Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

Centro de Investigación en Informática para la Ingeniería

ROS – Robot Operating System

- Framework de Desarrollo en Robótica
- Ofrece herramientas y librerías para el desarrollo de sistemas robóticos
- Infraestructura para desarrollo y ejecución
- Basado en herramientas

ROS – Características

- Abstracción de Hardware
- Independiente del lenguaje (c++ ,python ,octave ,java)
- Sistema Distribuido
- Módulos reutilizables
- Licencia BSD

ROS – Proporciona

- Algoritmos de bajo y alto nivel
- Herramientas de desarrollo y simulación
- Integración (OpenCV, Player, Kinect, etc)
- Drivers para diferentes robots
- Topología p2p
- Herramientas de linea de comando.
- Encapsulamiento de proyectos en paquetes o stacks

ROS – ¿Quien lo utiliza?

- Industria:

  Willow Garage, Bosch R&D, Vanadium Labs
- Universidades:

  Brown, CCNY, TUM, Berkeley, USC, Rice, MIT,

  UMD, GT, SAIL, CMU, UPenn Grasp.
- Robots: youBot (KUKA)

Robots que utilizan ROS












ROS - INSTALACION

Select Your Platform

Supported:

Ubuntu

Experimental:

OS X

Arch

Fedora

Gentoo

OpenSUSE

Slackware

Debian

Or, Select your robot

Robots:

AscTec Pelican/Hummingbird

TurtleBot

Care-O-bot

PR2

Erratic

Shadow Robot

Lego NXT

Partial functionality:

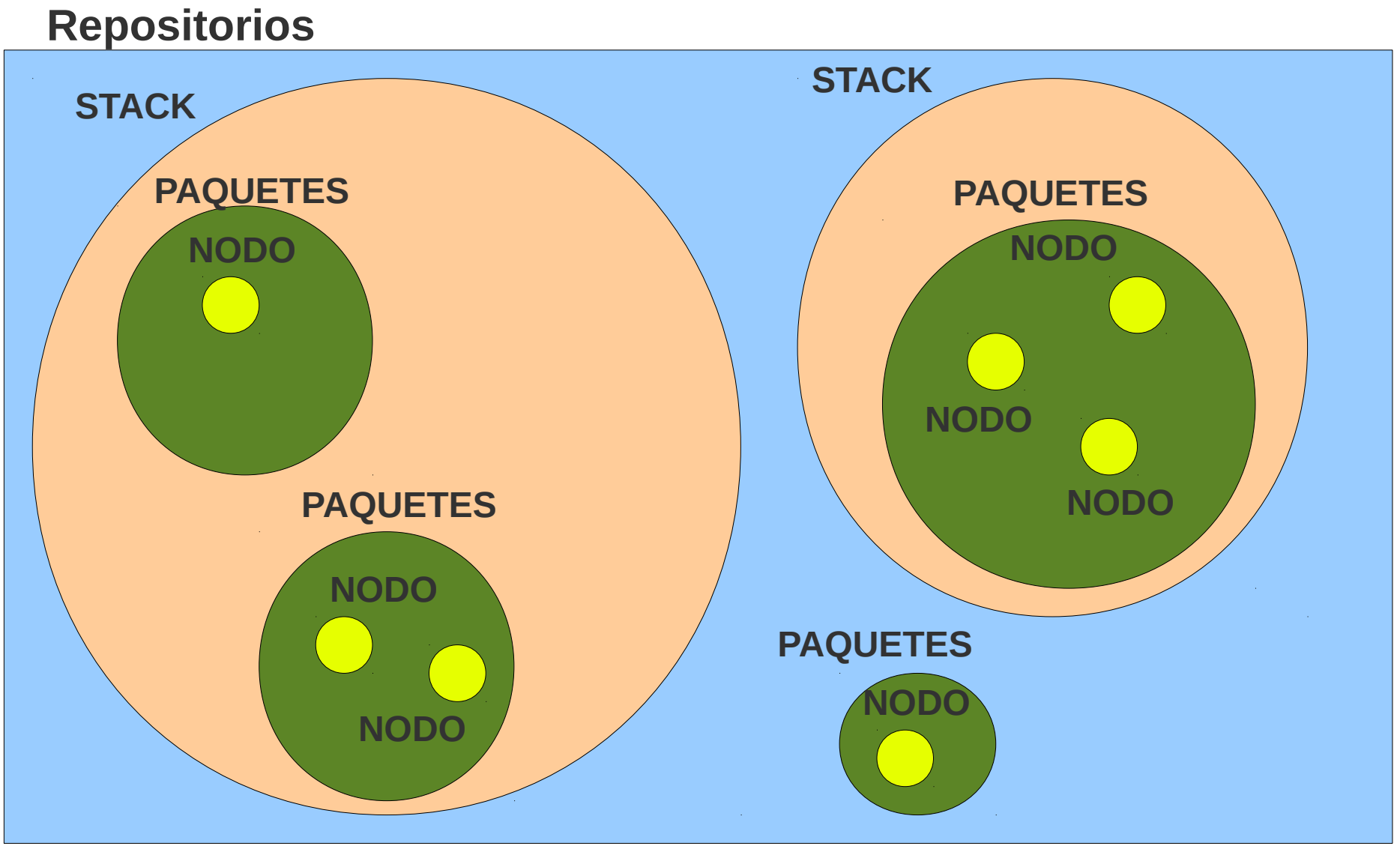
Windows

FreeBSD

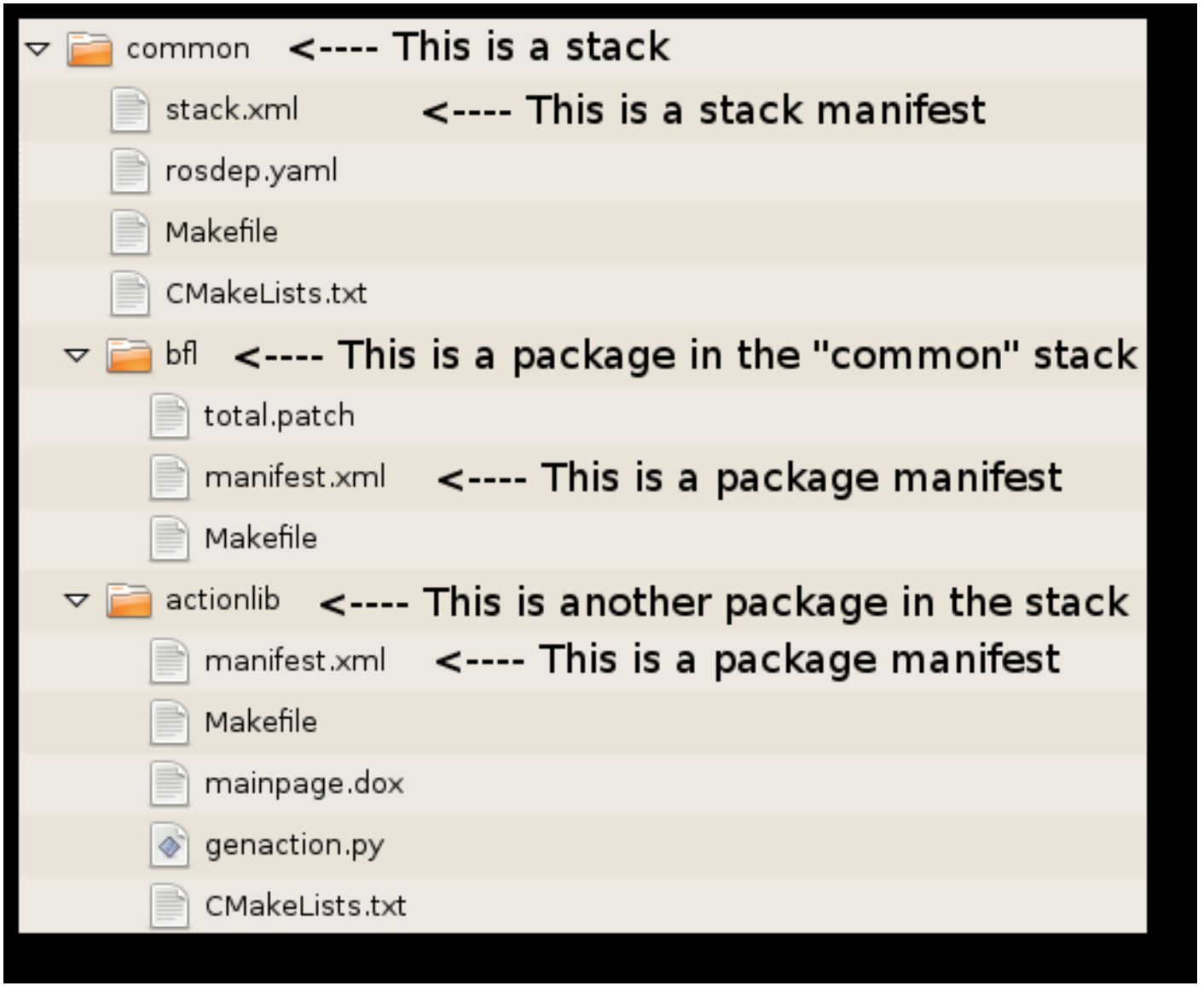
ROS - Nomenclatura

- **Nodos:** Procesos que ejecutan una acción. Modulo de software.
- **Mensajes:** Comunicación entre nodos. Es una estructura estricta de datos.
- **Tópico:** Un nodo envía un mensaje publicándolo en un tópico.
- **Servicios:** Definido por un nombre y por tipo de datos. Comunicación síncrona.

Sistema de archivos



Sistema de Archivos

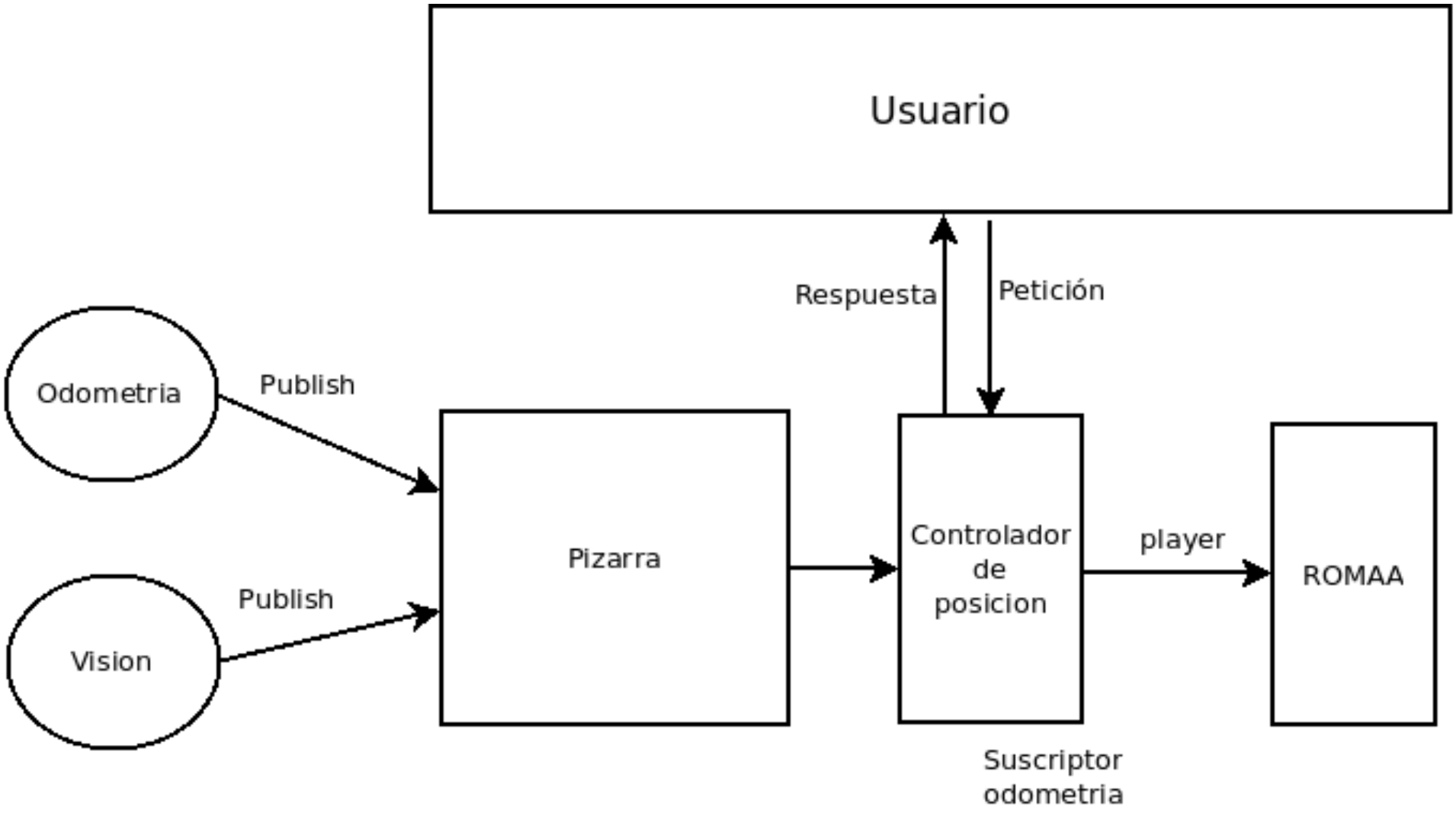


Sistema de Archivos

- Navegación
  - rospack find pkg_name
  - rosstack find [stack_name]
  - roscd pkg_name
  - rosls pkg_name

export ROS_PACKAGE_PATH=directorio_de_trabajo:$ROS_PACKAGE_PATH

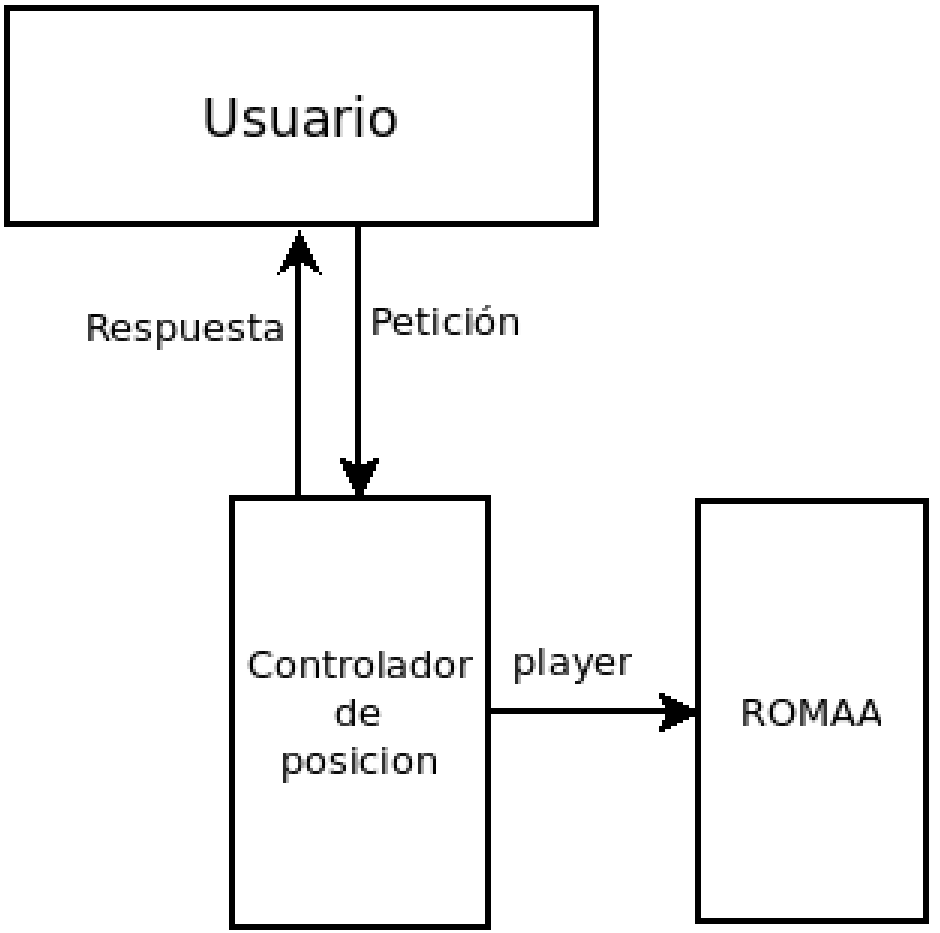
ROS y ROMAA


Usuario
Respuesta
Petición
Odometria
Publish
Vision
Publish
Pizarra
Controlador de posicion
player
ROMAA
Suscriptor odometria

ROS y ROMAA

- Servicio constituido por
  - Petición: Posición destino del robot
  - Respuesta: Posición final del robot
- ¿Que datos paso?

  Los que defino como mensajes



Mensajes y servicios en ROS

Mensaje

- Se define un archivo de texto que describe el tipo de dato. Esto generá el codigo fuente necesario en diferentes lenguajes.
- Se debe incluir la dependencia std_msgs al crear el paquete.

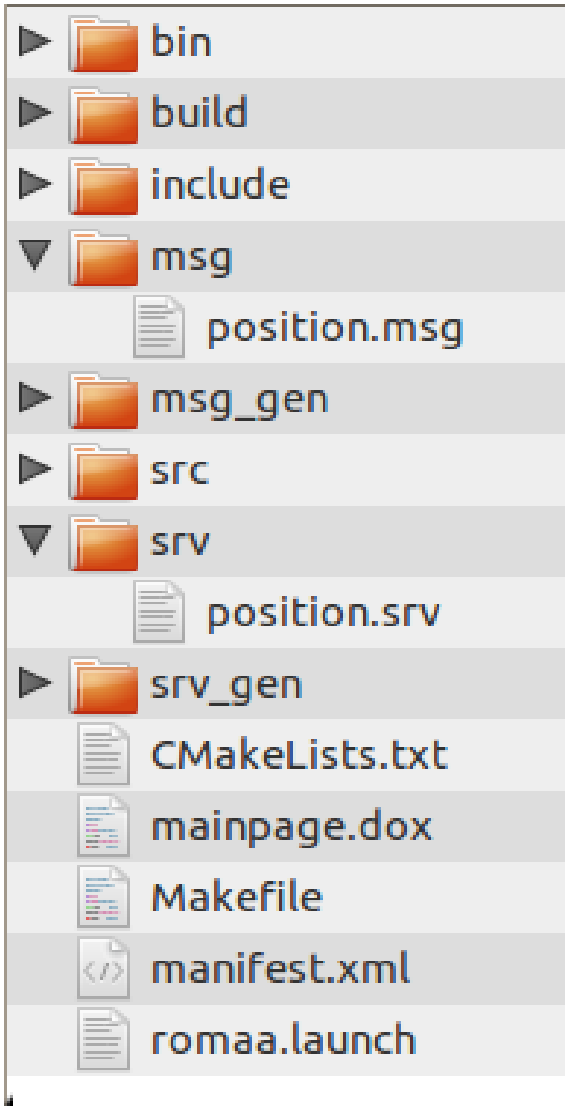
Servicio

- Se define un archivo de texto que describe la petición y la respuesta.

ROS - MENSAJES

- Mensajes Definidos en nombre_paquete/msg/*.msj
- Tipos basicos
  - int{8,16,32,64}
  - float{32,64}
  - String, time, duration, array[]
- Servicios definidos en nombre_paquete/srv/*.srv

Mensajes y Servicios en ROS



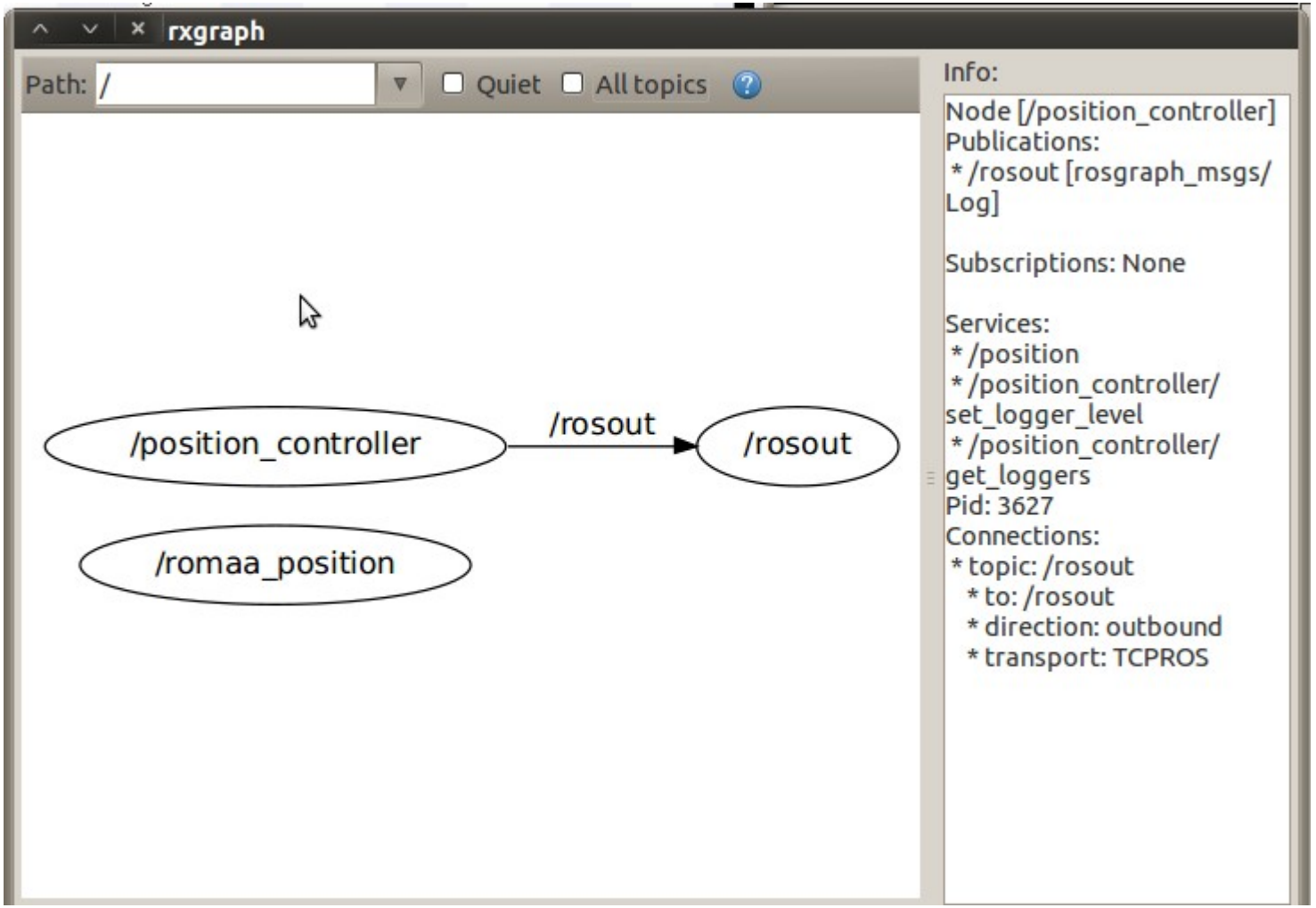
Position.srv

```
float64 x0
float64 y0
---
float64 x1
float64 y1
```

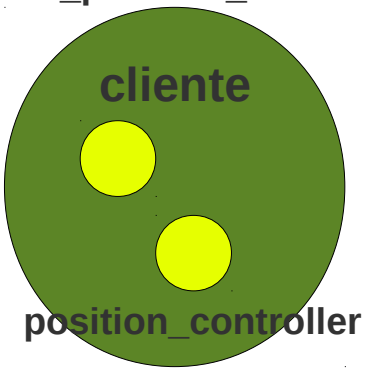
Position.msg

```
float64 num
```

ROS y ROMAA



PAQUETE:
romaa_position_controler



romaa_position_controller
nodo cliente

```
int main(int argc, char **argv)
{

  ros::init(argc, argv, "romaa_position");

  ros::NodeHandle n;
  ros::ServiceClient client = n.serviceClient<romaa_position_controler::position>("position");

  romaa_position_controler::position srv;

  srv.request.x0 = atof(argv[1]);
  srv.request.y0 = atof(argv[2]);

  if (client.call(srv))
  {
    ROS_INFO("x %f y %f", srv.response.x1, srv.response.y1);
  }
  return 0;
}
```

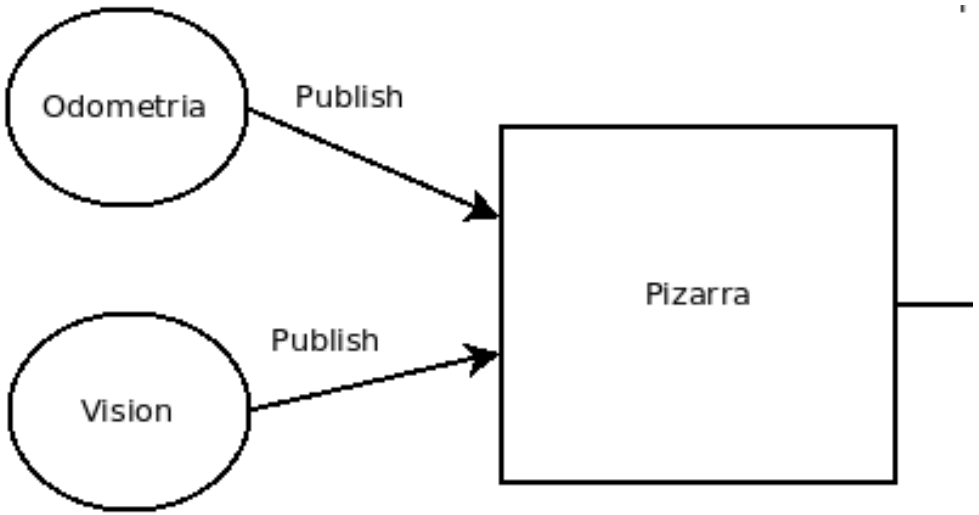
Codigo abreviado

romaa_position_controller nodo position_controller

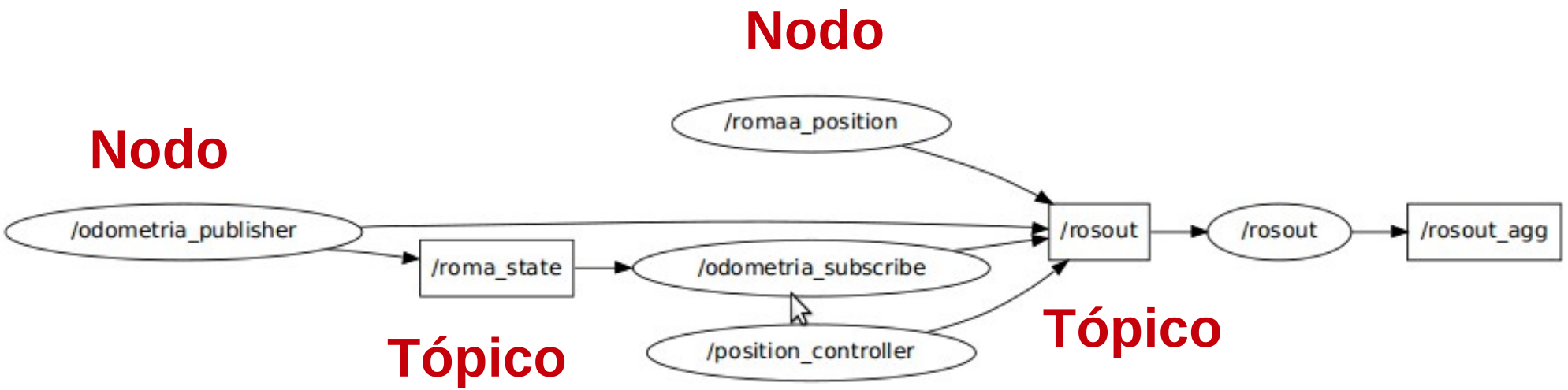
```
bool next_position( romaa_position_controler::position::Request  &req,
                    romaa_position_controler::position::Response &res )
{
    PlayerCc::PlayerClient* romaa_robot=new PlayerCc::PlayerClient("localhost", 6665 );
    PlayerCc::Position2dProxy* romaa_pos2d=new PlayerCc::Position2dProxy( romaa_robot, 0 );
    romaa_robot->SetDataMode(PLAYER_DATAMODE_PULL);
    romaa_robot->SetReplaceRule(true,PLAYER_MSGTYPE_DATA,-1,-1);
    position_controller PosControl;       //obtengo los datos del archivo de parametros
    bool CONDITION=1;
      x_ref = req.x0;
      y_ref = req.y0;
    while(1)
    {     if(CONDITION)
          {     if(romaa_robot->Peek()){
                    romaa_robot->Read();
                    x_world = romaa_pos2d->GetXPos();
                    y_world = romaa_pos2d->GetYPos();
                    w_world = romaa_pos2d->GetYaw();
                    CONDITION = PosControl.calculate(x_ref, y_ref, x_world, y_world, w_world);
                    romaa_pos2d->SetSpeed(PosControl.get_x_speed(),PosControl.get_w_speed());
}}}}
int main(int argc, char **argv)
{ ros::init(argc, argv, "position_controller");
  ros::NodeHandle ciii_position_controller;
  ros::ServiceServer service = ciii_position_controller.advertiseService("position", next_position);
   ros::spin();}
```

ROS y ROMAA

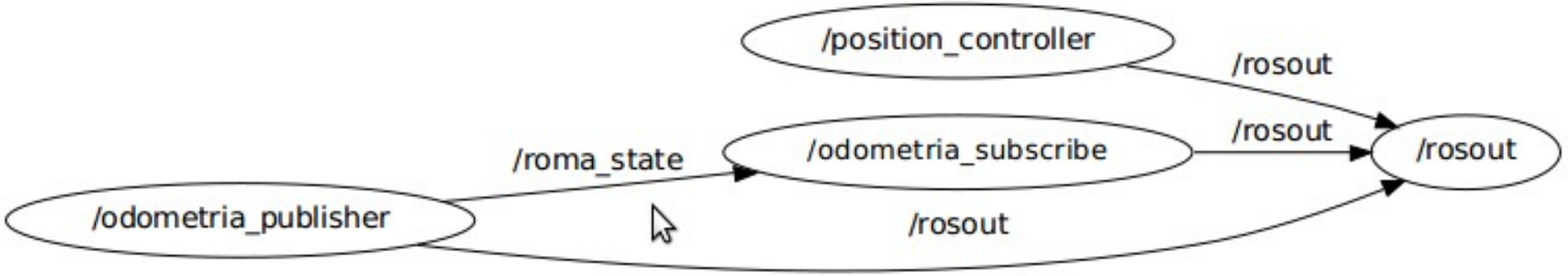
- Tópicos: Los nodos publicas datos en los tópicos y pueden subscribirse a otros



ROS - TÓPICOS


Nodo
Nodo
/romaa_position
/odometria_publisher
/rosout
/rosout
/rosout_agg
/roma_state
/odometria_subscribe
/position_controller
Tópico
Tópico

ROS - TÓPICOS


/position_controller
/rosout
/rosout
/rosout
/roma_state
/odometria_subscribe
/odometria_publisher
/rosout

ROMAA - PUBLISH

```
int main(int argc, char **argv)
{  ros::init(argc, argv, "romaa_state");

  ros::NodeHandle romaa_state;
  ros::Publisher romaa_state_pub =
romaa_state.advertise<std_msgs::Float32MultiArray>("romaa_state", 100);

ros::Rate loop_rate(5); //frecuencia en que quiero que se ejecute el nodo

      while (ros::ok())
  {
            std_msgs::Float32MultiArray state_msg;

            if(romaa_robot->Peek()){
                  romaa_robot->Read();
                  state_msg.data.push_back(romaa_pos2d->GetXPos());
                  state_msg.data.push_back(romaa_pos2d->GetYPos());
                  state_msg.data.push_back(romaa_pos2d->GetYaw());
                  romaa_state_pub.publish(state_msg);
            }
    ros::spinOnce();
    loop_rate.sleep();
  }
      return 0;}
```

ROMAA - SUBSCRIBE

```
#include "ros/ros.h"
#include "std_msgs/Float32MultiArray.h"

void romaa_state(const std_msgs::Float32MultiArray::ConstPtr& romaa_state_msg)
{
  ROS_INFO("odometria: %f %f %f", romaa_state_msg->data.at(0),romaa_state_msg-
>data.at(1),romaa_state_msg->data.at(2));
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "main");

  ros::NodeHandle roma_state_sub_node;

  ros::Subscriber sub1 = roma_state_sub_node.subscribe("romaa_state", 100, romaa_state);

  ros::spin();

  return 0;
}
```

ROS - HERRAMIENTAS

- rxgraph
- roslaunch
- rosbag y rxbag
- rxplot
- 3D Visualization: RVIZ
- Comandos

Herramientas - roslaunch

- Sirve para ejecutar nodos en forma local o remota via SSH
- roslaunch nombre_paquete archivo.launch
- Descripción en formato XML

Ejemplo

```
<launch>
<node name="position" pkg="romaa_position_controler" type="position_controller" />

 <!-- publisher -->
 <node name="cliente" pkg="romaa_position_controler" type="cliente" args="3 3" />

</launch>
```

Herramientas – rosbag y rxbag

- Grabación y visualización de datos.
- Crea un archivo .bag

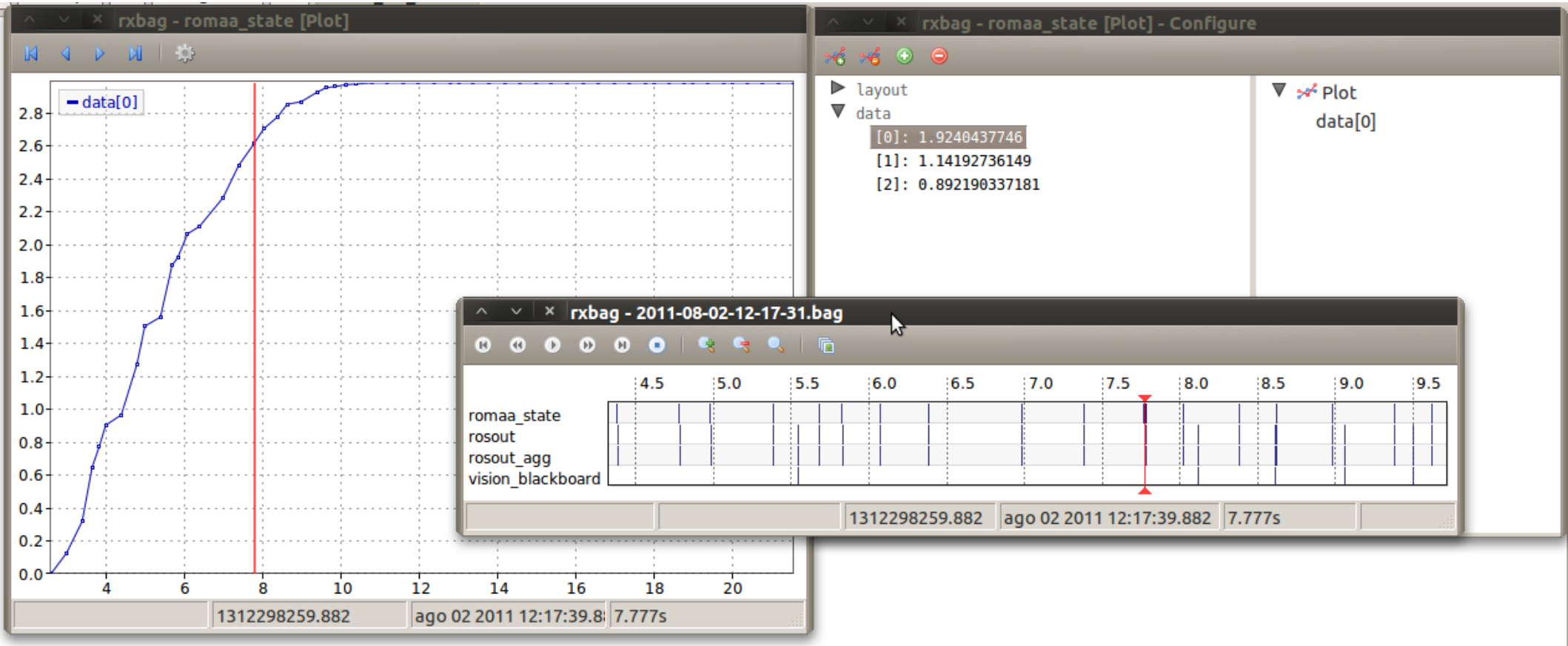
  rosbag record -a

  rosbag record -O nombre topico
- Para visualizar los datos

  rxbag archivo.bag

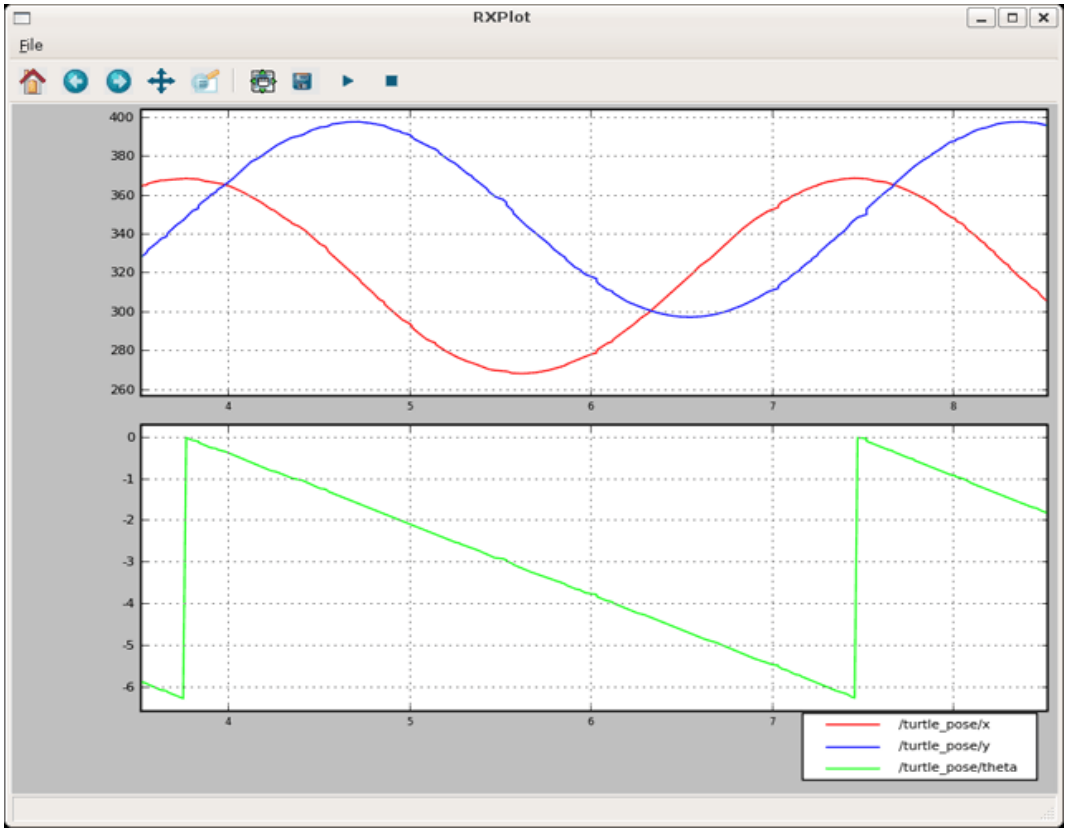
Ver: http://www.ros.org/wiki/rosbag/Cookbook

Herramientas – rosbag y rxbag

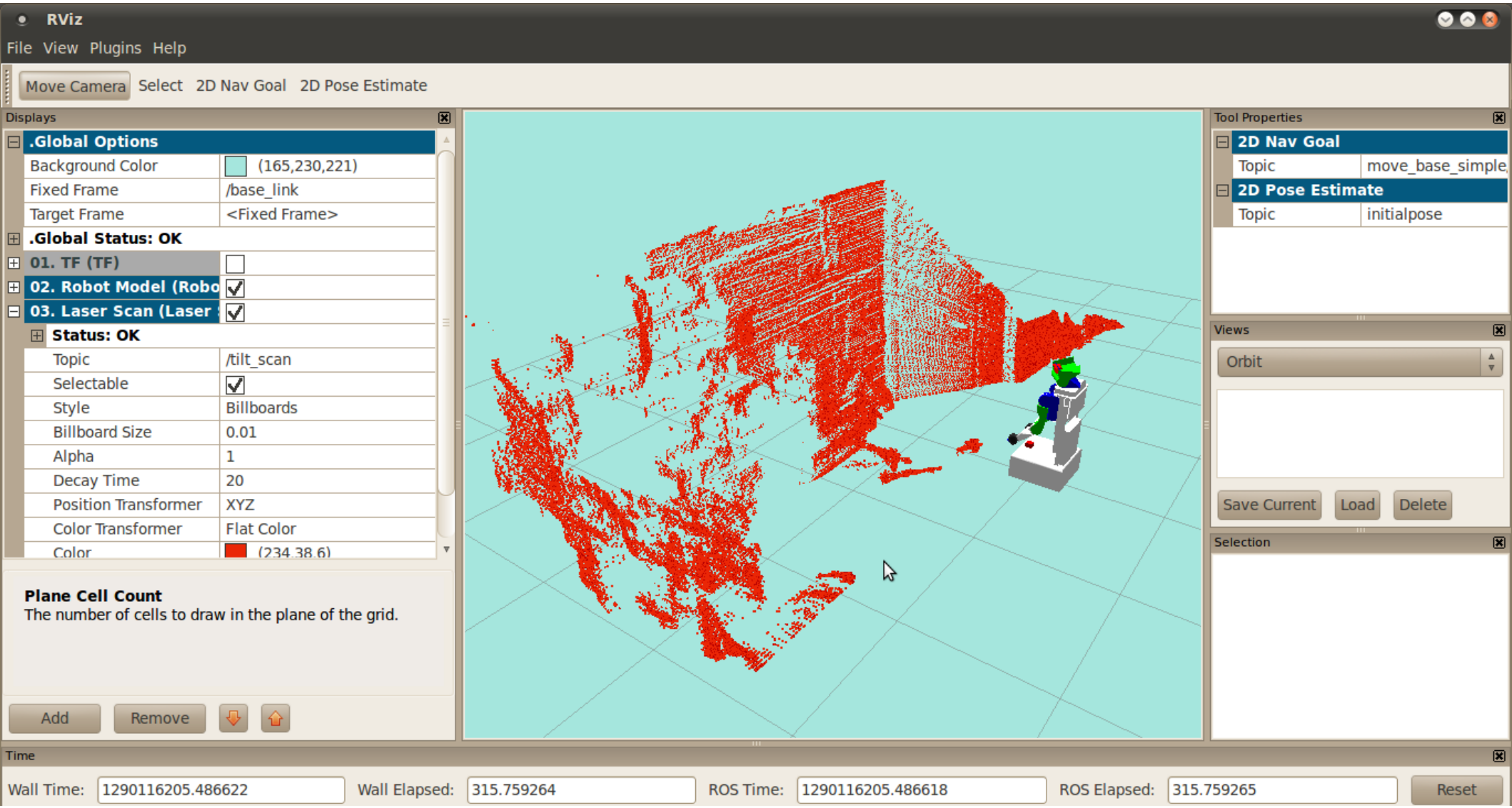


Herramientas - rxplot

- Tienen que ser escalares
- rxplot topico/campo



Herramientas - rviz



Mensajes y servicios en ROS

- Rosservice [list | info | call]
- rosmsg/rossrv [show | package]
- rostopic [list | info | hz | echo]

links

http://www.ros.org/wiki/
http://answers.ros.org/questions/
http://www.ros.org/wiki/Papers
http://www.ros.org/wiki/ROS/Tutorials/