Contenido

[Comandos de terminal de ROS 3](#_Toc116644292)

[Creación de los paquetes 3](#_Toc116644293)

[Ejecución de nodos 3](#_Toc116644294)

[Comando para paquetes 3](#_Toc116644295)

[Comandos para los topics 3](#_Toc116644296)

[Comandos para nodos 3](#_Toc116644297)

[Comando para mensajes personalizados 3](#_Toc116644298)

[Comando para servicios 3](#_Toc116644299)

[Comando para acciones 3](#_Toc116644300)

[Comandos de Rospy 4](#_Toc116644301)

[Nodos 4](#_Toc116644302)

[Topics 4](#_Toc116644303)

[Mensajes 4](#_Toc116644304)

[Bucles 4](#_Toc116644305)

[Servicios 4](#_Toc116644306)

[Servidor 4](#_Toc116644307)

[Clientes 4](#_Toc116644308)

[Acciones 5](#_Toc116644309)

[Servidor 5](#_Toc116644310)

[Cliente 5](#_Toc116644311)

[Nodos 5](#_Toc116644312)

[Publicador 5](#_Toc116644313)

[Suscriptor 5](#_Toc116644314)

[Mensajes personalizados 6](#_Toc116644315)

[Servicios 7](#_Toc116644316)

[Cliente 7](#_Toc116644317)

[Servidor 7](#_Toc116644318)

[Mensajes personalizados 8](#_Toc116644319)

[Acciones 10](#_Toc116644320)

# Comandos de terminal de ROS

## Creación de los paquetes

catkin\_create\_package "nombre del paquete" rospyà Crea un paquete con rospy

catkin\_make à Compila todos los paquetes

catkin\_make --only\_pkg\_with\_deps “nombre\_del\_paquete”à Compila un paquete

## Ejecución de nodos

roscore à Inicia el nodo maestro de ROS

rosrun "nombre del paquete" "nombre del archivo" à ejecuta un programa de Ros

## Comando para paquetes

rospack list à Devuelve la lista de paquetes ROS

## Comandos para los topics

rostopic list àPara ver la lista de topics activos

rostopic echo "topic" à Para ver el contenido de un topic

rostopic info "topic" à Para ver la información de un topic

## Comandos para nodos

rosnode list à Devuelve la lista de nodos activos

rosnode info "nombre del nodo" à Devuelve la información de un nodo

## Comando para mensajes personalizados

rosmsg list à Devuelve la lista de todos los mensajes

rosmsg show "Nombre del mensaje" à Devuelve la información de un mensaje

rosmsg list | grep “Mensaje” à Devuelve la ruta de un mensaje

## Comando para servicios

rosservice list à Devuelve la lista de servicios activos

rosservice info "servicio" à Devuelve la información de un servicio

rossrv show "paquete/mensaje" à Devuelve la información de los argumentos de un servicio

## Comando para acciones

rostopic list à Devuelve la lista de topics y las acciones

\* "accion"/cancel à Cancela la acción

\* "accion"/feedback à Muestra el feedback

\* "accion"goal à Muestra el objetivo

\* "accion"/result à Muestra el resultado

\* "accion"/status à Muestra el estado de la acción

rostopic pub "nombre\_accion/accion" à Ejecuta la acción

rostopic info "acción" à Devuelve la información de la acción

rosmsg list | grep "acción" à Muestra la lista de mensajes de las acciones

\* "accion"/MensajeAction à

\* "accion"/MensajeActionFeedback à

\* "accion"/MensajeActionGoal à

\* "accion"/MensajeActionResult à

\* "accion"/MensajeFeedback à

\* "accion"/MensajeGoal à

\* "accion"/MensajeResult à

rosmsg show "accion"/Mensaje\_\_\_ “ à Para devolver información del mensaje

# Comandos de Rospy

## Nodos

rospy.init\_node(" ") à Creación de un nodo

## Topics

“nombre” = Publisher("nombre", Tipo de mensaje, queue\_size=10) à Creación de un publicador

“nombre”.publish(mensaje) àPublica el mensaje en el topic

rospy.Subscriber("nombre", Tipo de mensaje, callback) à Creación de un suscriptor

## Mensajes

Tipos de mensajes básicos : Int32,Int64, String …

“nombre” = Int32()à Creación de un mensaje

“nombre”.”variable del mensaje” = “valor” à Asignación de un valor al mensaje

## Bucles

rospy.spin()àBucle infinito

“nombre”=rospy.Rate(“frecuencia”)à Crea una frecuencia para un bucle

rospy.is\_shutdown()à Se usa en un While not para finalizar el bucle con **Control+C**

rate.sleep()à Pausa para tener la frecuencia del bucle

## Servicios

### Servidor

“nombre”= rospy.Service(Nombre, Tipo de mensaje, callback )

### Clientes

rospy.wait\_for\_services("nombre") à Espera a que el servidor este activo

“nombre\_servicio”= rospy.ServiceProxy("/servicio", Tipo de mensaje) à Creación de un cliente para un servicio

“nombre” = “nombre\_servicio”(“petición”) à Envía la petición al servicio

## Acciones

### Servidor

### Cliente

# Nodos

## Publicador

Ejemplo:

#!/usr/bin/env python

import rospy

from std\_msgs.msg import Int32

rospy.init\_node("Primer nodo") #Para crear un nodo

pub = rospy.Publisher("laser", Int32, queue\_size=10) #Crea el topic

a = Int32() #Comfiguramos a como un mensaje de tipo Int32

contador = 0 #Creamos una variable contador

rate = rospy.Rate(10) #Para crear un rate de 10Hz

while not rospy.is\_shutdown(): #Mientras no se cierre el nodo

    contador += 1

    a.data = contador

    pub.publish(a) #Publicamos el mensaje

    rate.sleep() #Esperamos para que sea la frecuencia de 10Hz

## Suscriptor

Ejemplo :

#!/usr/bin/env python

import rospy

from std\_msgs.msg import Int32

def callback(msg):

    """

    Funcion que se ejecuta cuando se recibe un mensaje

    """

    print ( msg )

rospy.init\_node('lectura')#Creamos el nodo

rospy.Subscriber('laser', Int32, callback) #Nos suscribimos al topic laser

rospy.spin() #Mantener el nodo activo

## Mensajes personalizados

1. Crear una carpeta msg en el paquete
2. Generamos un archivo “nombre”.msg y añadimos las variables del mensaje
3. Modificar el archivo CMakeLists.txt
4. En la función **find\_package ()** dentro de ella escribimos "**message\_generation**" y también añadimos los **paquetes** que use el mensaje creado (En líneas diferentes)
5. En la función **add\_message\_files ()** la descomentamos y añadimos el **nombre del archivo .msg** que creamos (En líneas diferentes)
6. En la función **generate\_messages ()** la descomentamos y **añadimos el nombre de los paquetes** que usamos en el mensaje creado
7. En la función **catkin\_package ()** la descomentamos y en la **línea de CATKIN\_DEPENDS añadimos rospy y message\_rutine**
8. Comprobamos si la función **include\_directories()** **esta descomentada y si ${catkin\_INCLUDE\_DIRS} esta dentro** de ella, pero dejamos comentado incluye

4.**Modificar package.xml**

1) En el final modificamos el archivo para que siga el siguiente esquema

**<build\_depend> rospy </build\_depend>**

**<build\_depend> message\_generation </build\_depend>**

**<build\_export\_depend> rospy </build\_export\_depend>**

**<build\_export\_depend> message\_rutine </build\_export\_depend>**

**<exec\_depend> rospy </exec\_depend>**

**<exec\_depend> message\_rutine </exec\_depend>**

5. **Compilar el paquete** ejecutando **catkin\_make --only\_pkg\_with\_deps “nombre\_del\_paquete”** dentro de la carpeta catkin\_ws

# Servicios

## Cliente

Ejemplo:

#!/usr/bin/env python

import rospy

from paquete.srv import Mensaje, MensajeRequest

rospy.init\_node('cliente\_servicio') #Crea el topic

rospy.wait\_for\_service('/nombre\_servicio')#Espera asta que el servicio se este ejecutando para ofrecer el servicio

#Se crea la conexion con el servicio

servicio = rospy.ServiceProxy("/servicio", "nombre del servicio")

#Se crea el objeto de "Tipo de mensaje" para enviarlo al servicio

peticion = MensajeRequest()

peticion.dato = "dato" #Asignacion de los datos a la peticion

#Se envia la peticion al servicio

resultado = servicio(peticion)

#Se imprime el resultado

print(resultado)

## 

## Servidor

Ejemplo:

#!/usr/bin/env python

import rospy

from std\_srvs.srv import Empty, EmptyResponse #Importa el tipo de mensaje que necesitamos

def mi\_callback(request):

    """

    Funcion que sera ejecutada cuando se llame al servicio mi\_servicio

    """

    print("Peticion recibida")

    return (EmptyResponse) #Retorno un mensaje vacio

rospy.init\_node('servidor') #Inicializa el nodo

mi\_servicio = rospy.Service("mi\_servicio", Empty, mi\_callback) #Crea el servicio. Argumentos: Nombre del servicio, Tipo de mensaje, Funcion que se ejecuta cuando se recibe una peticion

rospy.spin() #Mantiene el nodo activo

## Mensajes personalizados

1. Crear una carpeta srv en el paquete del servidor del servicio
2. Creamos y editamos el fichero "Nombre".srv con el siguiente esquema

#Petición

int32 dato

---

#Respuesta

bool éxito

1. Modificamos el CMLists.txt para que se compile el servicio
2. En la función **find package()**, la descomentamos y **añadimos message\_generation** y añadimos las **dependencias** del archivo
3. En la función **add\_service\_files** **añadimos el nombre del archivo** de mensaje
4. En la función **generate\_messages**, la descomentamos y **añadimos las dependencias del archivo**
5. En la función **catkin\_package** **añadimos rospy y message\_runtime en la línea de CATKIN\_DEPENDS**

1. **Modificar package.xml**

1.En el final modificamos el archivo para que siga el siguiente esquema.

**<build\_depend> rospy </build\_depend>**

**<build\_depend> message\_generation </build\_depend>**

**<build\_export\_depend> rospy </build\_export\_depend>**

**<build\_export\_depend> message\_rutine </build\_export\_depend>**

**<exec\_depend> rospy </exec\_depend>**

**<exec\_depend> message\_rutine </exec\_depend>**

1. **Compilamos** el paquete usando **catkin\_make --only\_pkg\_with\_deps “nombre\_del\_paquete”** dentro de la carpeta catkin\_ws

# Acciones

## Cliente

## Servidor

## Mensajes personalizados