

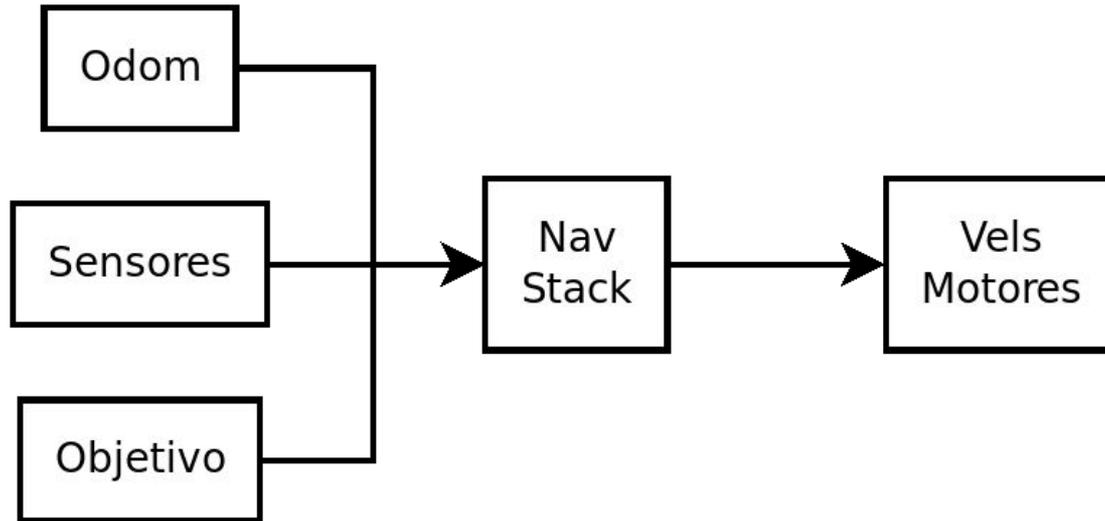
# Stack de navegación en ROS

<http://wiki.ros.org/navigation>

# Introducción

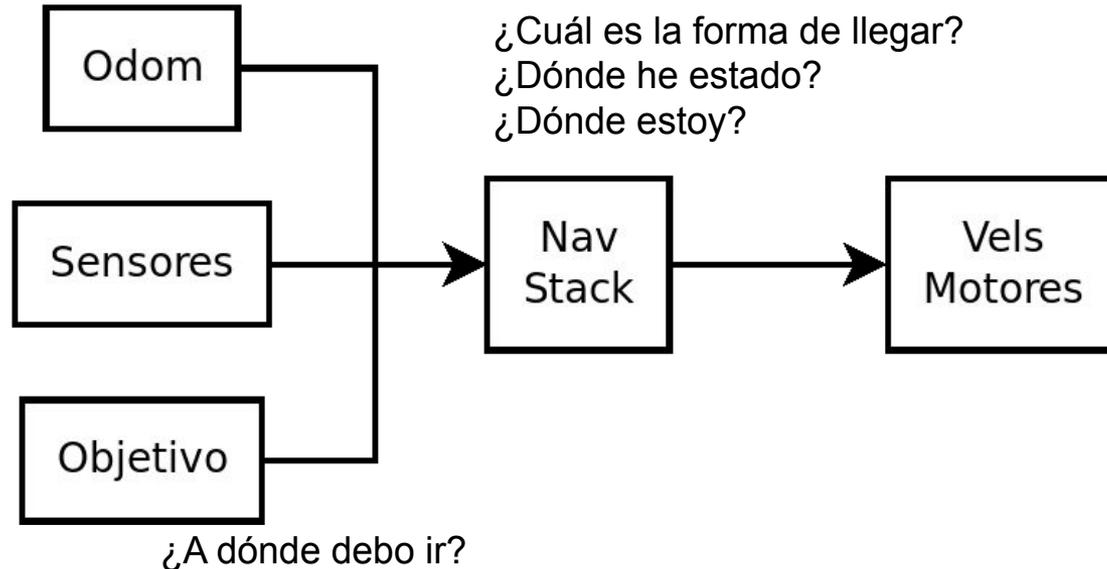
# A vista de pájaro

“A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.”



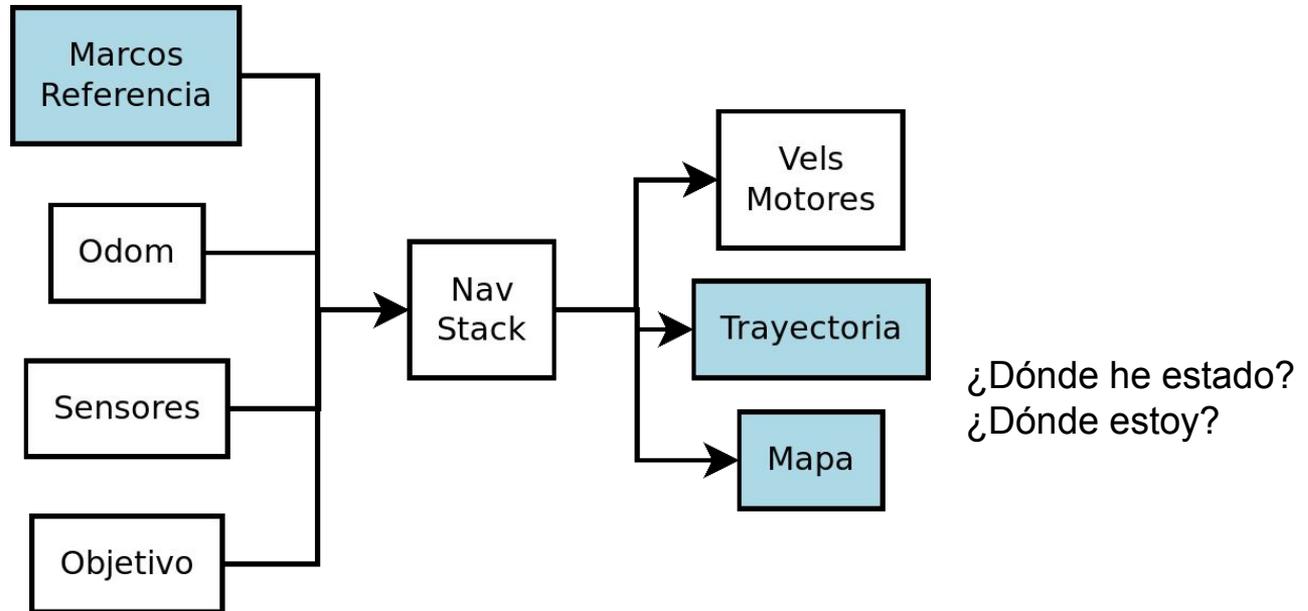
# A vista de pájaro

“A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.”



# A vista de pájaro

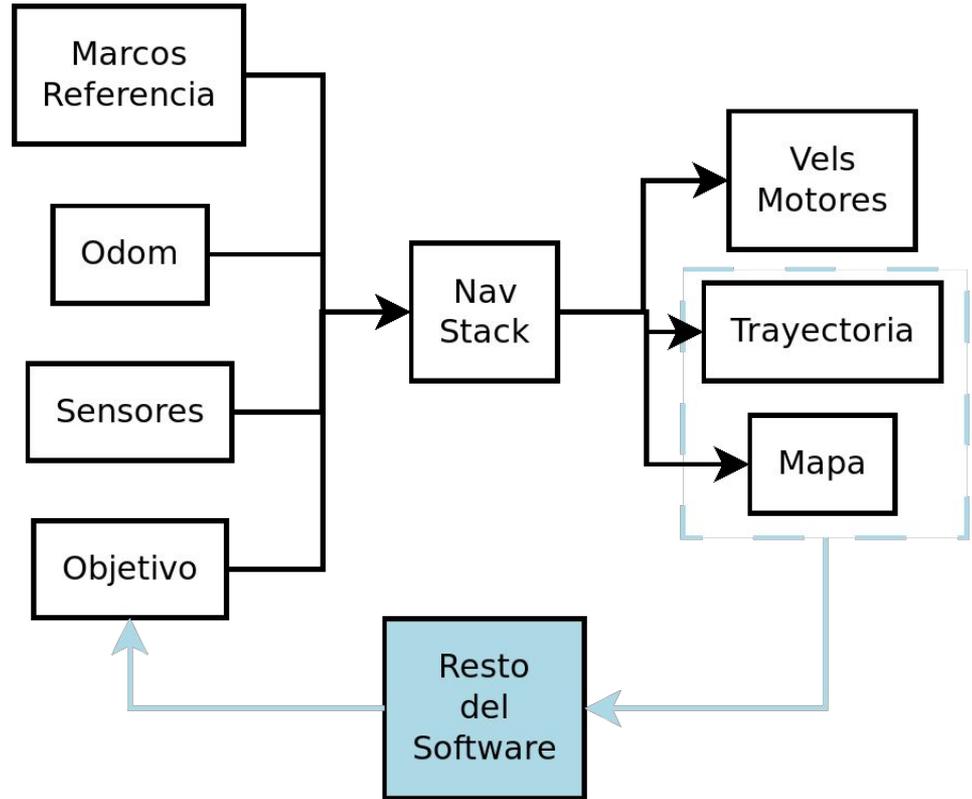
“A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.”



# Integración con el resto del sistema

El mapa y trayectoria informan al resto del software para decidir el objetivo

El stack de navegación abstrae todo menos la decisión del objetivo, e informa a esta última.



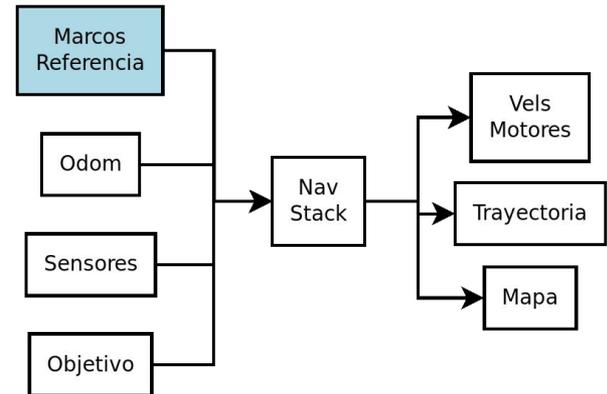
# Ambiente de demos

1. all.launch
2. mostrar gazebo
3. mostrar rviz

# Lecturas

<http://wiki.ros.org/navigation>

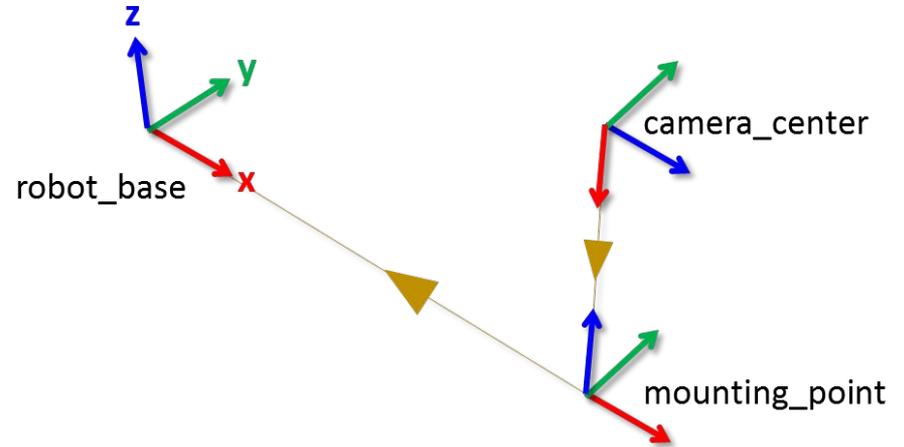
# Marcos de Referencia



# Introducción

Marco de referencia:

- Un nombre para un sistema de ejes en el espacio
- Una transformada desde otro marco de referencia

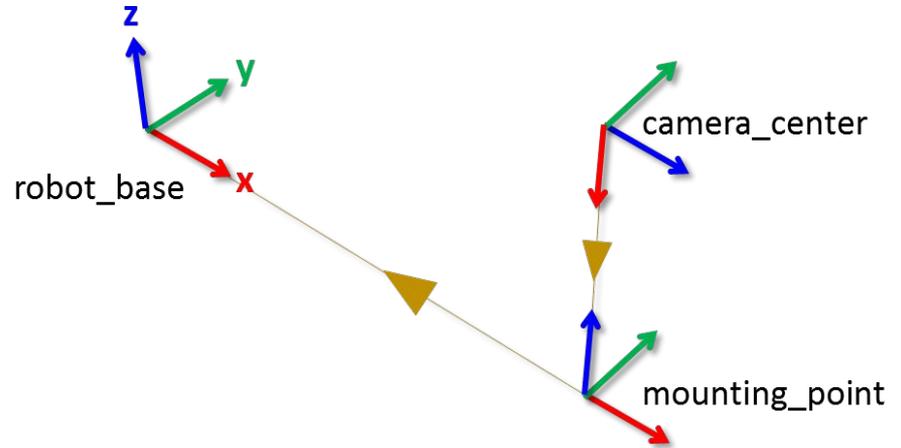


# TF

ROS implementa una forma estándar de manejar marcos de referencia: TF.

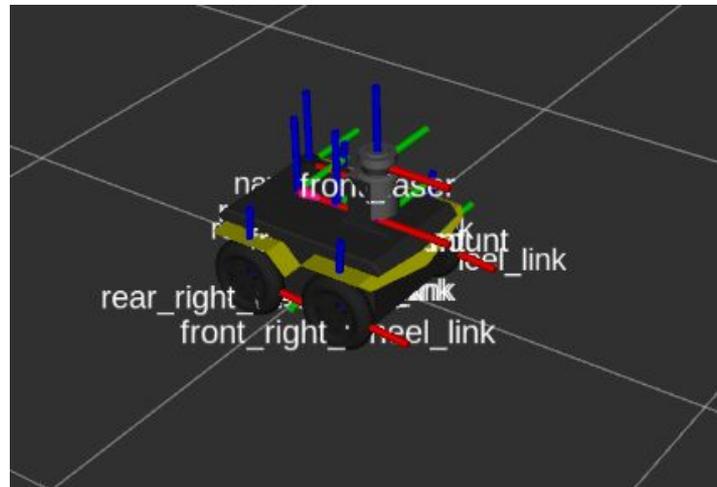
TF se encarga de:

- Computar la transformada entre dos marcos arbitrarios
- Mantener un buffer de transformadas entre pares de marcos
- Utilidades de debugging (p.e. Árbol de transformadas)
- Hacer todo esto de manera performante y “sin errores”



# Demo

1. all.launch
2. rviz
3. Quitar todo menos modelo
4. Agregar tf
5. view\_frames
  - a. `roslaunch tf view_frames`
6. Publicar un nuevo par
  - a. `roslaunch tf static_transform_publisher .1 .1 .1 0 0 0 front_laser nueva 10`
  - b. ir a rviz
  - c. republicar el arbol



# robot\_state\_publisher

Facilita la publicación de todos los pares de transformadas

Toma un modelo Unified Robot Description Format (URDF) del robot

[http://wiki.ros.org/robot\\_state\\_publisher](http://wiki.ros.org/robot_state_publisher)

```
82 <xacro:wheel prefix="front_left">
83   <origin xyz="{wheelbase/2} ${track/2} $
{wheel_vertical_offset}" rpy="0 0 0" />
84 </xacro:wheel>
85 <xacro:wheel prefix="front_right">
86   <origin xyz="{wheelbase/2} ${-track/2} $
{wheel_vertical_offset}" rpy="0 0 0" />
87 </xacro:wheel>
88 <xacro:wheel prefix="rear_left">
89   <origin xyz="{-wheelbase/2} ${track/2} $
{wheel_vertical_offset}" rpy="0 0 0" />
90 </xacro:wheel>
91 <xacro:wheel prefix="rear_right">
92   <origin xyz="{-wheelbase/2} ${-track/2} $
{wheel_vertical_offset}" rpy="0 0 0" />
```

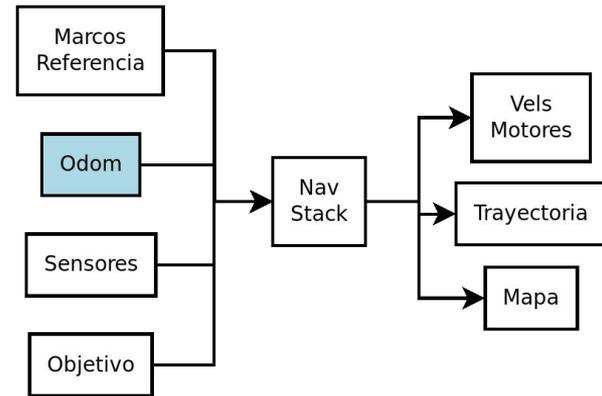
# Lecturas

<http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF>

<http://wiki.ros.org/tf/Tutorials>

[http://wiki.ros.org/robot\\_state\\_publisher](http://wiki.ros.org/robot_state_publisher)

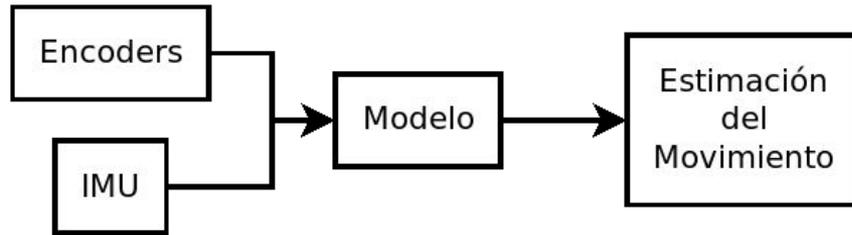
# Odometría



# Introducción

Odometría: información del movimiento del robot, según fuentes de información propia (p.e. encoders)

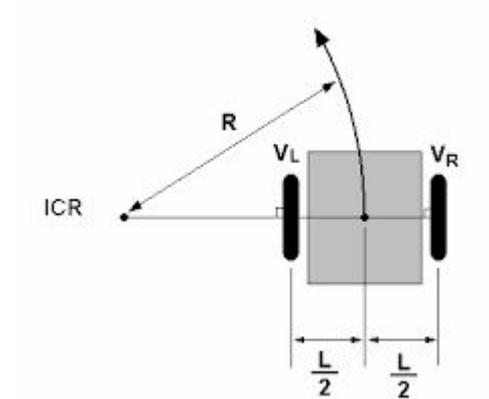
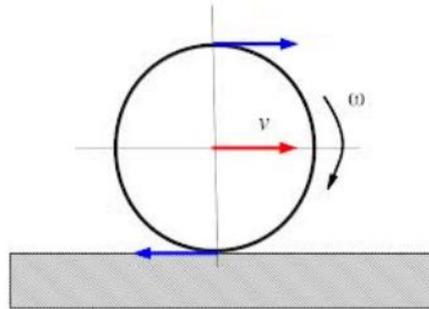
Asumen hipótesis que no siempre se cumplen (p.e. no deslizamiento de las ruedas)



# Modelos

Las hipótesis de trabajo y morfología del robot determinan el modelo a utilizar para estimar:

- diferencial
- ackerman (bicicletas/autos)
- skid-steer (pala mecánica)
- holonómico



# Convenciones de ROS

Dos formas de publicar odometría

- Mensaje nav\_msgs/Odometry
- Marco TF /odom

TF:

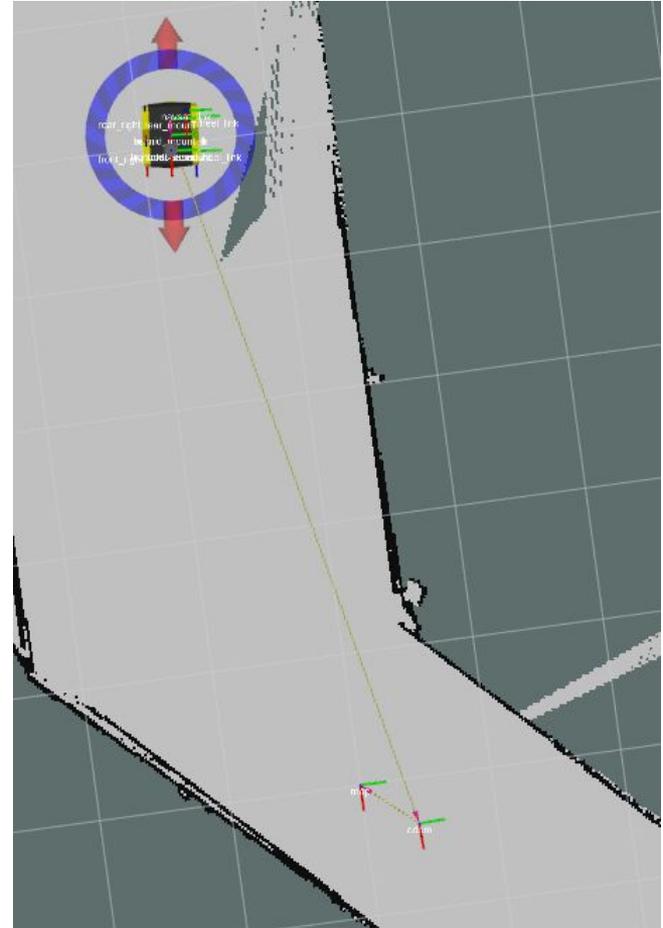
- map: el marco global
- odom: un marco donde se expresa el movimiento continuo del robot
- base\_link: el marco principal del robot

map -> odom -> base\_link:

- map->base\_link: localización global, integrando toda la info, con saltos
- odom->base\_link: localización continua, integrando solo odom, sin saltos

# Demo

1. all.launch
2. gazebo
3. mostrar
  - a. tf
  - b. interactive markers
  - c. robot model
4. manejar por ahi, rotando
5. observar drift entre map y odom

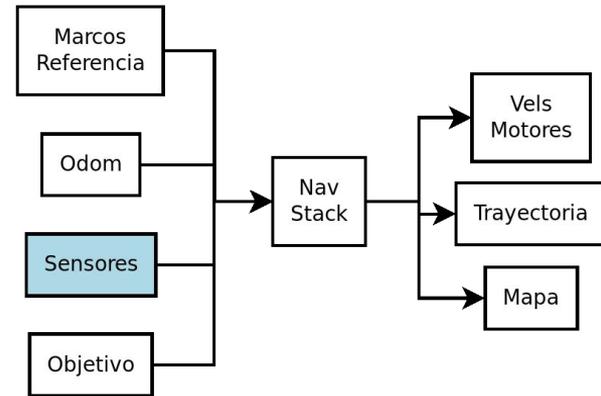


# Lecturas

<http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom>

<https://answers.ros.org/question/37029/is-odom-frame-in-this-wiki-mean-a-world-coordinate/>

# Sensores



# Introducción

Sensores se publican usualmente como mensajes

Aspectos importantes:

- timestamp para sincronización
- asociados a un marco de referencia

Ejemplos:

- [http://docs.ros.org/en/melodic/api/sensor\\_msgs/html/msg/Range.html](http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Range.html)
- [https://docs.ros.org/en/api/sensor\\_msgs/html/msg/LaserScan.html](https://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html)
- [http://docs.ros.org/en/melodic/api/sensor\\_msgs/html/msg/Image.html](http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Image.html)

# Ejemplo: LaserScan

## sensor\_msgs/LaserScan Message

---

File: `sensor_msgs/LaserScan.msg`

### Raw Message Definition

---

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header          # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        #
                        # in frame frame_id, angles are measured around
                        # the positive Z axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min      # start angle of the scan [rad]
float32 angle_max      # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points
float32 scan_time      # time between scans [seconds]

float32 range_min      # minimum range value [m]
float32 range_max      # maximum range value [m]

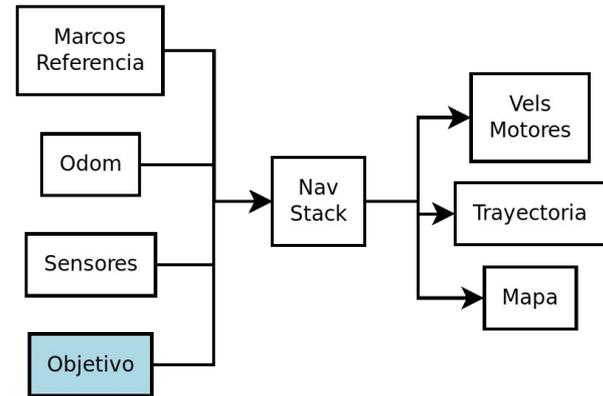
float32[] ranges       # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities  # intensity data [device-specific units]. If your
                        # device does not provide intensities, please leave
                        # the array empty.
```



# Lecturas

- [http://docs.ros.org/en/melodic/api/sensor\\_msgs/html/msg/Range.html](http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Range.html)
- [https://docs.ros.org/en/api/sensor\\_msgs/html/msg/LaserScan.html](https://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html)
- [http://docs.ros.org/en/melodic/api/sensor\\_msgs/html/msg/Image.html](http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Image.html)

# Objetivo



# Introducción

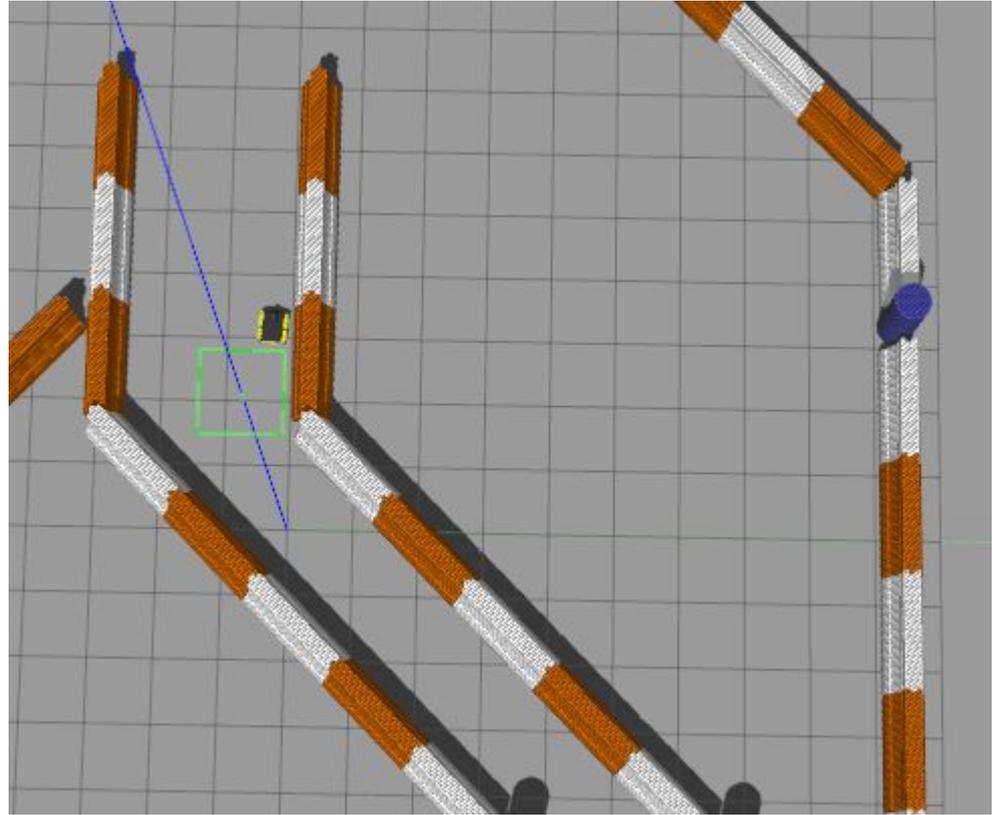
El objetivo especifica la posición deseada

Publicable desde rviz o de forma programática

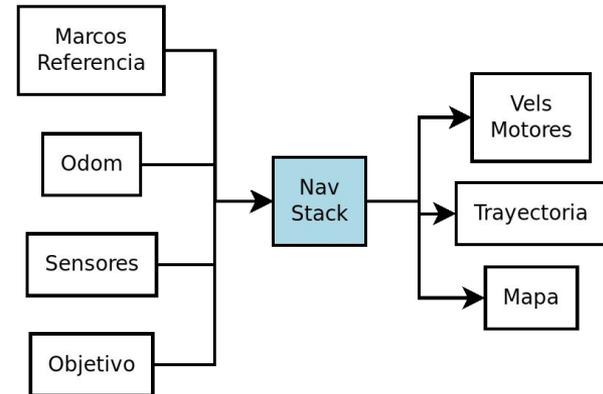


# Demo

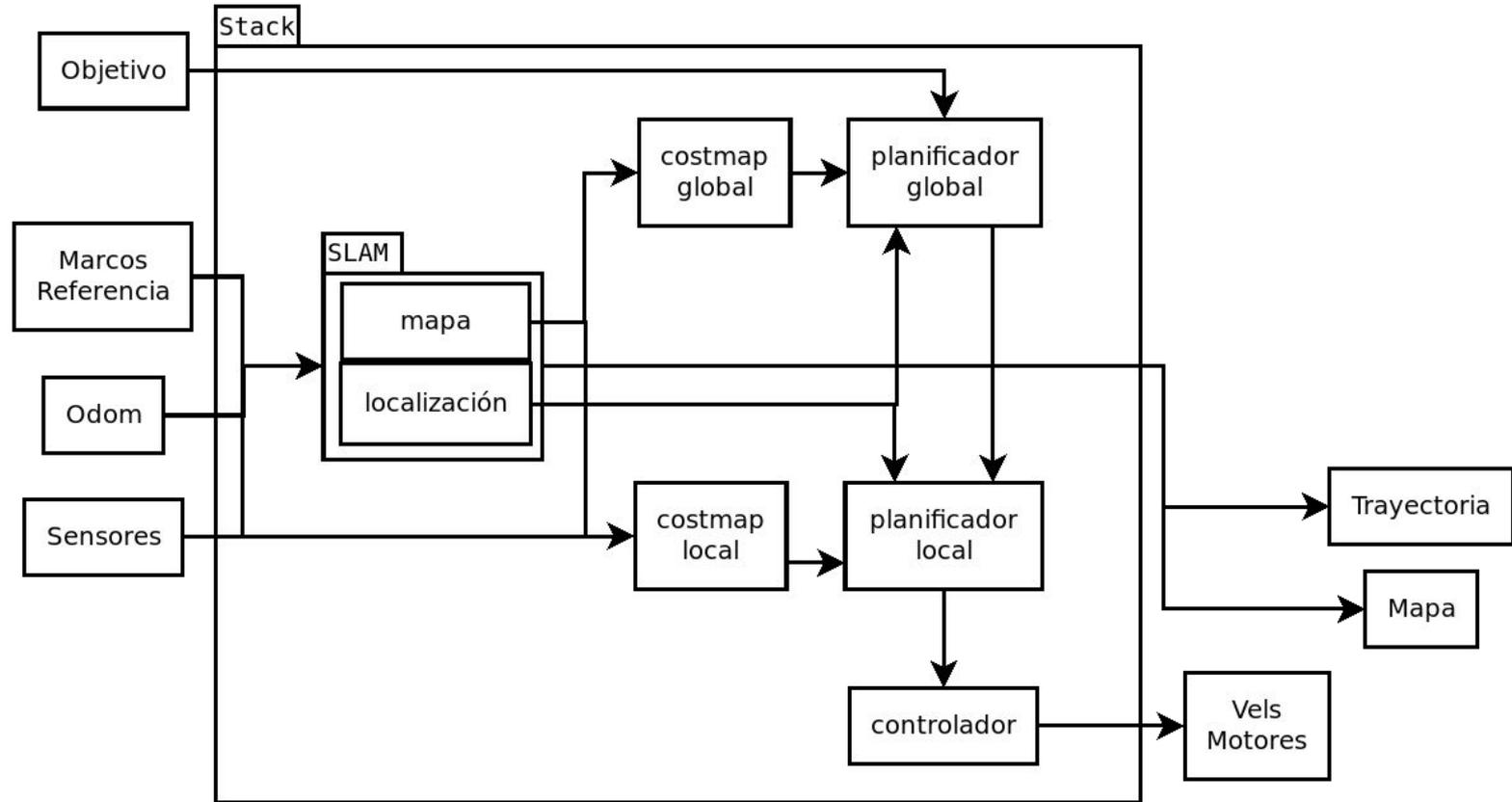
1. all.launch
2. Usar 2D Nav Goal
3. observar en Gazebo



# Adentro del Stack



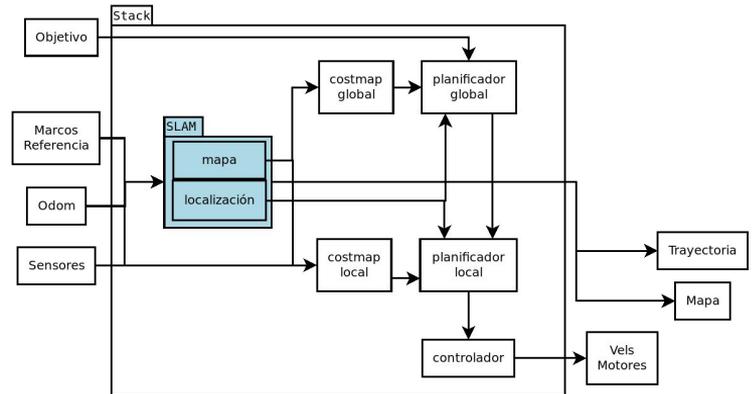
# Organización general



# Lecturas

[https://www.researchgate.net/publication/302986850\\_ROS\\_Navigation\\_Concepts\\_and\\_Tutorial](https://www.researchgate.net/publication/302986850_ROS_Navigation_Concepts_and_Tutorial)

# SLAM



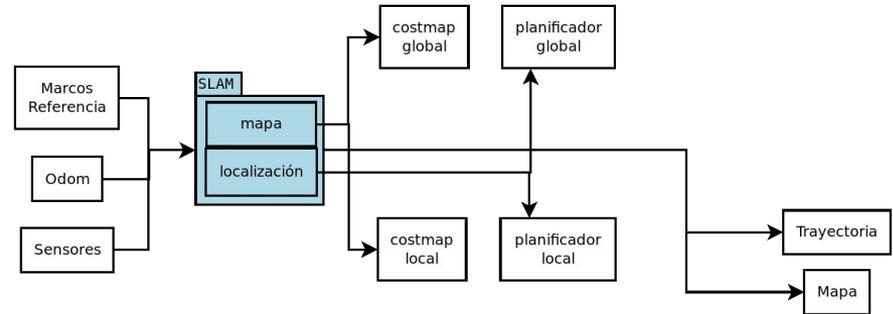
# Organización general

Encargado de

- Publicar un mapa de ocupación
  - Una discretización del entorno que lo divide en celdas ocupadas y libres
- Publicar la ubicación del robot en el mapa

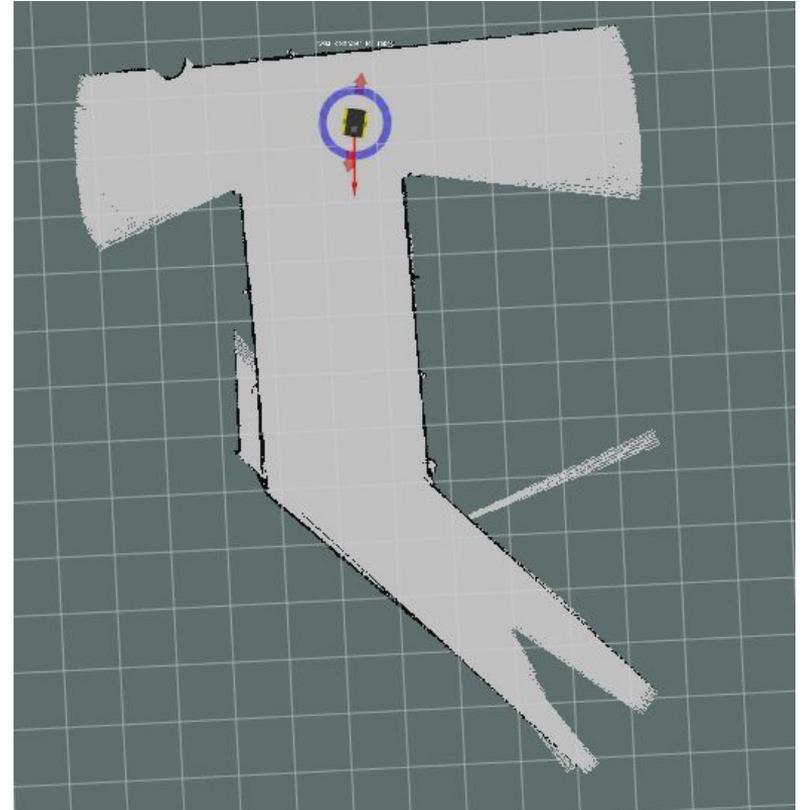
El módulo de SLAM puede generar el mapa (p.e. gmapping), o puede cargar uno previo y solo realizar localización sobre este (p.e. AMCL)

Cierre de ciclos: detectar un lugar antes visitado y ajustar el mapa de acuerdo a esta nueva información

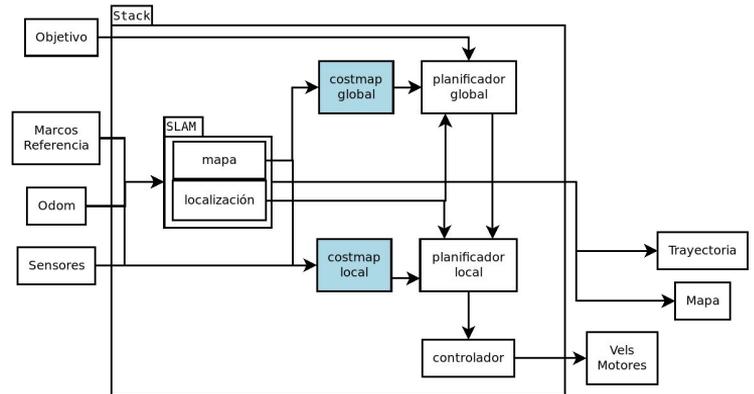


# Demo

1. all.launch
2. marcar
  - a. Maps (/map)
  - b. Laser scan
3. establecer un objetivo de navegación y mirar el mapa



# Costmaps



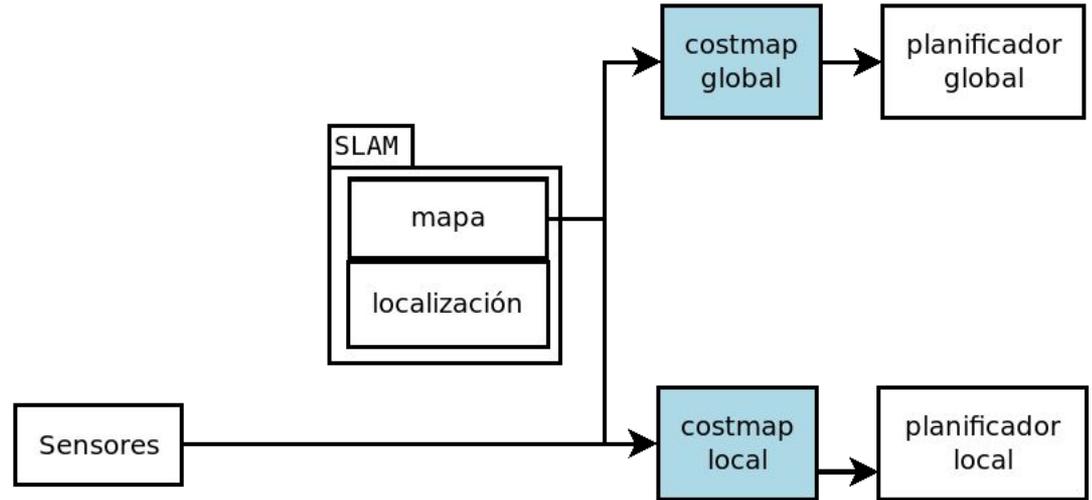
# Mapas de costo

Toman como entrada el mapa

Su salida es un nuevo mapa con costos de navegación

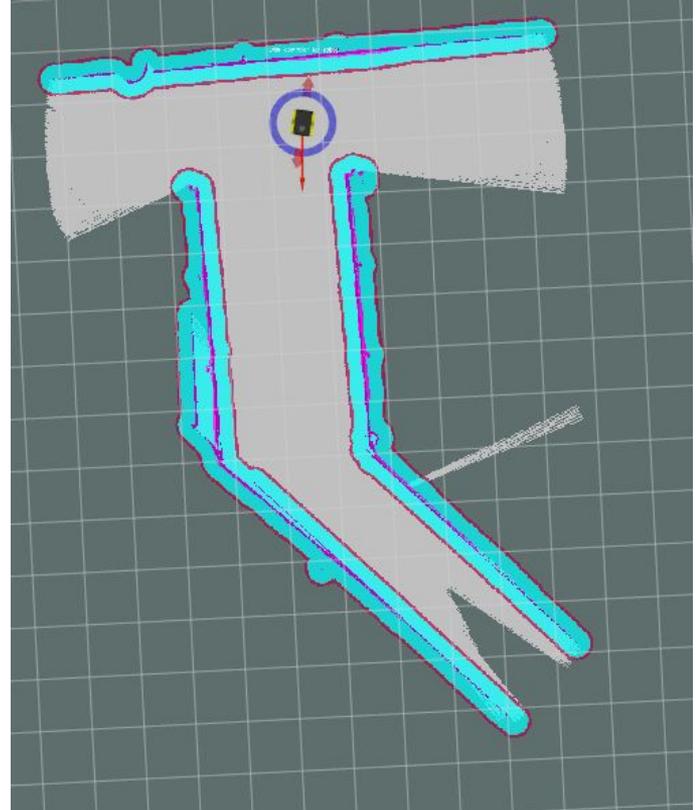
Son el principal insumo del planificador de rutas

Los obstáculos corresponden a costos altos. También se inflan para aumentar la distancia a estos durante la navegación



# Demo

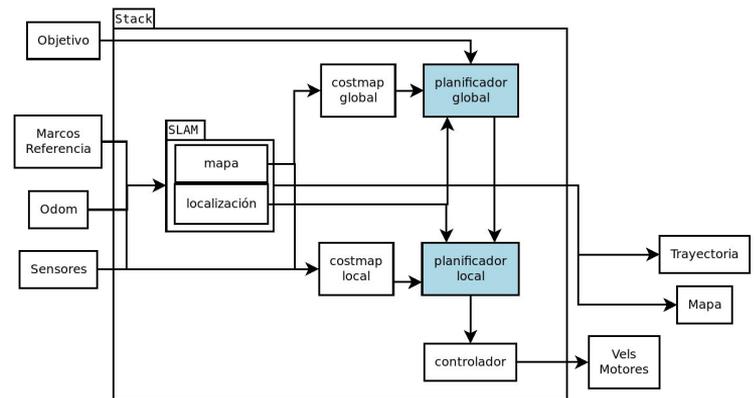
1. all.launch
2. marcar
  - a. costmap global
  - b. mapa
  - c. robot model
3. establecer un objetivo de navegación y mirar el mapa de costos
4. correr rqt\_reconfigure
  - a. Cambiar inflation radius



# Lecturas

- [http://wiki.ros.org/costmap\\_2d](http://wiki.ros.org/costmap_2d)
- [http://wiki.ros.org/costmap\\_2d/hydro/inflation](http://wiki.ros.org/costmap_2d/hydro/inflation)

# Planificadores



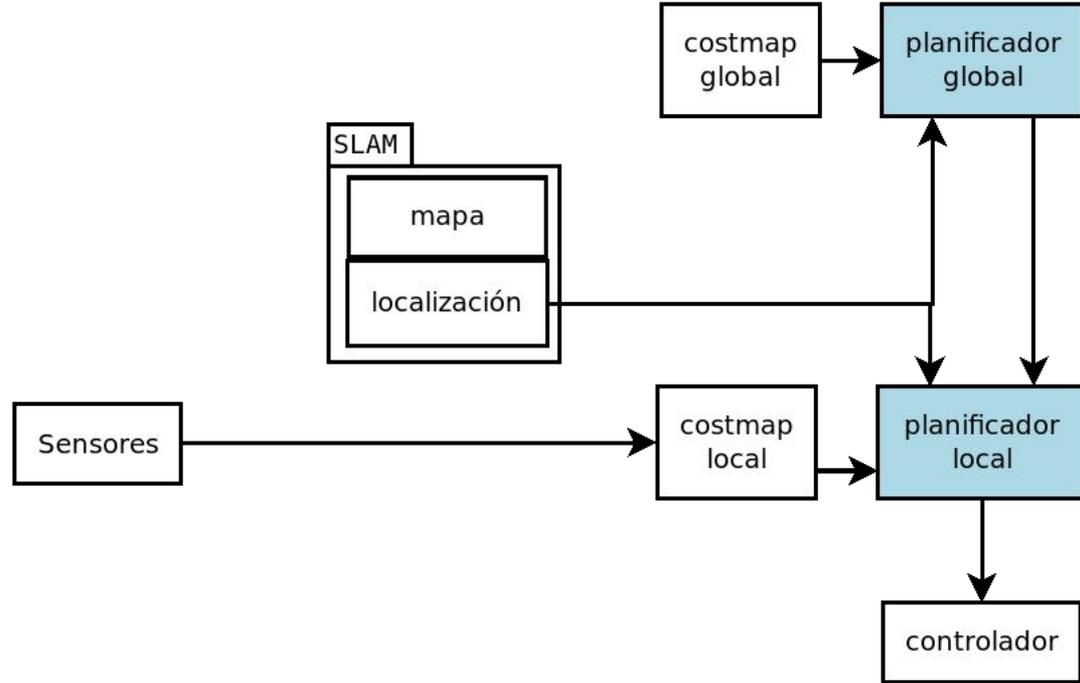
# Planificadores de rutas

Toman como entrada el mapa de costos y localización

Su salida es una ruta a seguir

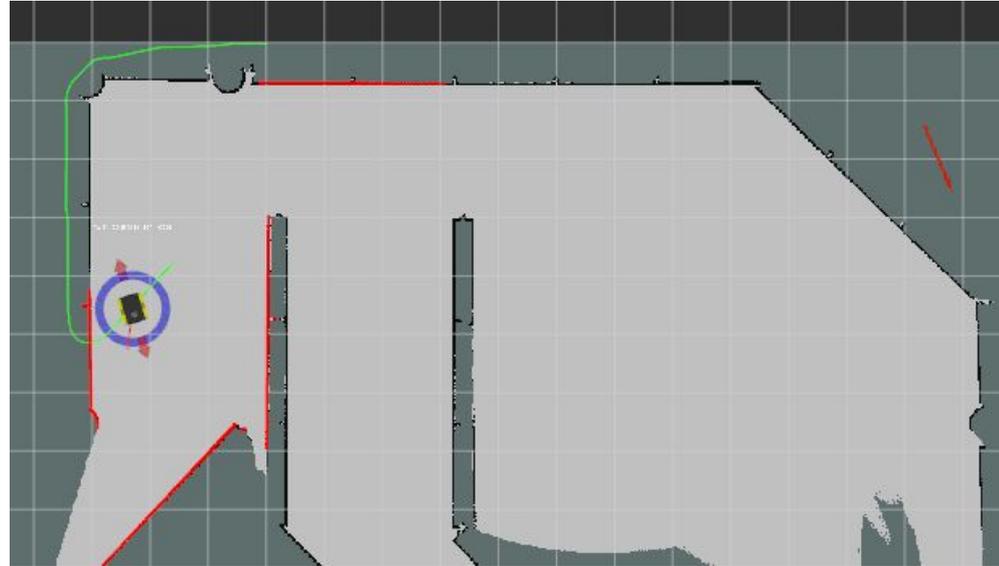
El planificador global observa todo el mapa y produce una ruta global a seguir a largo plazo

El planificador local observa solo un entorno, incluye sensores y produce una ruta local a seguir de manera inmediata



# Demo

1. all.launch
2. marcar
  - a. costmap global
  - b. mapa
  - c. robot model
  - d. Planes local y global
3. establecer un objetivo de navegación y mirar el mapa de costos
4. Observar las rutas
5. Establecer un punto afuera del mapa

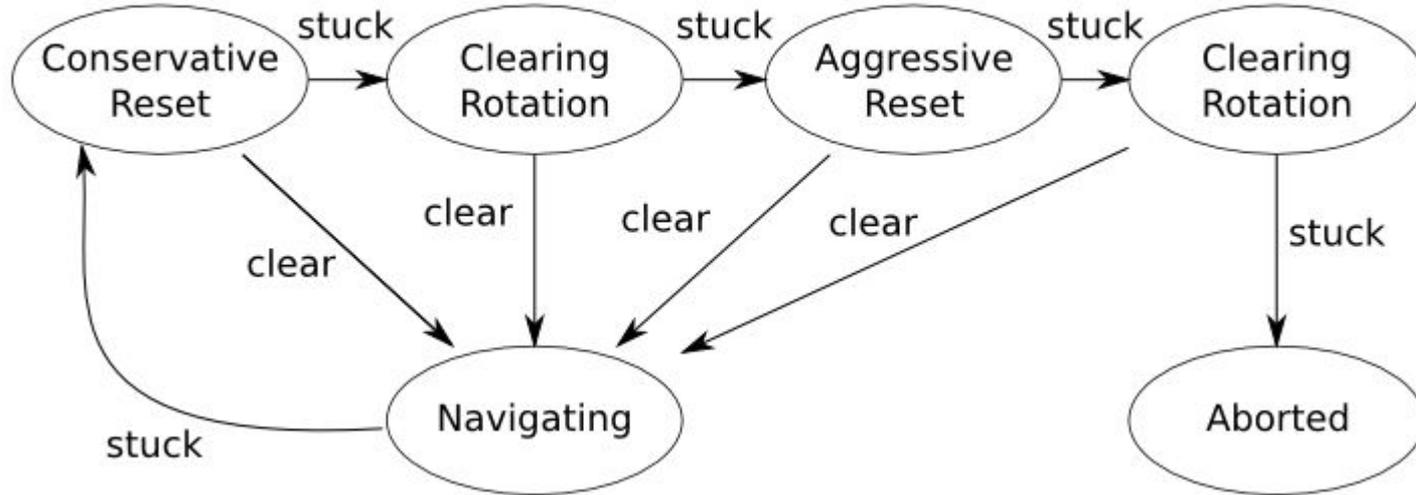


# Lecturas

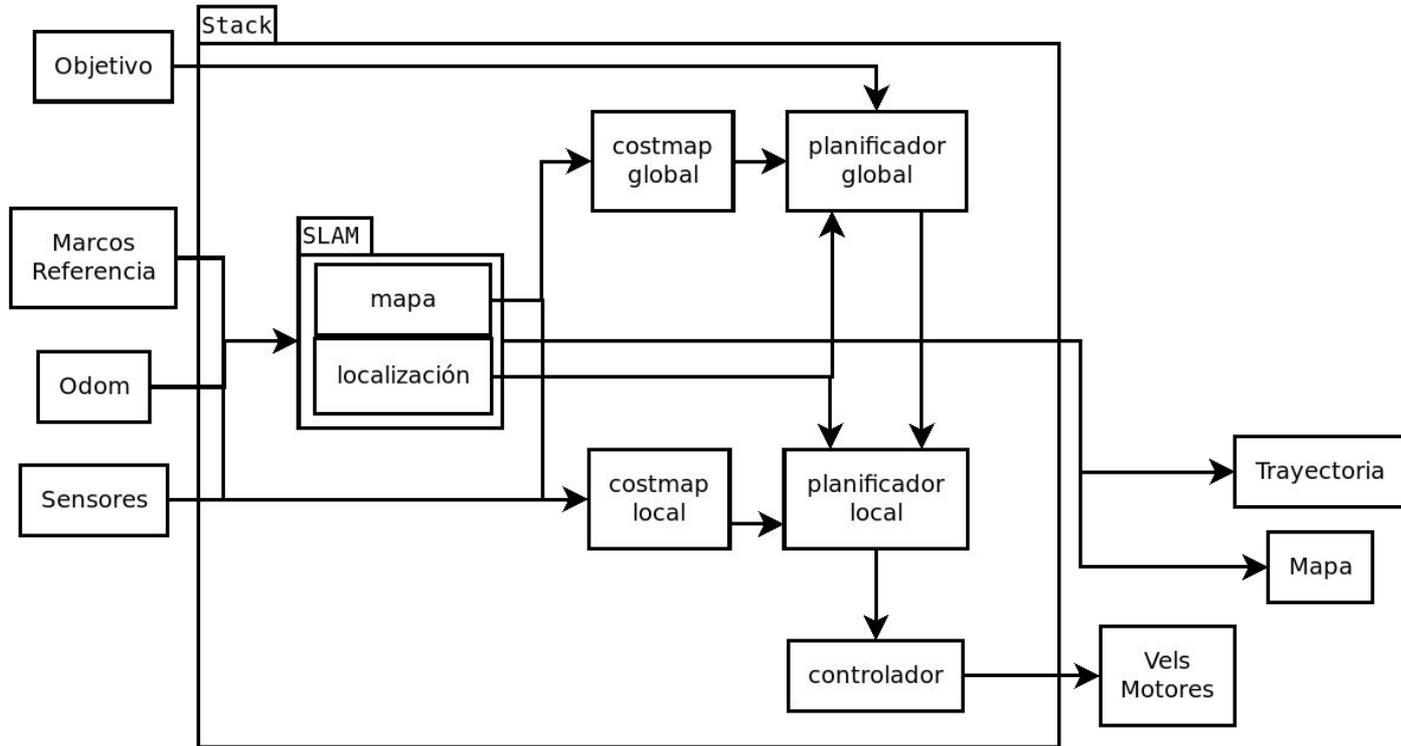
- [http://wiki.ros.org/global\\_planner](http://wiki.ros.org/global_planner)

# Comportamientos de recuperación

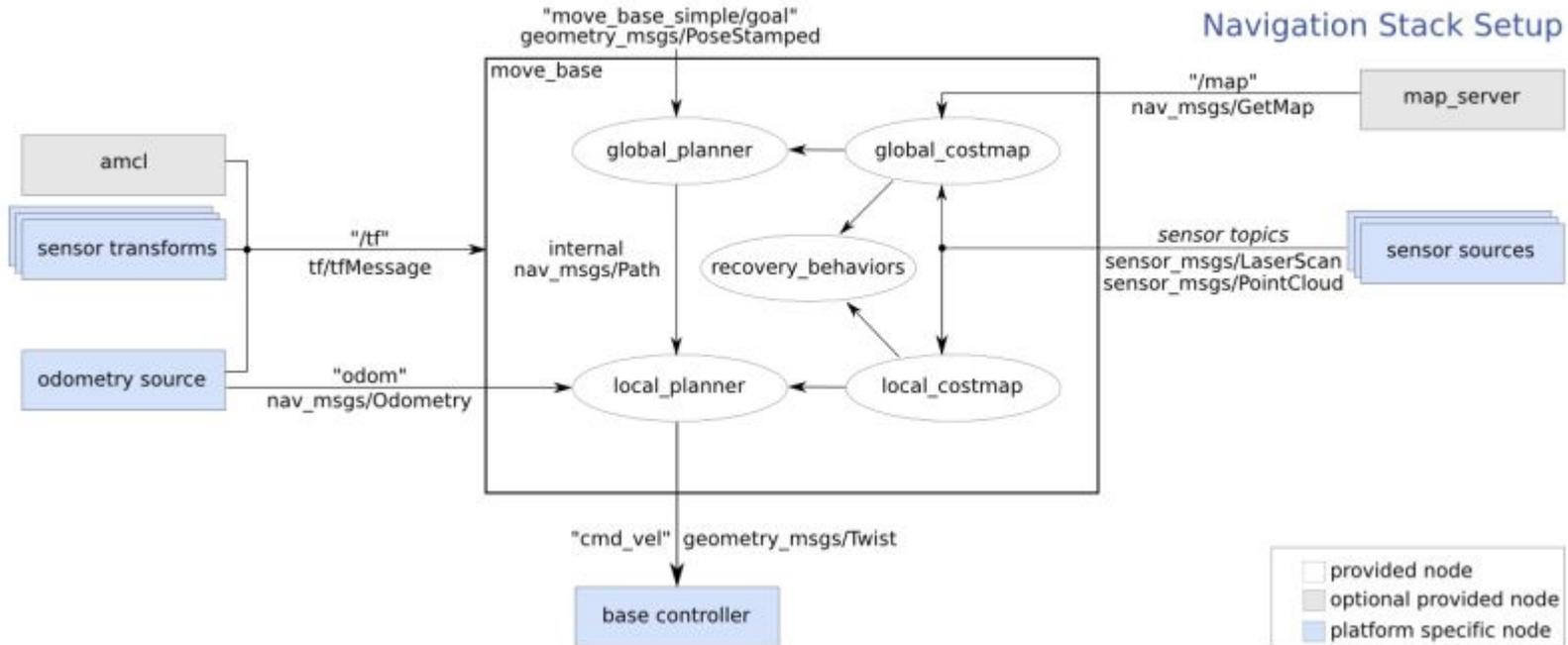
move\_base Default Recovery Behaviors



# Repaso



# Comportamientos de recuperación



# Lecturas

- [http://wiki.ros.org/move\\_base](http://wiki.ros.org/move_base)