mass value="1.807"/>
      <!--inertia ixx="0.00000074" ixy="0" ixz="-0.0000001" iyy="0.000001" iyz="
          -0.0000001" izz="0.00000089" /-->
      <inertia ixx="1.1816e-03" ixy="2.1927e-08" ixz="-1.6663e-04" iyy="1.7339e-03"
          iyz="-1.4618e-08" izz="1.4334e-03"/>
    </inertial>
    <visual>
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <origin rpy="0 0 3.14159265359" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular5.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
    </visual>
    <collision>
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <origin rpy="0 0 3.14159265359" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular5.dae"/
            >
        <!--box size="0.11 0.08 0.11" /-->
        <!--box size="0.01 0.01 0.01" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_6_joint" type="revolute">
    <origin rpy="0 0 0" xyz="0 0 0.1847"/>
    <parent link="arm_5_link"/>
    <child link="arm_6_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="20.1" lower="-3.34159265359" upper="3.34159265359" velocity="
        24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        -3.33159265359" soft_upper_limit="3.33159265359"/>
```

```
</joint>
<link name="arm_6_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-0.000000 -0.000000 -0.054637"/>
    <!--mass value="0.819"/-->
    <mass value="1"/>
    <!--inertia ixx="0.00000074" ixy="0" ixz="-0.0000001" iyy="0.000001" iyz="
        -0.0000001" izz="0.00000089" /-->
    <inertia ixx="0.0014132" ixy="0" ixz="0" iyy="0.0014132" iyz="0" izz="
        0.0006018"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/modular/modular6.dae"/
          >
    </geometry>
    <material name="Schunk/LightGrey"/>
  </visual>
  <collision>
    <!--origin xyz="0 0 -0.055" rpy="0 0 0" /-->
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/modular/modular6.dae"/
          >
      <!--box size="0.09 0.09 0.11" /-->
    </geometry>
  </collision>
</link>
<gazebo reference="arm_0_link">
  <material value="Gazebo/Red"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>false</selfCollide>
</gazebo>
<gazebo reference="arm_1_link">
  <material value="Gazebo/Green"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<gazebo reference="arm_2_link">
  <material value="Gazebo/Red"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<gazebo reference="arm_3_link">
  <material value="Gazebo/Green"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<gazebo reference="arm_4_link">
  <material value="Gazebo/Red"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<gazebo reference="arm_5_link">
  <material value="Gazebo/Green"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<gazebo reference="arm_6_link">
  <material value="Gazebo/Red"/>
  <turnGravityOff>false</turnGravityOff>
  <selfCollide>true</selfCollide>
</gazebo>
<transmission name="arm_1_trans">
  <type>transmission_interface/SimpleTransmission</type>
```

```
    <joint name="arm_1_joint"/>
    <actuator name="arm_1_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_2_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_2_joint"/>
    <actuator name="arm_2_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_3_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_3_joint"/>
    <actuator name="arm_3_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_4_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_4_joint"/>
    <actuator name="arm_4_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_5_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_5_joint"/>
    <actuator name="arm_5_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_6_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_6_joint"/>
    <actuator name="arm_6_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <joint name="hand_base_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <parent link="arm_6_link"/>
    <child link="hand_base_link"/>
  </joint>
  <link name="hand_base_link">
    <inertial>
      <mass value="1.0"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/palm_282.dae"/
            >
      </geometry>
      <material name="Grey">
        <color rgba="0.75 0.75 0.75 1.0"/>
      </material>
```

```
    </visual>
    <collision>
      <origin rpy="0 0 0 " xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/palm_282.dae"/
            >
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <link name="hand_finger_31_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="-0.046 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="-0.04 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/knuckle_fixed.
            dae"/>
      </geometry>
      <material name="Black"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="-0.04 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/knuckle_fixed.
            dae"/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j31_joint" type="fixed">
    <parent link="hand_base_link"/>
    <child link="hand_finger_31_link"/>
    <origin rpy="0 0 -1.570796327" xyz="0 0 0.0252"/>
  </joint>
  <link name="hand_finger_32_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j32_joint" type="revolute">
    <parent link="hand_finger_31_link"/>
    <child link="hand_finger_32_link"/>
    <origin rpy="1.5708 -8.8281E-17 0" xyz="-0.05 0 0.0339"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
```

```
</joint>
<transmission name="hand_j32_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j32_joint"/>
  <actuator name="hand_j32">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_33_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j33_joint" type="revolute">
  <parent link="hand_finger_32_link"/>
  <child link="hand_finger_33_link"/>
  <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j33_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j33_joint"/>
  <actuator name="hand_j33">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_11_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="-0.046 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
    <material name="Green"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
```

```
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
  </collision>
</link>
<joint name="hand_j11_joint" type="revolute">
  <parent link="hand_base_link"/>
  <child link="hand_finger_11_link"/>
  <origin rpy="0 0 1.5708" xyz="-0.025 0 0.0252"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0" upper="3.1416" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j11_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j11_joint"/>
  <actuator name="hand_j11">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_12_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j12_joint" type="revolute">
  <parent link="hand_finger_11_link"/>
  <child link="hand_finger_12_link"/>
  <origin rpy="1.5708 0 0" xyz="-0.05 0 0.0339"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j12_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j12_joint"/>
  <actuator name="hand_j12">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_13_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
```

```
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
            "/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
            "/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j13_joint" type="revolute">
    <parent link="hand_finger_12_link"/>
    <child link="hand_finger_13_link"/>
    <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j13_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j13_joint"/>
    <actuator name="hand_j13">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <link name="hand_finger_21_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="-0.046 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
      </geometry>
      <material name="Green"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
      </geometry>
    </collision>
  </link>
  <joint name="hand_j21_joint" type="revolute">
    <parent link="hand_base_link"/>
    <child link="hand_finger_21_link"/>
    <origin rpy="0 0 1.5708" xyz="0.025 0 0.0252"/>
    <axis xyz="0 0 1"/>
    <limit effort="30.0" lower="0" upper="3.1416" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j21_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j21_joint"/>
    <actuator name="hand_j21">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
```

```
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_22_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j22_joint" type="revolute">
  <parent link="hand_finger_21_link"/>
  <child link="hand_finger_22_link"/>
  <origin rpy="1.5708 -8.8281E-17 0" xyz="-0.05 0 0.0339"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j22_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j22_joint"/>
  <actuator name="hand_j22">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_23_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j23_joint" type="revolute">
```

```
    <parent link="hand_finger_22_link"/>
    <child link="hand_finger_23_link"/>
    <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j23_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j23_joint"/>
    <actuator name="hand_j23">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <gazebo reference="hand_base_link">
    <material>Gazebo/White</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_31_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_32_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_33_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_11_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_12_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_13_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_21_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_22_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_23_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <!--xacro:rescuer_sdh_urdf parent="arm_6_link" name="hand">
        <origin xyz="0 0 0" rpy="0 0 0" />
    </xacro:rescuer_sdh_urdf-->
  <!--xacro:rescuer_fake_gripper_urdf>
        <origin xyz="0 0 0" rpy="0 0 0" />
    </xacro:rescuer_fake_gripper_urdf-->
  <joint name="camera_rgb_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.635 -0.028 0.355"/>
    <parent link="rover_base_link"/>
    <child link="camera_rgb_frame"/>
```

```
</joint>
<link name="camera_rgb_frame">
  <inertial>
    <mass value="0.001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_rgb_optical_joint" type="fixed">
  <origin rpy="-1.5707963268 0 -1.5707963268" xyz="0 0 0"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_rgb_optical_frame"/>
</joint>
<link name="camera_rgb_optical_frame">
  <inertial>
    <mass value="0.001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_joint" type="fixed">
  <origin rpy="0 0 0" xyz="-0.031 0.0125 -0.016"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_link"/>
</joint>
<link name="camera_link">
  <visual>
    <!--origin xyz="0 0 0" rpy="0 0 ${pi/2}"/-->
    <origin rpy="0 0 0" xyz="0.032 0 0.01905"/>
    <geometry>
      <box size="0.064 0.121 0.0381"/>
      <!--mesh filename="package://rescuer_description/meshes/kinect/kinect.dae"/
          -->
    </geometry>
    <material name="Gazebo/Blue"/>
  </visual>
  <collision>
    <!--origin xyz="0.0 0.0 0.0" rpy="0 0 0"/-->
    <origin rpy="0 0 0" xyz="0.032 0 0.01905"/>
    <geometry>
      <box size="0.064 0.121 0.0381"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="0.001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_depth_joint" type="fixed">
  <!--origin xyz="0 0.028 0" rpy="${-pi/2} 0 ${-pi/2}" /-->
  <origin rpy="0 0 0" xyz="0 0.028 0"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_depth_frame"/>
</joint>
<link name="camera_depth_frame">
  <inertial>
    <mass value="0.0001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
```

```
<joint name="camera_depth_optical_joint" type="fixed">
  <!--origin xyz="0 0 0" rpy="0 0 0" /-->
  <origin rpy="-1.5707963268 0 -1.5707963268" xyz="0 0 0"/>
  <parent link="camera_depth_frame"/>
  <child link="camera_depth_optical_frame"/>
</joint>
<link name="camera_depth_optical_frame">
  <inertial>
    <mass value="0.0001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<gazebo reference="camera_link">
  <material value="Gazebo/Blue"/>
  <turnGravityOff>false</turnGravityOff>
  <sensor name="openni_camera_camera" type="depth">
    <always_on>1</always_on>
    <visualize>true</visualize>
    <camera>
      <horizontal_fov>1.047</horizontal_fov>
      <image>
        <width>640</width>
        <height>480</height>
        <format>R8G8B8</format>
      </image>
      <depth_camera>

            </depth_camera>
      <clip>
        <near>0.1</near>
        <far>100</far>
      </clip>
    </camera>
    <plugin filename="libgazebo_ros_openni_kinect.so" name="
        camera_link_controller">
      <!--robotNamespace>/rescuer</robotNamespace-->
      <baseline>0.2</baseline>
      <alwaysOn>true</alwaysOn>
      <updateRate>1.0</updateRate>
      <cameraName>rescuer/camera</cameraName>
      <imageTopicName>depth_registered/image_raw</imageTopicName>
      <cameraInfoTopicName>depth_registered/camera_info</cameraInfoTopicName>
      <depthImageTopicName>depth_registered/image_raw</depthImageTopicName>
      <depthImageInfoTopicName>depth_registered/camera_info</
          depthImageInfoTopicName>
      <pointCloudTopicName>depth_registered/points</pointCloudTopicName>
      <frameName>camera_depth_optical_frame</frameName>
      <pointCloudCutoff>0.5</pointCloudCutoff>
      <distortionK1>0</distortionK1>
      <distortionK2>0</distortionK2>
      <distortionK3>0</distortionK3>
      <distortionT1>0</distortionT1>
      <distortionT2>0</distortionT2>
      <CxPrime>0</CxPrime>
      <Cx>0</Cx>
      <Cy>0</Cy>
      <focalLength>0</focalLength>
      <hackBaseline>0</hackBaseline>
      <depthImageCameraInfoTopicName>depthcamerainfo</
          depthImageCameraInfoTopicName>
    </plugin>
  </sensor>
</gazebo>
<gazebo reference="camera_depth_frame">
```

```
    <material value="Gazebo/Black"/>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="camera_depth_optical_frame">
    <material value="Gazebo/Black"/>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <!--xacro:rescuer_fake_wheels_urdf >
    </xacro:rescuer_fake_wheels_urdf-->
</robot>
```

C.2. Descripción completa simulador en urdf

```
<?xml version="1.0" ?>
<!--
    ================================================================================
     -->
<!-- |    This document was autogenerated by xacro from rescuer.urdf.simulator.
    xacro    | -->
<!-- |    EDITING THIS FILE BY HAND IS NOT RECOMMENDED
                                          | -->
<!--
    ================================================================================
     -->
<robot name="rescuer" xmlns:xacro="http://www.ros.org/wiki/xacro">
  <material name="Rover/Red">
    <color rgba="1.0 0.0 0.0 1.0"/>
  </material>
  <material name="Rover/Black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/LightGrey">
    <color rgba="0.7 0.7 0.7 1.0"/>
  </material>
  <material name="Schunk/DarkGrey">
    <color rgba="0.4 0.4 0.4 1.0"/>
  </material>
  <material name="Schunk/Black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/DarkGolden">
    <color rgba="0.4 0.4 0.3 1.0"/>
  </material>
  <material name="Schunk/Blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="Schunk/Red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>
  <material name="Schunk/Green">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="invisible">
    <color rgba="0.0 0.0 0.0 0.0"/>
  </material>
  <gazebo>
    <plugin filename="libgazebo_ros_control.so" name="gazebo_ros_control">
      <robotNamespace>/rescuer</robotNamespace>
      <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
    </plugin>
    <plugin filename="libgazebo_ros_joint_pose_trajectory.so" name="
        gazebo_ros_joint_pose_trajectory">
              </plugin>
  </gazebo>
```

```
<!--xacro:include filename="$(find rescuer_description)/urdf/sdh/sdh.urdf.xacro"
    /-->
<!--xacro:include filename="$(find rescuer_description)/urdf/fake_gripper/
    fake_gripper.urdf.xacro" /-->
<link name="base_footprint">
  </link>
<joint name="base_joint" type="fixed">
  <origin rpy="0 0 0" xyz="0 0 0.08"/>
  <parent link="base_footprint"/>
  <child link="rover_base_link"/>
</joint>
<joint name="rover_base_joint" type="fixed">
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <parent link="rover_base_link"/>
  <child link="rover_chains_link"/>
</joint>
<link name="rover_base_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-0.002618 0.000006 0.204921"/>
    <!--mass value="0.001"/-->
    <mass value="200"/>
    <inertia ixx="2.7532e-02" ixy="1.5577e-06" ixz="-7.8932e-04" iyy="5.8836e-02"
         iyz="-8.9012e-07" izz="7.1819e-02"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/rover/rescuer_uji_3n.
          dae"/>
    </geometry>
    <material name="Rover/Red"/>
  </visual>
  <collision>
    <!--origin xyz="0 0 0.175" rpy="0 0 0" /-->
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <!--box size="1.20 0.78 0.50" /-->
      <mesh filename="package://rescuer_description/meshes/rover/rescuer_uji_3n.
          dae"/>
    </geometry>
  </collision>
</link>
<link name="rover_chains_link">
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <mass value="10"/>
    <inertia ixx="1.0" ixy="0" ixz="0" iyy="1.0" iyz="0" izz="1.0"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/rover/
          rescuer_uji_cadena3.dae"/>
    </geometry>
    <material name="Rover/Black"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/rover/
          rescuer_uji_cadena3.dae"/>
      <!--box size="0.01 0.01 0.01" /-->
    </geometry>
  </collision>
</link>
<gazebo reference="rover_base_link">
```

```
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="rover_chains_link">
    <material value="Gazebo/Black"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <joint name="arm_0_joint" type="fixed">
    <origin rpy="1.5708 1.5708 0" xyz="0.5 0.085 0.5"/>
    <parent link="rover_base_link"/>
    <child link="arm_0_link"/>
  </joint>
  <link name="arm_0_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <mass value="2.0"/>
      <inertia ixx="10" ixy="0" ixz="0" iyy="10" iyz="0" izz="10"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <!--cylinder radius="0.045" length="0.18" /-->
        <cylinder length="0.066" radius="0.045"/>
      </geometry>
      <material name="Schunk/Red"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <!--box size="0.01 0.01 0.01" /-->
        <!--cylinder radius="0.045" length="0.18" /-->
        <cylinder length="0.066" radius="0.045"/>
      </geometry>
    </collision>
  </link>
  <joint name="arm_1_joint" type="revolute">
    <origin rpy="0 0 1.5707963268" xyz="0 0 -0.240"/>
    <parent link="arm_0_link"/>
    <child link="arm_1_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <!--dynamics damping="${damping}" friction="${friction}"/-->
    <dynamics damping="0"/>
    <limit effort="370" lower="-3.31" upper="3.31" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="50" soft_lower_limit="-3.3"
        soft_upper_limit="3.3"/>
  </joint>
  <link name="arm_1_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.000003 -0.009599 0.141766"/>
      <!--mass value="1.29364"/-->
      <mass value="4.89364"/>
      <!--inertia ixx="0.000012" ixy="0" ixz="0" iyy="0.000012" iyz="0.000002" izz=
          "0.000005" /-->
      <inertia ixx="5.7745e-03" ixy="-2.6628e-07" ixz="-1.6643e-08" iyy="5.4967e-03
          " iyz="-1.0778e-03" izz="2.5060e-03"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular1.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
```

```
    </visual>
    <collision>
      <!--origin xyz="0 0 0.15" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular1.dae"/
            >
        <!--cylinder radius="0.065" length="0.30" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_2_joint" type="revolute">
    <origin rpy="1.5707963268 0 0" xyz="0 0 0"/>
    <parent link="arm_1_link"/>
    <child link="arm_2_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="370" lower="-2.61" upper="2.61" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="50" soft_lower_limit="-2.6"
        soft_upper_limit="2.6"/>
  </joint>
  <link name="arm_2_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-0.000001 -0.071960 -0.043237"/>
      <!--mass value="1.68311"/-->
      <mass value="5.28311"/>
      <!--inertia ixx="0.00005" ixy="0" ixz="0" iyy="0.000011" iyz="-0.000011" izz=
          "0.00005" /-->
      <inertia ixx="2.0603e-02" ixy="1.9586e-08" ixz="1.9586e-08" iyy="4.0753e-03"
          iyz="-3.9778e-03" izz="1.8736e-02"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular2.dae"/
            >
      </geometry>
      <material name="Schunk/LightGrey"/>
    </visual>
    <collision>
      <!--origin xyz="0 -0.25 0" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular2.dae"/
            >
        <!--box size="0.12 0.50 0.18" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_3_joint" type="revolute">
    <origin rpy="-1.5707963268 0 0" xyz="0 -0.395 0"/>
    <parent link="arm_2_link"/>
    <child link="arm_3_link"/>
    <axis xyz="0 1 0"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="176" lower="-2.7" upper="2.7" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="-2.69"
        soft_upper_limit="2.69"/>
  </joint>
  <link name="arm_3_link">
    <inertial>
      <!--origin xyz="0 -0.395 0" rpy="-1.5708 0 0"/-->
      <origin rpy="0 0 0" xyz="0 -0.013305 -0.006775"/>
      <!--mass value="2.1"/-->
```

```
      <mass value="4.1"/>
      <!--inertia ixx="0.0000035" ixy="0" ixz="0" iyy="0.0000026" iyz="-0.00000025"
          izz="0.0000031" /-->
      <inertia ixx="0.0025313" ixy="0" ixz="0" iyy="0.0018840" iyz="-0.0001840" izz
          ="0.0022512"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular3.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular3.dae"/
            >
        <!--box size="0.11 0.14 0.14" /-->
        <!--box size="0.01 0.01 0.01" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_4_joint" type="revolute">
    <origin rpy="1.5707963268 0 0" xyz="0 0 -0.2715"/>
    <parent link="arm_3_link"/>
    <child link="arm_4_link"/>
    <axis xyz="0 1 0"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="176" lower="-3.14159265359" upper="3.14159265359" velocity="24.0
        "/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        -3.13159265359" soft_upper_limit="3.13159265359"/>
  </joint>
  <link name="arm_4_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.000015 0.106227 -0.008144"/>
      <!--mass value="1.68311"/-->
      <mass value="2.88311"/>
      <!--inertia ixx="0.0000024" ixy="0" ixz="0" iyy="0.00000093" iyz="-0.00000048
          " izz="0.0000023" /-->
      <inertia ixx="3.2829e-03" ixy="1.5715e-06" ixz="7.4912e-07" iyy="1.2719e-03"
          iyz="-6.5480e-04" izz="3.1497e-03"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular4.dae"/
            >
      </geometry>
      <material name="Schunk/LightGrey"/>
    </visual>
    <collision>
      <!--origin xyz="0 0.11 0" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular4.dae"/
            >
        <!--box size="0.10 0.22 0.10" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_5_joint" type="revolute">
    <!--origin xyz="0 0 0" rpy="${pi/2} ${-pi/2} 0" /-->
```

```
    <origin rpy="1.5707963268 1.5707963268 0" xyz="0 0 0"/>
    <parent link="arm_4_link"/>
    <child link="arm_5_link"/>
    <!--axis xyz="1 0 0" /-->
    <axis xyz="1 0 0"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <!--limit effort="41.6" velocity="0.4363" lower="-1.57079" upper="1.57079" /-->
    <!--limit effort="20" velocity="2" /-->
    <!--safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        ${-1.57079 + 0.01}" soft_upper_limit="${1.57079 - 0.01}" /-->
    <limit effort="176" lower="-1.56" upper="1.56" velocity="24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="-1.55"
        soft_upper_limit="1.55"/>
  </joint>
  <link name="arm_5_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.016561 0.000000 0.008731"/>
      <!--mass value="0.807"/-->
      <mass value="1.807"/>
      <!--inertia ixx="0.00000074" ixy="0" ixz="-0.0000001" iyy="0.000001" iyz="
          -0.0000001" izz="0.00000089" /-->
      <inertia ixx="1.1816e-03" ixy="2.1927e-08" ixz="-1.6663e-04" iyy="1.7339e-03"
           iyz="-1.4618e-08" izz="1.4334e-03"/>
    </inertial>
    <visual>
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <origin rpy="0 0 3.14159265359" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular5.dae"/
            >
      </geometry>
      <material name="Schunk/Blue"/>
    </visual>
    <collision>
      <!--origin xyz="0 0 0" rpy="0 0 0" /-->
      <origin rpy="0 0 3.14159265359" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular5.dae"/
            >
        <!--box size="0.11 0.08 0.11" /-->
        <!--box size="0.01 0.01 0.01" /-->
      </geometry>
    </collision>
  </link>
  <joint name="arm_6_joint" type="revolute">
    <origin rpy="0 0 0" xyz="0 0 0.1847"/>
    <parent link="arm_5_link"/>
    <child link="arm_6_link"/>
    <axis xyz="0 0 1"/>
    <calibration rising="0"/>
    <dynamics damping="0" friction="1.0"/>
    <limit effort="20.1" lower="-3.34159265359" upper="3.34159265359" velocity="
        24.0"/>
    <safety_controller k_position="20" k_velocity="25" soft_lower_limit="
        -3.33159265359" soft_upper_limit="3.33159265359"/>
  </joint>
  <link name="arm_6_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-0.000000 -0.000000 -0.054637"/>
      <!--mass value="0.819"/-->
      <mass value="1"/>
      <!--inertia ixx="0.00000074" ixy="0" ixz="-0.0000001" iyy="0.000001" iyz="
          -0.0000001" izz="0.00000089" /-->
      <inertia ixx="0.0014132" ixy="0" ixz="0" iyy="0.0014132" iyz="0" izz="
          0.0006018"/>
```

```
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular6.dae"/
            >
      </geometry>
      <material name="Schunk/LightGrey"/>
    </visual>
    <collision>
      <!--origin xyz="0 0 -0.055" rpy="0 0 0" /-->
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/modular/modular6.dae"/
            >
        <!--box size="0.09 0.09 0.11" /-->
      </geometry>
    </collision>
  </link>
  <gazebo reference="arm_0_link">
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>false</selfCollide>
  </gazebo>
  <gazebo reference="arm_1_link">
    <material value="Gazebo/Green"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="arm_2_link">
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="arm_3_link">
    <material value="Gazebo/Green"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="arm_4_link">
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="arm_5_link">
    <material value="Gazebo/Green"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <gazebo reference="arm_6_link">
    <material value="Gazebo/Red"/>
    <turnGravityOff>false</turnGravityOff>
    <selfCollide>true</selfCollide>
  </gazebo>
  <transmission name="arm_1_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_1_joint"/>
    <actuator name="arm_1_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="arm_2_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_2_joint"/>
    <actuator name="arm_2_motor">
```

```
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
<transmission name="arm_3_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_3_joint"/>
  <actuator name="arm_3_motor">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
<transmission name="arm_4_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_4_joint"/>
  <actuator name="arm_4_motor">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
<transmission name="arm_5_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_5_joint"/>
  <actuator name="arm_5_motor">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
<transmission name="arm_6_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_6_joint"/>
  <actuator name="arm_6_motor">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
<joint name="hand_base_joint" type="fixed">
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <parent link="arm_6_link"/>
  <child link="hand_base_link"/>
</joint>
<link name="hand_base_link">
  <inertial>
    <mass value="1.0"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/palm_282.dae"/
          >
    </geometry>
    <material name="Grey">
      <color rgba="0.75 0.75 0.75 1.0"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0 " xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/palm_282.dae"/
          >
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
```

```
<link name="hand_finger_31_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="-0.046 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="-0.04 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle_fixed.
          dae"/>
    </geometry>
    <material name="Black"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="-0.04 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle_fixed.
          dae"/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j31_joint" type="fixed">
  <parent link="hand_base_link"/>
  <child link="hand_finger_31_link"/>
  <origin rpy="0 0 -1.570796327" xyz="0 0 0.0252"/>
</joint>
<link name="hand_finger_32_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j32_joint" type="revolute">
  <parent link="hand_finger_31_link"/>
  <child link="hand_finger_32_link"/>
  <origin rpy="1.5708 -8.8281E-17 0" xyz="-0.05 0 0.0339"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j32_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j32_joint"/>
  <actuator name="hand_j32">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
```

```
<link name="hand_finger_33_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <material name="Blue"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j33_joint" type="revolute">
  <parent link="hand_finger_32_link"/>
  <child link="hand_finger_33_link"/>
  <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j33_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j33_joint"/>
  <actuator name="hand_j33">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_11_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="-0.046 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
    <material name="Green"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
  </collision>
</link>
<joint name="hand_j11_joint" type="revolute">
  <parent link="hand_base_link"/>
  <child link="hand_finger_11_link"/>
  <origin rpy="0 0 1.5708" xyz="-0.025 0 0.0252"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0" upper="3.1416" velocity="2.0"/>
```

```
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j11_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j11_joint"/>
    <actuator name="hand_j11">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <link name="hand_finger_12_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j12_joint" type="revolute">
    <parent link="hand_finger_11_link"/>
    <child link="hand_finger_12_link"/>
    <origin rpy="1.5708 0 0" xyz="-0.05 0 0.0339"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j12_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j12_joint"/>
    <actuator name="hand_j12">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <link name="hand_finger_13_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
            "/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
```

```
      <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
          "/>
    </geometry>
    <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
  </collision>
</link>
<joint name="hand_j13_joint" type="revolute">
  <parent link="hand_finger_12_link"/>
  <child link="hand_finger_13_link"/>
  <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
  <axis xyz="0 0 -1"/>
  <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j13_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j13_joint"/>
  <actuator name="hand_j13">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_21_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="-0.046 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
    <material name="Green"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://rescuer_description/meshes/barrett/knuckle.dae"/>
    </geometry>
  </collision>
</link>
<joint name="hand_j21_joint" type="revolute">
  <parent link="hand_base_link"/>
  <child link="hand_finger_21_link"/>
  <origin rpy="0 0 1.5708" xyz="0.025 0 0.0252"/>
  <axis xyz="0 0 1"/>
  <limit effort="30.0" lower="0" upper="3.1416" velocity="2.0"/>
  <dynamics damping="100.0" friction="1.0"/>
</joint>
<transmission name="hand_j21_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="hand_j21_joint"/>
  <actuator name="hand_j21">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
    <motorTorqueConstant>1</motorTorqueConstant>
  </actuator>
</transmission>
<link name="hand_finger_22_link">
  <inertial>
    <mass value="0.1"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
  </inertial>
```

```
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger.dae"/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j22_joint" type="revolute">
    <parent link="hand_finger_21_link"/>
    <child link="hand_finger_22_link"/>
    <origin rpy="1.5708 -8.8281E-17 0" xyz="-0.05 0 0.0339"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="2.44" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j22_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j22_joint"/>
    <actuator name="hand_j22">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <link name="hand_finger_23_link">
    <inertial>
      <mass value="0.1"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
            "/>
      </geometry>
      <material name="Blue"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://rescuer_description/meshes/barrett/finger_tip.dae
            "/>
      </geometry>
      <contact_coefficients kd="1.0" kp="1000.0" mu="0"/>
    </collision>
  </link>
  <joint name="hand_j23_joint" type="revolute">
    <parent link="hand_finger_22_link"/>
    <child link="hand_finger_23_link"/>
    <origin rpy="0 0 0" xyz="-0.069936 0.003 0"/>
    <axis xyz="0 0 -1"/>
    <limit effort="30.0" lower="0.0" upper="0.84" velocity="2.0"/>
    <dynamics damping="100.0" friction="1.0"/>
  </joint>
  <transmission name="hand_j23_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="hand_j23_joint"/>
```

```
    <actuator name="hand_j23">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
      <motorTorqueConstant>1</motorTorqueConstant>
    </actuator>
  </transmission>
  <gazebo reference="hand_base_link">
    <material>Gazebo/White</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_31_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_32_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_33_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_11_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_12_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_13_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_21_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_22_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <gazebo reference="hand_finger_23_link">
    <material>Gazebo/Grey</material>
    <turnGravityOff>false</turnGravityOff>
  </gazebo>
  <!--xacro:rescuer_sdh_urdf parent="arm_6_link" name="hand">
        <origin xyz="0 0 0" rpy="0 0 0" />
    </xacro:rescuer_sdh_urdf-->
  <!--xacro:rescuer_fake_gripper_urdf>
        <origin xyz="0 0 0" rpy="0 0 0" />
    </xacro:rescuer_fake_gripper_urdf-->
  <joint name="camera_rgb_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.635 -0.028 0.355"/>
    <parent link="rover_base_link"/>
    <child link="camera_rgb_frame"/>
  </joint>
  <link name="camera_rgb_frame">
    <inertial>
      <mass value="0.001"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
          />
    </inertial>
  </link>
  <joint name="camera_rgb_optical_joint" type="fixed">
```

```
  <origin rpy="-1.5707963268 0 -1.5707963268" xyz="0 0 0"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_rgb_optical_frame"/>
</joint>
<link name="camera_rgb_optical_frame">
  <inertial>
    <mass value="0.001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_joint" type="fixed">
  <origin rpy="0 0 0" xyz="-0.031 0.0125 -0.016"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_link"/>
</joint>
<link name="camera_link">
  <visual>
    <!--origin xyz="0 0 0" rpy="0 0 ${pi/2}"/-->
    <origin rpy="0 0 0" xyz="0.032 0 0.01905"/>
    <geometry>
      <box size="0.064 0.121 0.0381"/>
      <!--mesh filename="package://rescuer_description/meshes/kinect/kinect.dae"/
          -->
    </geometry>
    <material name="Gazebo/Blue"/>
  </visual>
  <collision>
    <!--origin xyz="0.0 0.0 0.0" rpy="0 0 0"/-->
    <origin rpy="0 0 0" xyz="0.032 0 0.01905"/>
    <geometry>
      <box size="0.064 0.121 0.0381"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="0.001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_depth_joint" type="fixed">
  <!--origin xyz="0 0.028 0" rpy="${-pi/2} 0 ${-pi/2}" /-->
  <origin rpy="0 0 0" xyz="0 0.028 0"/>
  <parent link="camera_rgb_frame"/>
  <child link="camera_depth_frame"/>
</joint>
<link name="camera_depth_frame">
  <inertial>
    <mass value="0.0001"/>
    <origin xyz="0 0 0"/>
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<joint name="camera_depth_optical_joint" type="fixed">
  <!--origin xyz="0 0 0" rpy="0 0 0" /-->
  <origin rpy="-1.5707963268 0 -1.5707963268" xyz="0 0 0"/>
  <parent link="camera_depth_frame"/>
  <child link="camera_depth_optical_frame"/>
</joint>
<link name="camera_depth_optical_frame">
  <inertial>
    <mass value="0.0001"/>
    <origin xyz="0 0 0"/>
```

```
    <inertia ixx="0.0001" ixy="0.0" ixz="0.0" iyy="0.0001" iyz="0.0" izz="0.0001"
        />
  </inertial>
</link>
<gazebo reference="camera_link">
  <material value="Gazebo/Blue"/>
  <turnGravityOff>false</turnGravityOff>
  <sensor name="openni_camera_camera" type="depth">
    <always_on>1</always_on>
    <visualize>true</visualize>
    <camera>
      <horizontal_fov>1.047</horizontal_fov>
      <image>
        <width>640</width>
        <height>480</height>
        <format>R8G8B8</format>
      </image>
      <depth_camera>

            </depth_camera>
      <clip>
        <near>0.1</near>
        <far>100</far>
      </clip>
    </camera>
    <plugin filename="libgazebo_ros_openni_kinect.so" name="
        camera_link_controller">
      <!--robotNamespace>/rescuer</robotNamespace-->
      <baseline>0.2</baseline>
      <alwaysOn>true</alwaysOn>
      <updateRate>1.0</updateRate>
      <cameraName>rescuer/camera</cameraName>
      <imageTopicName>depth_registered/image_raw</imageTopicName>
      <cameraInfoTopicName>depth_registered/camera_info</cameraInfoTopicName>
      <depthImageTopicName>depth_registered/image_raw</depthImageTopicName>
      <depthImageInfoTopicName>depth_registered/camera_info</
          depthImageInfoTopicName>
      <pointCloudTopicName>depth_registered/points</pointCloudTopicName>
      <frameName>camera_depth_optical_frame</frameName>
      <pointCloudCutoff>0.5</pointCloudCutoff>
      <distortionK1>0</distortionK1>
      <distortionK2>0</distortionK2>
      <distortionK3>0</distortionK3>
      <distortionT1>0</distortionT1>
      <distortionT2>0</distortionT2>
      <CxPrime>0</CxPrime>
      <Cx>0</Cx>
      <Cy>0</Cy>
      <focalLength>0</focalLength>
      <hackBaseline>0</hackBaseline>
      <depthImageCameraInfoTopicName>depthcamerainfo</
          depthImageCameraInfoTopicName>
    </plugin>
  </sensor>
</gazebo>
<gazebo reference="camera_depth_frame">
  <material value="Gazebo/Black"/>
  <turnGravityOff>false</turnGravityOff>
</gazebo>
<gazebo reference="camera_depth_optical_frame">
  <material value="Gazebo/Black"/>
  <turnGravityOff>false</turnGravityOff>
</gazebo>
<link name="right_wheel_link">
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0"/>
```

```
      <mass value="1.0"/>
      <inertia ixx="0.00001" ixy="0" ixz="0" iyy="0.00001" iyz="0" izz="0.00001"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.05" radius="0.075"/>
      </geometry>
      <material name="invisible"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.05" radius="0.075"/>
      </geometry>
    </collision>
  </link>
  <joint name="right_wheel_joint" type="continuous">
    <parent link="base_footprint"/>
    <child link="right_wheel_link"/>
    <!--origin xyz="0 -0.375 0" rpy="${-pi/2} 0 0" /-->
    <origin rpy="-1.5707963268 0 0" xyz="0 -0.4171 0.067"/>
    <axis xyz="0 0 1"/>
  </joint>
  <link name="left_wheel_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <mass value="1.0"/>
      <inertia ixx="0.00001" ixy="0" ixz="0" iyy="0.00001" iyz="0" izz="0.00001"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.05" radius="0.075"/>
      </geometry>
      <material name="invisible"/>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.05" radius="0.075"/>
      </geometry>
    </collision>
  </link>
  <joint name="left_wheel_joint" type="continuous">
    <parent link="base_footprint"/>
    <child link="left_wheel_link"/>
    <!-- origin xyz="0 0.375 0" rpy="${-pi/2}  0 0" /-->
    <origin rpy="-1.5707963268  0 0" xyz="0 0.4171 0.067"/>
    <axis xyz="0 0 1"/>
  </joint>
  <gazebo>
    <plugin filename="libgazebo_ros_diff_drive.so" name="
        differential_drive_controller">
      <!--robotNamespace>/rescuer</robotNamespace-->
      <alwaysOn>true</alwaysOn>
      <updateRate>50</updateRate>
      <leftJoint>left_wheel_joint</leftJoint>
      <rightJoint>right_wheel_joint</rightJoint>
      <wheelSeparation>0.77</wheelSeparation>
      <wheelDiameter>0.26</wheelDiameter>
      <torque>20</torque>
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>/rescuer/odom_combined</odometryTopic>
      <odometryFrame>/rescuer/odom_combined</odometryFrame>
      <robotBaseFrame>/rescuer/base_footprint</robotBaseFrame>
```

```
      <publishWheelTF>true</publishWheelTF>
      <publishWheelJointState>true</publishWheelJointState>
      <rosDebugLevel>Info</rosDebugLevel>
      <wheelAcceleration>1</wheelAcceleration>
      <wheelTorque>20</wheelTorque>
      <odometrySource>world</odometrySource>
      <publishTf>1</publishTf>
    </plugin>
  </gazebo>
  <transmission name="right_wheel_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="right_wheel_joint"/>
    <actuator name="right_wheel_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="left_wheel_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="left_wheel_joint"/>
    <actuator name="left_wheel_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
</robot>
```

Bibliografía

[Leó15] J. J. León. Adaptación del robot rescuer a ros. Master's thesis, Universitat Jaume I, 2015.

[OSF15] Open Source Robotics Foundation OSFR. Which combination of ros/gazebo versions to use. http://gazebosim.org/tutorials?tut=ros_wrapper_versions&cat=connect_ros, 08 2015.

[OSR15a] Open Source Robotics Foundation OSRF. Find out inertial parameters. http://gazebosim.org/tutorials?tut=inertia, 08 2015.

[OSR15b] Open Source Robotics Foundation OSRF. Installing gazebo_ros_pkgs. http://gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros, 08 2015.

[ROS14a] ROS. Urdf. http://wiki.ros.org/urdf, 8 2014.

[ROS14b] ROS. Urdf tutorials. http://wiki.ros.org/urdf/Tutorials, 8 2014.

[ROS15a] ROS. Moveit ikfast. http://docs.ros.org/indigo/api/moveit_ikfast/html/doc/ikfast_tutorial.html, 09 2015.

[ROS15b] ROS. Moveit tutorials. http://moveit.ros.org/documentation/tutorials/, 08 2015.

[ROS15c] ROS. Navigation robot setup. http://wiki.ros.org/navigation/Tutorials/RobotSetup, 08 2015.

[ROS15d] ROS. Ros navigation documentation. http://wiki.ros.org/navigation, 08 2015.

[ROS15e] ROS. Tracks drive plugin. https://bitbucket.org/osrf/gazebo/issues/863/tracksdriveplugin, 11 2015.

[ROS15f] ROS. Tutorial: Ros control. http://gazebosim.org/tutorials/?tut=ros_control, 08 2015.

[Rvi15] rviz 1.11.8 does not load stl. https://github.com/ros-visualization/rviz/issues/913, 11 2015.

Nomenclatura

DAE	Formato Collada, esquema XML abierto para el intercambio entre aplicaciones 3D interactivas
ROS	Robot Operating System
STL	STereoLithography, is a file format native to the stereolithography CAD software created by 3D Systems
URDF	Unified Robot Description Format
Xacro	Lenguaje de macros para XML