CAPÍTULO 1

Software

- AI (VIRBOT)
- Navegación
- Visión
- Habla

1.1. VIRBOT

El sistema VIRBOT consiste de varios subsistemas los cuales controlan la operación del robot móvil.

1.2. Guia principal para el desarrollo de codigo

- Todo código fuente DEBE estar contenido en la carpeta *catkin_ws/src*.
- Sólo el código contenido en la carpeta catkin_ws/src/hardware puede interactuar con el harware del robot
- El punto anterior implica que todos los otros programas deberán implementar SÓLO algoritmos. Todas las interacciones con el hardware (e.g.. obtener una imagen desde la cámara, leer el láser, mover la base o la cabeza, hablar, etc.) debe hacerse intercambiando información con los paquetes contenidos en la carpeta **hardware**, a través de los tópicos y servicios de ROS.
- Los códigos contenidos en todas las carpetas dentro de *catkin_ws/src*, excepto las carpetas de herramientas, DEBEN contener sólo código escrito por el propio desarrollador (de cualquier paquete). Todas las bibliotecas necesarias o código de otras fuentes (bibliotecas serial, arduino, julius, dynamixel, etc.), si no están instaladas en algún default path

(/opt/ros, /usr/local/, etc.), deben ser puestas dentro de la carpeta *catkin_ws/src/tools* en una sub-carpeta apropiada.

- Los desarrolladores deben tratar de usar sólo mensajes ya definidos en algún paquete de ROS o pila, sin embargo, si mensajes personalizados son requeridos, éstos deben ser puestos dentro de *catkin_ws/src/subsystem/subsystem_maga*, así que, muchos mensajes pueden ser usados sin necesidad de ejecutar todos los demás subsistemas.

1.3. Estructura de la carpeta

```
catkin_ws
  build
  devel
  src
      hardware
  arms
  battery
  hardware_state
  justina_urdf
  hardware_msgs
  head
  mobile_base
  point_cloud_manager
  speakers
  torso
      hri
  gesture_recog
  hri_msgs
  justina_gui
  natural_language
  speech_recog
      interoperation
  bbros_bridge
  joy_teleop
  pc_teleop
  roah_rsbb
      manipulation
  arms_predef_movs
  arms_path_planning
  arms_trajectory_planning
  head_predef_movs
```

```
  head_tracking_point
  manipulation_msgs
      navigation
  localization
  mapping
  moving
  navigation_msgs
  path_planning
  point_traking
      planning
  planning_msgs
  pomdp
  rule_based
  semantic_database
  state_machines
      surge_et_ambula
  launch
  rviz_files
      testing
  any_not_stable_node
      tools
  ros_tools
  libraries
      serial_arduino
      serial_dynamixel
      julius
      festival
      vision
  door_detector
  furniture_recog
  object_detector
  object_recog
  person_detection
  person_recog
  vision_msgs
user_manual
```

Cada paquete en la carpeta de *hardware* debe tener su versión simulada, así que, el resto del software (todas las otras carpetas se supone que contienen sólo algoritmos y no interacción con el hardware del robot) puedes correr inmediatamente el modo de simulación. Eligiendo entre simulado o real debe ser hecho en la carpeta de ejecución.

1.4. Nodos de ROS

Con el comando *rosrun* se puede ejecutar un nodo de un paquete sin tener que conocer la ruta completa, indicando el nombre del paquete dentro del cual se encuentra el nodo que deseamos lanzar y el nombre del ejecutable.

Uso:

```
rosrun package_name executable_name
```

Por otro lado, *roslaunch* es una herramienta para lanzar fácilmente múltiples nodos de ROS de forma local y remota a través de SSH, así como establecer parámetros en el Servidor de Parámetros. Antes de iniciar cualquier nodo, *roslaunch* determinará si *roscore* ya está en ejecución y, en caso contrario, lo iniciara automáticamente. Los archivos *.launch* del robot Justina se encuentran dentro del paquete *surge_et_ambula* en la carpeta *launch*.

Uso:

```
roslaunch package_name file_name.launch
```

Los archivos *launch* son documentos XML, y cada documento XML debe tener un elemento raíz, para el caso de los archivos *launch* de ROS, el elemento raíz se define mediante un par de etiquetas `launch`:

```
<launch>
</launch>
```

Todos los demás elementos del archivo se deben incluir dentro de estas etiquetas.

La etiqueta <`node`> especifica un nodo ROS que se desea lanzar y tiene tres atributos requeridos, ésta etiqueta se ve de esta forma:

```
<node name="node-name" pkg="package-name" type="executable-name" />
```

Los atributos `pkg` y `type` identifican qué programa debe ejecutar ROS para iniciar este nodo. Estos son los mismos dos argumentos que toma el comando *rosrun*, especificando el nombre del paquete y el nombre del ejecutable, respectivamente. El atributo `name` asigna un nombre al nodo. Esto anula cualquier nombre que el nodo se asignaría normalmente a sí mismo en su llamada a `ros::init`.

Existen otros atributos opcionales utilizados en la etiqueta <`node`>:

- `args=‘‘arg1 arg2 arg3’’`: Pasar argumentos al nodo.
- `output=‘‘screen’’`: Los nodos iniciados con este atributo mostrarán sus salidas stdout/stderr en la pantalla.

La etiqueta <`group`> facilita la aplicación de configuraciones a un grupo de nodos. Tiene un atributo `ns` que le permite definir un *namespace* independiente para un grupo de nodos.

```
<group ns="namespace">
        <node name="node-name" pkg="package-name" type="executable-name" />
</group>
```

La etiqueta <`group`> es equivalente a la etiqueta de nivel superior <`launch`> y simplemente actúa como un contenedor para las etiquetas que se encuentran dentro. Esto significa que puede usar cualquier etiqueta como se usaría normalmente dentro de una etiqueta <`launch`>.

Los nodos de ROS también admiten reasignaciones, que proporcionan un nivel de control para modificar los nombres utilizados por los nodos. Las reasignaciones se basan en la idea de sustitución: cada reasignación proporciona un nombre original y un nuevo nombre. Para reasignar nombres dentro de un archivo *launch* se utiliza la etiqueta <`remap`>:

```
<remap from="original-name" to="new-name"/>
```

Si la etiqueta <`remap`> aparece en el nivel superior como hijo de la etiqueta <`launch`>, esta reasignación se aplicará a todos los nodos subsiguientes. Estos elementos de reasignación también pueden aparecer como hijos de una etiqueta <`node`>, en este caso, los reasignamientos dados se aplican solamente al nodo en el que estén contenidos, como en este ejemplo:

```
<node name="node-name" pkg="package-name" type="executable-name" >
        <remap from="original-name" to="new-name" />
</node>
```

La etiqueta <`param`> define un parámetro que se establecerá en el Servidor de Parámetros. Esta etiqueta, como uno habría de esperar, asigna el valor dado al parámetro con el nombre dado.

```
<param name="param-name" value="param-value" />
```

La etiqueta <param> se puede colocar dentro de una etiqueta <node>, en cuyo caso el parámetro se trata como un parámetro privado.

```
<node name="node-name" pkg="package-name" type="executable-name" >
        <param name="param-name" value="param-value" />
</node>
```

Para obtener más información acerca de estas etiquetas y de los archivos *launch*, por favor consulte: http://wiki.ros.org/roslaunch/XML.

1.4.1. Nodo /robot_state_publisher

Este nodo se encuentra dentro del paquete *robot_state_publisher*, y permite publicar el estado del robot a *tf*. Una vez que el estado se publica, está disponible para todos los componentes del sistema que utlizan *tf*. El paquete toma las posiciones de las juntas del robot como entrada y publica las poses 3D de los eslabones usando un modelo de árbol cinemático. *tf* es un paquete que permite al usuario realizar el seguimiento de varios marcos de referencia a lo largo del tiempo.

robot_state_publisher usa el URDF especificado por el parámetro *robot_description*, y las posiciones de las juntas del tópico *joint_states* para calcular la cinemática directa del robot y publicar los resultados a través de *tf*. URDF (Unified Robot Description Format) es un formato XML para representar el modelo de un robot.

Tópicos suscritos	/joint_states [sensor_msgs/JointState]	Se suscribe a la información de la posición de las juntas
Tópicos publicados	/tf [tf/tfMessage]	Publica el estado del robot
Parámetros	robot_description [urdf map, default:]	Archivo XML del modelo del robot
	tf_prefix [string, default:]	Establece el prefijo tf para el namespace
	publish_frequency [double, default: 50Hz]	Frecuencia a la que publica el nodo
	use_tf_static [bool, default: false]	Define si se quiere utilizar /tf_static

Tabla 1.1: Nodo /robot_state_publisher

Sintaxis en un archivo launch
Lanzamiento del nodo y ajuste del parámetro *robot_description*.

```
<param name="robot_description" command="cat $(find knowledge)/hardware/justina.xml"/>
<node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher"/>$
```

En el archivo *justina.xml* se encuentra el modelo del robot Justina; en la Figura 1.1 se tiene el árbol de transformaciones, los ovalos azules representan las juntas, mientras que los recuadros negros representan los sistemas de referencia asociados a los eslabones del robot. En el grafo dirigido se muestran los offset de traslación en x, y y z, y los offset de los ángulos de rotación *roll*, *pitch* y *yaw* que se tienen entre los sistemas de referencia, las unidades están en metros y radianes respectivamente.

En el grafo dirigido de la Figura 1.2 se muestran todas las transformaciones entre sistemas de referencia para el robot, en éste grafo se incluyen además los marcos *odom* y *map*. Para más información acerca de los sistemas de referencia para plataformas móviles consulte: `http://www.ros.org/reps/rep-0105.html`.

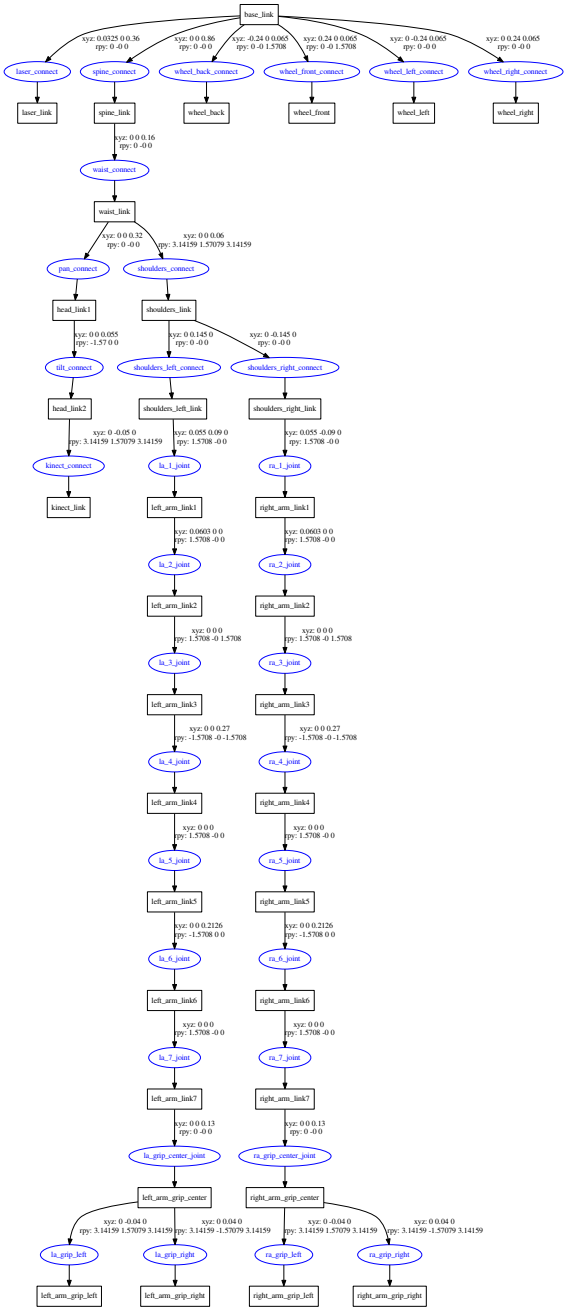

Figura 1.1: Árbol de transformaciones

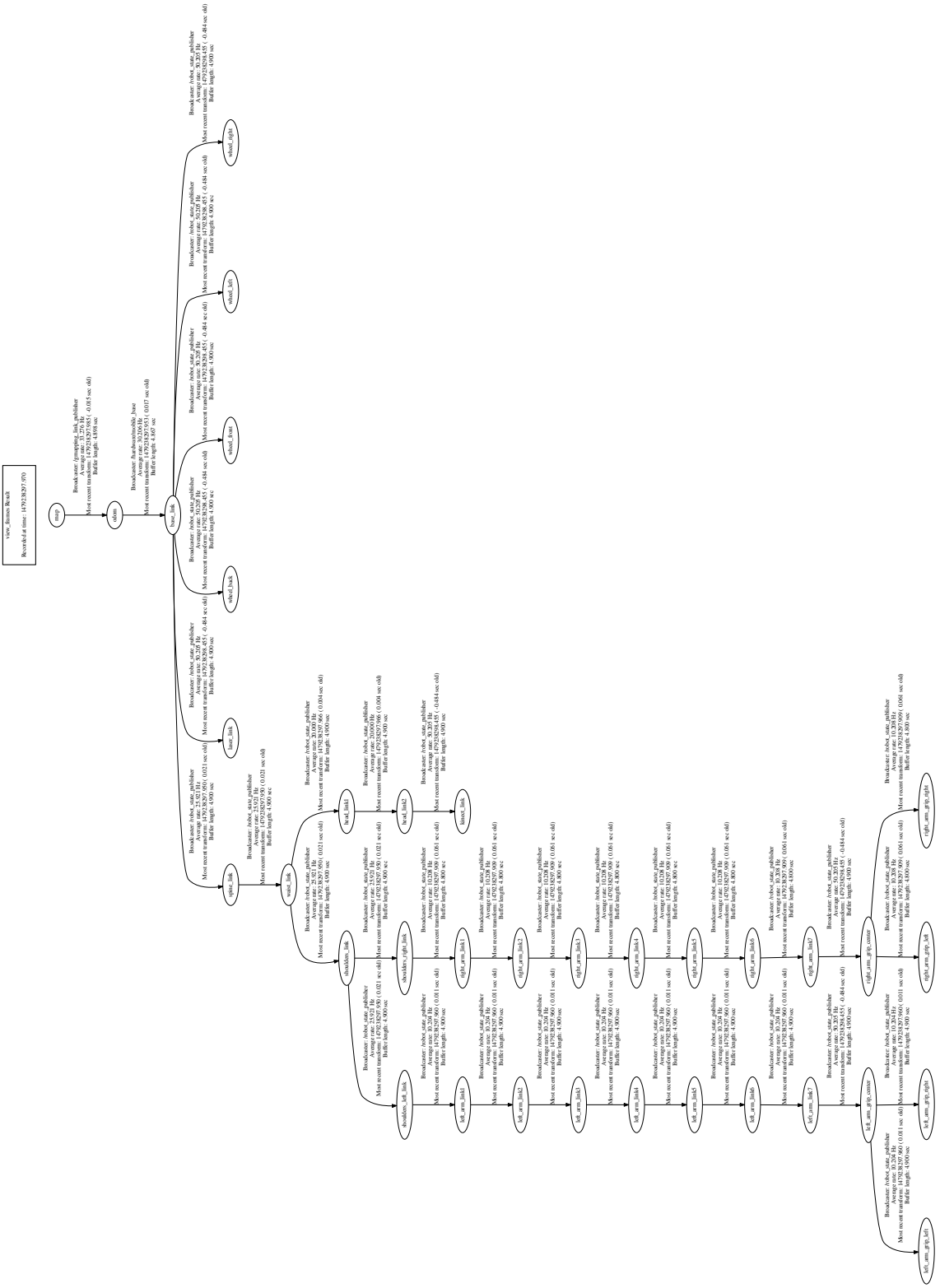

Figura 1.2: Árbol de transformaciones con los marcos de referencia *map* y *odom*

1.4.2. Nodo /hardware/head

Este nodo se encarga de controlar la posición de la cabeza mediante los grados de libertad *pan* y *tilt*. La posición deseada se establece en radianes, en caso de que estos valores se encuentren fuera del rango de alcance de la junta, entonces se posicionará en la cota superior o inferior según sea el caso.

Tópicos suscritos	/hardware/head/goal_pose [std_msgs/Float32MultiArray]	Posición deseada de las juntas de revolución de la cabeza
Tópicos publicados	/hardware/head/current_pose [std_msgs/Float32MultiArray]	Posición actual de las juntas de la cabeza
	/joint_states [sensor_msgs/JointState]	Descripción del estado de las juntas de la cabeza
	/hardware/robot_state/head_battery [std_msgs/Float32]	Voltaje de alimentación de los servo motores de la cabeza

Tabla 1.2: Nodo /hardware/head

Sintaxis en un archivo launch

Para correr este nodo es necesario indicar como argumentos el puerto serial en el cual esta conectado el dispositivo *USB2Dynamixel* asociado a los servo motores de la cabeza, así como el baudaje al cual se establecerá la comunicación.

```
<node name="head" pkg="head" type="head_node.py" output="screen" args="--port
/dev/justinaHead --baud 1000000"/>
```

1.4.3. Nodo /hardware/hokuyo_node

Este nodo se encarga de la adquisición de datos en un sensor Hokuyo Láser y, los hace accesibles mediante un mensaje de tipo *LaserScan*. Los escaneos del Hokuyo se toman en sentido contrario a las agujas del reloj, así mismo, los ángulos se miden en sentido contrario a las agujas del reloj con 0 apuntando directamente hacia delante.

Tópicos publicados	/hardware/scan [sensor_msgs/LaserScan]	Datos de un escaneo
Parámetros	port [string, default: /dev/ttyACM0] frame_id [string, default: laser]	Puerto donde se encuentra el dispositivo Hokuyo Marco de referencia asociado al láser

Tabla 1.3: Nodo /hardware/hokuyo_node

Sintaxis en un archivo launch
Para lanzar este nodo por medio de un archivo *lauch* es necesario indicar como parámetros el puerto en el que se encuentra el dispositivo y el marco de referencia asociado al láser, dicho marco se encuentra definido en el archivo *justina.xml*.

```
<node name="hokuyo_node" pkg="hokuyo_node" type="hokuyo_node" output="screen">
    <param name="port" type="string" value="/dev/justinaHokuyo" />
    <param name="frame_id" type="string" value="laser_link" />
</node>
```

1.4.4. Nodo /hardware/joy

Este nodo conecta un joystick genérico de Linux a ROS; publica un mensaje de tipo *Joy* que contiene el estado actual de cada uno de los botones y ejes del joystick.

Tópicos publicados	/hardware/joy [sensor_msgs/Joy]	Reporta el estado de los ejes y botones del joystick
Parámetros	dev [string, default: /dev/input/js0]	Dispositivo desde el cual se leen los eventos

Tabla 1.4: Nodo /hardware/joy

Sintaxis en un archivo launch
Para lanzar este nodo por medio de un archivo *lauch* únicamente es necesario indicar el nombre

que se le desea dar al nodo, el paquete en el que se encuentra y el nombre del ejecutable.

```
<node name="joy" pkg="joy" type="joy_node" output="screen"/>
```

1.4.5. Nodo /hardware/left_arm

Este nodo se encarga de controlar la posición de los 7 grados de libertad del brazo izquierdo del robot Justina, de igual modo controla el agarre del *gripper*; la posición deseada para cada uno de los GDL se establece en radianes.

Tópicos publicados	/hardware/left_arm/current_pose [std_msgs/Float32MultiArray]	Posición actual de las juntas del brazo izquierdo
	/hardware/left_arm/current_gripper [std_msgs/Float32]	Posición actual del gripper
	/joint_states [sensor_msgs/JointState]	Descripción del estado de las juntas del brazo izquierdo
	/hardware/robot_state/left_arm_battery [std_msgs/Float32]	Voltaje de alimentación de los servo motores del brazo izquierdo
Tópicos suscritos	/hardware/left_arm/goal_gripper [std_msgs/Float32]	Posición deseada del gripper
	/hardware/left_arm/torque_gripper [std_msgs/Float32]	Par deseado en el gripper para tareas de manipulación de objetos
	/hardware/left_arm/goal_pose [std_msgs/Float32MultiArray]	Posición deseada de las juntas de revolución del brazo izquierdo. Si se especifican 7 datos, éstos serán las posiciones deseadas de los 7 GDL, si son 14 datos, los 7 adicionales serán las rapideces de los servo motores a las que se desea alcanzar dicha posición

Tabla 1.5: Nodo /hardware/left_arm

Sintaxis en un archivo launch

Para correr este nodo es necesario indicar como argumentos el puerto serial en el cual esta conectado el dispositivo *USB2Dynamixel* asociado a los servo motores del brazo izquierdo, además del baudaje al cual se establecerá la comunicación.

```
<node name="left_arm" pkg="arms" type="left_arm_node.py" output="screen" args="--por
/dev/justinaLeftArm --baud1 1000000"/>
```

1.4.6. Nodo /hardware/mobile_base

Este nodo se encarga de controlar el movimiento de la base estableciendo la rapidez de los motores por medio de controladores *RoboClaw*, la velocidad y rapidez lineal están dadas en metros por segundo y, la velocidad angular en radianes por segundo. Aparte de esto, se determina la ubicación del robot en un mapa estático utilizando *tf* y un mensaje de tipo *Odometry*, para obtener información más detallada consulte por favor: `http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom`.

Tópicos publicados	/hardware/mobile_base/odometry [nav_msgs/Odometry]	Odometría calculada con las lecturas de los encoder de los motores
	/hardware/robot_state/base_battery [std_msgs/Float32]	Voltaje de alimentación de los motores de la base
Tópicos suscritos	/hardware/robot_state/stop [std_msgs/Empty]	Tópico para parar los motores de la base
	/hardware/mobile_base/cmd_vel [geometry_msgs/Twist]	Velocidad lineal deseada en el plano *xy*, y velocidad angular deseada en *z*
	/hardware/mobile_base/speeds [std_msgs/Float32MultiArray]	Rapideces instantáneas derecha e izquierda deseadas

Tabla 1.6: Nodo /hardware/mobile_base

Sintaxis en un archivo launch

La base móvil con la que cuenta el robot Justina actualmente posee cuatro motores, es por ello que se requieren dos controladores *RoboClaw*. Para lanzar este nodo se especifica mediante argumentos los dos puertos en los cuales están conectados los controladores.

```
<node name="mobile_base" pkg="mobile_base" type="omni_base_node.py" output="screen"
args="--port1 /dev/justinaRC15 --port2 /dev/justinaRC30"/>
```

1.4.7. Nodo /hardware/torso

Tópicos publicados	/hardware/torso/goal_reached [std_msgs/Bool] /tf [tf/tfMessage] /joint_states [sensor_msgs/JointState] /hardware/torso/current_pose [std_msgs/Float32MultiArray]	
Tópicos suscritos	/hardware/torso/goal_pose [std_msgs/Float32MultiArray] /hardware/torso/goal_rel_pose [std_msgs/Float32MultiArray]	

Tabla 1.7: Nodo /hardware/torso

1.4.8. Nodo /hri/human_follower

Tópicos publicados	/hardware/mobile_base/speeds [std_msgs/Float32MultiArray]	
Tópicos suscritos	/hri/human_following/start_follow [std_msgs/Bool] /hri/leg_finder/leg_poses [geometry_msgs/PointStamped]	

Tabla 1.8: Nodo /hri/human_follower

1.4.9. Nodo /hri/justina_gui

Tópicos publicados	/hardware/point_cloud_man/save_cloud [std_msgs/String] /navigation/path_planning/ simple_move/goal_lateral [std_msgs/Float32] /hardware/torso/goal_pose [std_msgs/Float32MultiArray] /manipulation/manip_pln/hd_goto_loc [std_msgs/String] /hardware/robot_state/stop [std_msgs/Empty] /vision/face_recognizer/start_recog_old [std_msgs/Empty] /manipulation/manip_pln/ra_goto_loc [std_msgs/String] /manipulation/manip_pln/ la_goto_angles [std_msgs/Float32MultiArray] /manipulation/manip_pln/ la_pose_wrt_robot [std_msgs/Float32MultiArray] /navigation/path_planning/ simple_move/goal_dist [std_msgs/Float32]	

Tabla 1.9: Nodo /hri/justina_gui

Tópicos publicados	/vision/face_recognizer/ run_face_recognizer_id [std_msgs/String] /hardware/point_cloud_man/ stop_saving_cloud [std_msgs/Empty] /hardware/mobile_base/cmd_vel [geometry_msgs/Twist] /manipulation/manip_pln/ ra_pose_wrt_robot [std_msgs/Float32MultiArray] /navigation/mvn_pln/get_close_xya [std_msgs/Float32MultiArray] /hardware/right_arm/goal_torque [std_msgs/Float32MultiArray] /manipulation/manip_pln/la_move [std_msgs/String] /navigation/mvn_pln/get_close_loc [std_msgs/String] /vision/obj_reco/enableRecognizeTopic [std_msgs/Bool] /hardware/left_arm/goal_gripper [std_msgs/Float32]	

Tabla 1.10: Nodo /hri/justina_gui

Tópicos publicados	/hardware/mobile_base/speeds [std_msgs/Float32MultiArray] /recognizedSpeech [hri_msgs/RecognizedSpeech] /hardware/head/goal_pose [std_msgs/Float32MultiArray] /navigation/path_planning/ simple_move/goal_rel_pose [geometry_msgs/Pose2D] /hri/human_following/start_follow [std_msgs/Bool] /vision/obj_reco/enableDetectWindow [std_msgs/Bool] /hardware/right_arm/torque_gripper [std_msgs/Float32] /vision/face_recognizer/ run_face_trainer_frames [vision_msgs/VisionFaceTrainObject] /hardware/torso/goal_rel_pose [std_msgs/Float32MultiArray] /navigation/obs_avoid/enable [std_msgs/Bool]	

Tabla 1.11: Nodo /hri/justina_gui

Tópicos publicados	/vision/thermal_vision/stop_video [std_msgs/Empty] /vision/skeleton_finder/stop_recog [std_msgs/Empty] /vision/face_recognizer/clearfacesdb [std_msgs/Empty] /manipulation/manip_pln/ hd_goto_angles [std_msgs/Float32MultiArray] /hardware/left_arm/goal_torque [std_msgs/Float32MultiArray] /vision/face_recognizer/clearfacesdbbyid [std_msgs/String] /hardware/left_arm/torque_gripper [std_msgs/Float32] /navigation/path_planning/ simple_move/goal_dist_angle [std_msgs/Float32MultiArray] /manipulation/manip_pln/ la_pose_wrt_arm [std_msgs/Float32MultiArray] /hardware/left_arm/goal_pose [std_msgs/Float32MultiArray]	

Tabla 1.12: Nodo /hri/justina_gui

Tópicos publicados	/navigation/mvn_pln/add_location [navig_msgs/Location] /hri/leg_finder/enable [std_msgs/Bool] /vision/face_recognizer/ run_face_recognizer [std_msgs/Empty] /hri/sp_rec/recognized [std_msgs/String] /vision/thermal_vision/start_video [std_msgs/Empty] /vision/face_recognizer/run_face_trainer [std_msgs/String] /manipulation/manip_pln/la_goto_loc [std_msgs/String] /vision/face_recognizer/stop_recog [std_msgs/Empty] /hardware/right_arm/goal_pose [std_msgs/Float32MultiArray] /vision/face_recognizer/start_recog [std_msgs/Empty]	

Tabla 1.13: Nodo /hri/justina_gui

Tópicos publicados	/navigation/path_planning/simple_move /goal_path [nav_msgs/Path] /vision/skeleton_finder/start_recog [std_msgs/Empty] /navigation/path_planning/simple_move /goal_pose [geometry_msgs/Pose2D] /vision/qr/start_qr [std_msgs/Bool] /manipulation/manip_pln /ra_goto_angles [std_msgs/Float32MultiArray] /hardware/right_arm/goal_gripper [std_msgs/Float32] /manipulation/manip_pln /ra_pose_wrt_arm [std_msgs/Float32MultiArray]	

Tabla 1.14: Nodo /hri/justina_gui

Tópicos suscritos	/hardware/left_arm/current_pose [std_msgs/Float32MultiArray] /hardware/robot_state/stop [std_msgs/Empty] /vision/face_recognizer/faces [vision_msgs/VisionFaceObjects] /recognizedSpeech [hri_msgs/RecognizedSpeech] /hardware/left_arm/current_gripper [std_msgs/Float32] /hri/leg_finder/legs_found [std_msgs/Empty] /hardware/right_arm/current_gripper [] /navigation/localization/current_pose [geometry_msgs/ PoseWithCovarianceStamped] /hardware/head/current_pose [std_msgs/Float32MultiArray] /hardware/torso/goal_reached [std_msgs/Bool]	

Tabla 1.15: Nodo /hri/justina_gui

Tópicos suscritos	/tf [tf/tfMessage] /navigation/obs_avoid/obs_in_front [std_msgs/Bool] /tf_static [tf2_msgs/TFMessage] /manipulation/hd_goal_reached [std_msgs/Bool] /hri/sp_rec/recognized [std_msgs/String] /hardware/robot_state/ right_arm_battery [] /hardware/right_arm/current_pose [] /hardware/torso/current_pose [std_msgs/Float32MultiArray] /manipulation/ra_goal_reached [std_msgs/Bool] /navigation/global_goal_reached [std_msgs/Bool]	

Tabla 1.16: Nodo /hri/justina_gui

Tópicos suscritos	/navigation/obs_avoid/collision_risk [std_msgs/Bool] /navigation/goal_reached [std_msgs/Bool] /vision/face_recognizer/trainer_result [std_msgs/Int32] /hardware/robot_state/left_arm_battery [std_msgs/Float32] /hardware/robot_state/head_battery [std_msgs/Float32] /hri/qr/recognized [std_msgs/String] /manipulation/la_goal_reached [std_msgs/Bool] /hardware/robot_state/base_battery [std_msgs/Float32]	

Tabla 1.17: Nodo /hri/justina_gui

1.4.10. Nodo /hri/leg_finder

Tópicos publicados	/hri/leg_finder/legs_found [std_msgs/Empty] /hri/leg_finder/leg_poses [geometry_msgs/PointStamped]	
Tópicos suscritos	/hardware/scan [sensor_msgs/LaserScan] /tf [tf/tfMessage] /tf_static [tf2_msgs/TFMessage] /hri/leg_finder/enable [std_msgs/Bool]	

Tabla 1.18: Nodo /hri/leg_finder

1.4.11. Nodo /hri/qr_reader

Tópicos publicados	/hri/qr/recognized [std_msgs/String]	
Tópicos suscritos	/tf [tf/tfMessage] /tf_static [tf2_msgs/TFMessage] /vision/qr/start_qr [std_msgs/Bool]	

Tabla 1.19: Nodo /hri/qr_reader

1.4.12. Nodo /hri/rviz

Tópicos publicados	/clicked_point [geometry_msgs/PointStamped] /move_base_simple/goal [geometry_msgs/PoseStamped] /initialpose [geometry_msgs/ PoseWithCovarianceStamped]	
Tópicos suscritos	/hri/rviz/location_markers [visualization_msgs/Marker] /hardware/scan [sensor_msgs/LaserScan] /hri/rviz/location_markers_array [] /tf [tf/tfMessage] /tf_static [tf2_msgs/TFMessage] /hri/leg_finder/leg_poses [geometry_msgs/PointStamped] /navigation/localization/map_updates [] /navigation/mvn_pln/last_calc_path [nav_msgs/Path] /navigation/localization/map [nav_msgs/OccupancyGrid]	

Tabla 1.20: Nodo /hri/rviz

1.4.13. Nodo /hri/sp_gen

Tópicos suscritos	/hri/sp_gen/say []	

Tabla 1.21: Nodo /hri/sp_gen

1.4.14. Nodo /interoperation/joystick_teleop

Este nodo se suscribe a un mensaje de tipo *Joy* para controlar mediante un joystick de una consola Xbox el movimiento de la base, la posición de la cabeza y del torso. Las posiciones angulares y lineales, así como las velocidades lineales y angulares están dadas en radianes, metros, metros por segundo y radianes por segundo respectivamente.

Para mover la cabeza del robot se utiliza el *stick* izquierdo, la base se opera por medio del *stick* derecho, mientras que el botón rojo del joystick es para el paro de la base.

Tópicos publicados	/hardware/robot_state/stop [std_msgs/Empty]	Tópico de paro para los motores de la base
	/hardware/mobile_base/cmd_vel [geometry_msgs/Twist]	Velocidad lineal deseada deseada de la base en el plano xy, y velocidad angular deseada en z
	/hardware/head/goal_pose [std_msgs/Float32MultiArray]	Posición deseada de la cabeza
	/hardware/torso/goal_spine [std_msgs/Float32]	Posición deseada de la junta prismática para cambiar la altura del torso
	/hardware/torso/goal_shoulders [std_msgs/Float32]	Posición deseada de la junta de revolución para orientar el torso (roll)
	/hardware/torso/goal_waist [std_msgs/Float32]	Posición deseada de la junta de revolución para orientar el torso (yaw)
Tópicos suscritos	/hardware/joy [sensor_msgs/Joy]	Estado de los ejes y botones de un joystick

Tabla 1.22: Nodo /interoperation/joystick_teleop

Sintaxis en un archivo launch

Para lanzar este nodo por medio de un archivo *lauch* sólo se necesita indicar el nombre que se le desea dar al nodo, el paquete dentro del que se encuentra y el nombre del ejecutable.

```
<node name="joystick_teleop" pkg="joystick_teleop" type="joystick_teleop_node.py"
output="screen" />
```

1.4.15. Nodo /manipulation/ik_geometric

Servicios	/manipulation/ik_geometric/ ik_float_array /manipulation/ik_geometric/ik_path /manipulation/ik_geometric/ik_pose /manipulation/ik_geometric/ direct_kinematics	

Tabla 1.23: Nodo /manipulation/ik_geometric

1.4.16. Nodo /manipulation/manip_pln

Tópicos publicados	/hardware/right_arm/goal_torque [std_msgs/Float32MultiArray] /hardware/head/goal_pose [std_msgs/Float32MultiArray] /hardware/left_arm/goal_torque [std_msgs/Float32MultiArray] /hardware/left_arm/goal_pose [std_msgs/Float32MultiArray] /manipulation/hd_goal_reached [std_msgs/Bool] /manipulation/ra_goal_reached [std_msgs/Bool] /hardware/right_arm/goal_pose [std_msgs/Float32MultiArray] /hardware/head/goal_torque [std_msgs/Float32MultiArray] /manipulation/la_goal_reached [std_msgs/Bool]	

Tabla 1.24: Nodo /manipulation/manip_pln

Tópicos suscritos	/hardware/left_arm/current_pose [std_msgs/Float32MultiArray] /manipulation/manip_pln/ra_goto_loc [std_msgs/String] /manipulation/manip_pln/hd_goto_loc [std_msgs/String] /manipulation/manip_pln/ la_goto_angles [std_msgs/Float32MultiArray] /manipulation/manip_pln/ la_pose_wrt_robot [std_msgs/Float32MultiArray] /manipulation/manip_pln/ ra_pose_wrt_robot [std_msgs/Float32MultiArray] /manipulation/manip_pln/la_move [std_msgs/String] /hardware/head/current_pose [std_msgs/Float32MultiArray] /manipulation/manip_pln/hd_move [] /manipulation/manip_pln/ra_move []	

Tabla 1.25: Nodo /manipulation/manip_pln

Tópicos suscritos	/manipulation/manip_pln/ hd_goto_angles [std_msgs/Float32MultiArray] /tf [tf/tfMessage] /manipulation/manip_pln/ la_pose_wrt_arm [std_msgs/Float32MultiArray] /tf_static [tf2_msgs/TFMessage] /hardware/right_arm/current_pose [] /manipulation/manip_pln/la_goto_loc [std_msgs/String] /manipulation/manip_pln/ ra_pose_wrt_arm [std_msgs/Float32MultiArray] /manipulation/manip_pln/ ra_goto_angles [std_msgs/Float32MultiArray]	

Tabla 1.26: Nodo /manipulation/manip_pln

1.4.17. Nodo /navigation/localization/loc_amcl

Este nodo implementa el enfoque adaptativo de localización de Monte Carlo, que utiliza un filtro de partículas para rastrear la pose de un robot en un mapa conocido. AMCL es un sistema de localización probabilística para un robot que se mueve en un plano.

AMCL transforma los escaneos láser entrantes al sistema de referencia *odometry*. Por lo tanto, debe existir un camino a través del árbol *tf* desde el sistema de referencia en el que los escaneos láser se publican hacia el sistema de referencia de odometría.

Durante la operación AMCL estima la transformación del marco de referencia de la base con respecto al marco de referencia global(*map* para este caso), pero solamente publica la transformación entre el marco de referencia global(*map*) y el marco de referencia de odometría(*odometry*). Esencialmente, esta transformación considera la deriva que ocurre usando

Dead Reckoning. *Dead Reckoning* es el proceso de calcular la posición estimando la dirección y la distancia recorrida.

Para obtener información más detallada consulte: `http://wiki.ros.org/amcl`.

Tópicos publicados	/navigation/localization/current_pose [geometry_msgs/ PoseWithCovarianceStamped]	Posición estimada del robot en el mapa, con covarianza
	/tf [tf/tfMessage]	Publica la transformación de odom (que se puede reasignar a través del parámetro *odom_frame_id*) a map
	/navigation/localization/particlecloud [geometry_msgs/PoseArray]	Conjunto de poses estimadas mantenidas por el filtro
Servicios	/navigation/localization/ global_localization [std_srvs/Empty]	Inicio de la localización global, donde todas las partículas se dispersan al azar a través del espacio libre en el mapa

Tabla 1.27: Nodo /navigation/localization/loc_amcl

Tópicos suscritos	/navigation/localization/initialpose [geometry_msgs/ PoseWithCovarianceStamped]	Media y covarianza con la cual se (re-)inicializa el filtro de partículas
	/hardware/scan [sensor_msgs/LaserScan]	Escaneos láser
	/tf [tf/tfMessage]	Transformaciones del robot
Parámetros	update_min_a [double, default: π/6.0 radians]	Movimiento de rotación requerido antes de realizar una actualización del filtro
	laser_min_range [double, default: -1.0]	Rango de escaneo mínimo a considerar
	odom_model_type [string, default: "diff"]	Configuración del robot, ya sea "diff", "omni", "diff-corrected" o "omni-corrected"

Tabla 1.28: Nodo /navigation/localization/loc_amcl

Sintaxis en un archivo launch

Para correr este nodo se necesita indicar el tópico en el cual se publican los datos del láser (*/hardware/scan* para este caso), de igual forma se requiere modificar los parámetros *update_min_a*, *laser_min_range* y *odom_model_typ*.

```
<node name="loc_amcl" pkg="amcl" type="amcl" output="acreen" args="scan:=/hardware/s
    <param name="update_min_a" value="0.3"/>
    <param name="laser_min_range" value="0.3"/>
    <param name="odom_model_type" value="omni"/>
</node>
```

1.4.18. Nodo /navigation/localization/map_server

Este nodo ofrece datos de un mapa como un servicio de ROS. También proporciona la utilidad de línea de comandos *map_saver*, que permite guardar en un archivo los mapas generados

dinámicamente.

Los mapas manipulados por las herramientas de este paquete se almacenan en un par de archivos, un archivo YAML describe los metadatos del mapa y una imagen codifica los datos de ocupación. La imagen describe el estado de ocupación de cada celda del mundo en el color del píxel correspondiente. Los píxeles más blancos están libres, los píxeles más negros están ocupados y los píxeles entre estos colores son desconocidos. Los campos requeridos en el archivo YAML son seis: *image*, *resolution*, *origin*, *occupied_thresh*, *free_thresh* y *negate*.

Para obtener información más específica por favor consulte: `http://wiki.ros.org/map_server`.

Tópicos publicados	/navigation/localization/map_metadata [nav_msgs/MapMetaData] /navigation/localization/map [nav_msgs/OccupancyGrid]	Metadatos del mapa Mapa
Servicios	navigation/localization/static_map [nav_msgs/GetMap]	Obtención del mapa a través de este servicio
Parámetros	frame_id [string, default: "map"]	Marco de referencia establecido en el encabezado (*header*) del mapa publicado

Tabla 1.29: Nodo /navigation/localization/map_server

Sintaxis en un archivo launch

Para ejecutar este nodo se requiere especificar como argumento el archivo YAML que contiene los metadatos del mapa que se quiere proveer. Para este ejemplo, el archivo *bioroboanexo3.yaml* se encuentra dentro del paquete *knowledge* en la ruta *navigation/occupancy_grids*.

```
<node name="map_server" pkg="map_server" type="map_server" output="screen"
      args="$(find knowledge)/navigation/occupancy_grids/bioroboanexo3.yaml"/>$
```

1.4.19. Nodo /navigation/mvn_pln

Tópicos publicados	/navigation/path_planning/simple_move /goal_lateral [std_msgs/Float32] /hardware/torso/goal_pose [std_msgs/Float32MultiArray] /manipulation/manip_pln/hd_goto_loc [std_msgs/String] /manipulation/manip_pln/ra_goto_loc [std_msgs/String] /hri/rviz/location_markers [visualization_msgs/Marker] /manipulation/manip_pln/ la_goto_angles [std_msgs/Float32MultiArray] /manipulation/manip_pln/ la_pose_wrt_robot [std_msgs/Float32MultiArray] /navigation/path_planning/simple_move /goal_dist [std_msgs/Float32] /manipulation/manip_pln/ ra_pose_wrt_robot [std_msgs/Float32MultiArray]	

Tabla 1.30: Nodo /navigation/mvn_pln

Tópicos publicados	/navigation/mvn_pln/get_close_xya [std_msgs/Float32MultiArray] /manipulation/manip_pln/la_move [std_msgs/String] /navigation/mvn_pln/get_close_loc [std_msgs/String] /hardware/left_arm/goal_gripper [std_msgs/Float32] /navigation/path_planning/simple_move /goal_rel_pose [geometry_msgs/Pose2D] /hardware/right_arm/torque_gripper [std_msgs/Float32] /hardware/torso/goal_rel_pose [std_msgs/Float32MultiArray] /navigation/obs_avoid/enable [std_msgs/Bool] /manipulation/manip_pln/ hd_goto_angles [std_msgs/Float32MultiArray]	

Tabla 1.31: Nodo /navigation/mvn_pln

Tópicos publicados	/hardware/left_arm/torque_gripper [std_msgs/Float32] /navigation/path_planning/simple_move /goal_dist_angle [std_msgs/Float32MultiArray] /manipulation/manip_pln/ la_pose_wrt_arm [std_msgs/Float32MultiArray] /navigation/mvn_pln/add_location [navig_msgs/Location] /navigation/mvn_pln/last_calc_path [nav_msgs/Path] /navigation/global_goal_reached [std_msgs/Bool] /manipulation/manip_pln/la_goto_loc [std_msgs/String] /navigation/path_planning/simple_move /goal_path [nav_msgs/Path] /navigation/path_planning/simple_move /goal_pose [geometry_msgs/Pose2D]	

Tabla 1.32: Nodo /navigation/mvn_pln

Tópicos publicados	/manipulation/manip_pln/ ra_goto_angles [std_msgs/Float32MultiArray] /hardware/right_arm/goal_gripper [std_msgs/Float32] /manipulation/manip_pln/ ra_pose_wrt_arm [std_msgs/Float32MultiArray]	
Servicios	/navigation/mvn_pln/plan_path	

Tabla 1.33: Nodo /navigation/mvn_pln

Tópicos suscritos	/hardware/robot_state/stop [std_msgs/Empty] /navigation/mvn_pln/get_close_xya [std_msgs/Float32MultiArray] /clicked_point [geometry_msgs/PointStamped] /navigation/mvn_pln/get_close_loc [std_msgs/String] /navigation/localization/current_pose [geometry_msgs/ PoseWithCovarianceStamped] /hardware/scan [sensor_msgs/LaserScan] /hardware/torso/goal_reached [std_msgs/Bool] /tf [tf/tfMessage] /navigation/obs_avoid/obs_in_front [std_msgs/Bool]	

Tabla 1.34: Nodo /navigation/mvn_pln

Tópicos suscritos	/tf_static [tf2_msgs/TFMessage] /manipulation/hd_goal_reached [std_msgs/Bool] /navigation/mvn_pln/add_location [navig_msgs/Location] /manipulation/ra_goal_reached [std_msgs/Bool] /navigation/global_goal_reached [std_msgs/Bool] /navigation/obs_avoid/collision_risk [std_msgs/Bool] /navigation/goal_reached [std_msgs/Bool] /navigation/obs_avoid/collision_point [geometry_msgs/PointStamped] /manipulation/la_goal_reached [std_msgs/Bool]	

Tabla 1.35: Nodo /navigation/mvn_pln

1.4.20. Nodo /navigation/obs_avoid/obstacle_detector

Tópicos publicados	/navigation/obs_avoid/obs_in_front [std_msgs/Bool] /navigation/obs_avoid/collision_risk [std_msgs/Bool] /navigation/obs_avoid/collision_point [geometry_msgs/PointStamped]	
Tópicos suscritos	/hardware/scan [sensor_msgs/LaserScan] /navigation/obs_avoid/enable [std_msgs/Bool] /tf [tf/tfMessage] /tf_static [tf2_msgs/TFMessage] /navigation/mvn_pln/last_calc_path [nav_msgs/Path]	

Tabla 1.36: Nodo /navigation/obs_avoid/obstacle_detector

1.4.21. Nodo /navigation/path_planning/path_calculator

Este nodo se encarga de calcular una ruta y suavizarla desde una pose inicial hasta una pose objetivo utilizando el algoritmo de búsqueda A*, ésto mediante dos servicios de ROS.

Servicios	/navigation/path_planning/ path_calculator/wave_front_from_map [navig_msgs/PathFromMap]	
	/navigation/path_planning/ path_calculator/a_star_from_map [navig_msgs/PathFromMap]	Cálculo de una ruta utilizando el algoritmo de búsqueda A*

Tabla 1.37: Nodo /navigation/path_planning/path_calculator

Sintaxis en un archivo launch

Para correr este nodo sólo se requiere especificar el nombre que se le desea dar al nodo, el paquete en el que se encuentra y el nombre del ejecutable.

```
<node name="path_calculator" pkg="path_calculator" type="path_calculator_node"
output="screen"/>
```

1.4.22. Nodo /navigation/path_planning/simple_move

Tópicos publicados	/hardware/mobile_base/cmd_vel [geometry_msgs/Twist] /hardware/mobile_base/speeds [std_msgs/Float32MultiArray] /hardware/head/goal_pose [std_msgs/Float32MultiArray] /navigation/goal_reached [std_msgs/Bool]	
Tópicos suscritos	/hardware/robot_state/stop [std_msgs/Empty] /navigation/path_planning/simple_move /goal_lateral [std_msgs/Float32] /navigation/path_planning/simple_move /goal_dist [std_msgs/Float32] /navigation/path_planning/simple_move /goal_rel_pose [geometry_msgs/Pose2D]	

Tabla 1.38: Nodo /navigation/path_planning/simple_move

Tópicos suscritos	/navigation/localization/current_pose [geometry_msgs/ PoseWithCovarianceStamped] /tf [tf/tfMessage] /navigation/path_planning/simple_move /goal_dist_angle [std_msgs/Float32MultiArray] /tf_static [tf2_msgs/TFMessage] /navigation/obs_avoid/collision_risk [std_msgs/Bool] /navigation/path_planning/simple_move /goal_path [nav_msgs/Path] /navigation/path_planning/simple_move /goal_pose [geometry_msgs/Pose2D]	

Tabla 1.39: Nodo /navigation/path_planning/simple_move

1.4.23. Nodo /vision/face_recog

Tópicos publicados	/vision/face_recognizer/faces [vision_msgs/VisionFaceObjects] /vision/face_recognizer/trainer_result [std_msgs/Int32]	
Tópicos suscritos	/vision/face_recognizer/start_recog_old [std_msgs/Empty] /vision/face_recognizer/ run_face_recognizer_id [std_msgs/String] /vision/face_recognizer/ run_face_trainer_frames [vision_msgs/VisionFaceTrainObject] /vision/face_recognizer/clearfacesdb [std_msgs/Empty] /vision/face_recognizer/clearfacesdbbyid [std_msgs/String]	

Tabla 1.40: Nodo /vision/face_recog

Tópicos suscritos	/vision/face_recognizer/ run_face_recognizer [std_msgs/Empty] /vision/face_recognizer/run_face_trainer [std_msgs/String] /vision/face_recognizer/stop_recog [std_msgs/Empty] /vision/face_recognizer/start_recog [std_msgs/Empty]	

Tabla 1.41: Nodo /vision/face_recog

1.4.24. Nodo /vision/line_finder

Tópicos suscritos	/hardware/head/current_pose [std_msgs/Float32MultiArray]	
Servicios	/vision/line_finder/find_lines_ransac	

Tabla 1.42: Nodo /vision/line_finder

1.4.25. Nodo /vision/obj_reco

Tópicos publicados	/vision/obj_reco/recognizedObjectes [vision_msgs/VisionObjectList]	
Tópicos suscritos	/vision/obj_reco/enableRecognizeTopic [std_msgs/Bool] /vision/obj_reco/enableDetectWindow [std_msgs/Bool] /hardware/point_cloud_man/ rgbd_wrt_robot []	
Servicios	/vision/obj_reco/det_objs /vision/geometry_finder/findPlane /vision/obj_reco/trainObject	

Tabla 1.43: Nodo /vision/obj_reco

1.4.26. Nodo /vision/skeleton_finder

Tópicos publicados	/vision/skeleton_finder/skeletons [vision_msgs/Skeletons]	
Tópicos suscritos	/vision/skeleton_finder/start_recog [std_msgs/Empty] /vision/skeleton_finder/stop_recog [std_msgs/Empty]	

Tabla 1.44: Nodo /vision/skeleton_finder