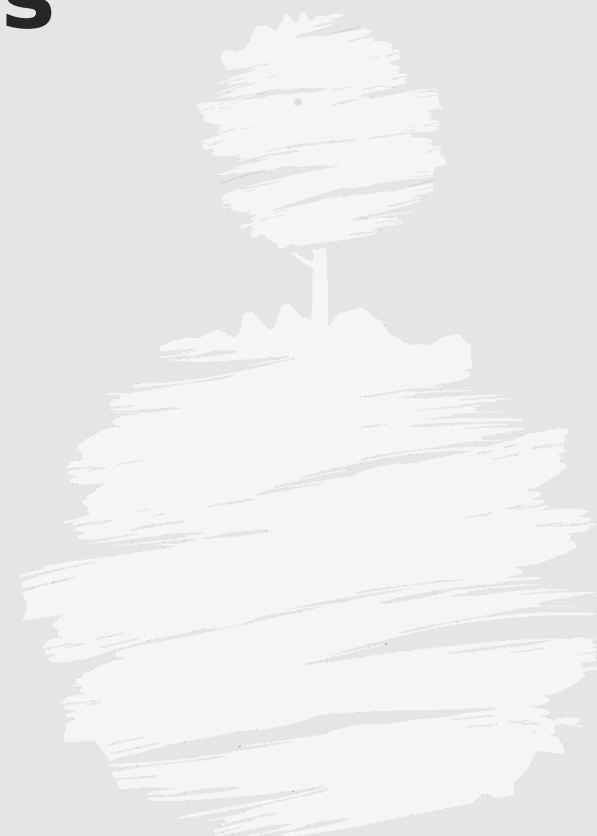


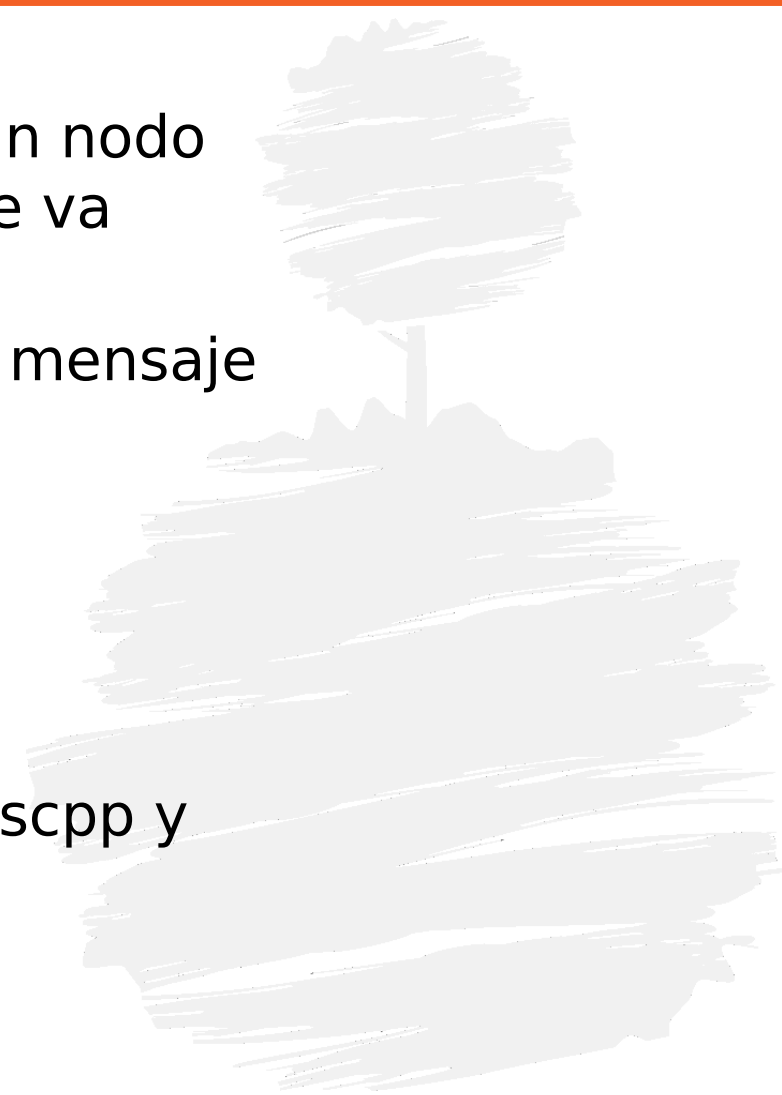
ROS y C++. Topics - Publicadores

Plataformas de Software en Robótica



Crear un publicador básico

- En esta presentación vamos a mostrar como crear un nodo publicador básico que publica un valor entero que se va incrementando (contador)
- Para poder publicar un valor entero necesitamos un mensaje que lo soporte
 - ▷ Mensajes estándar (básicos)
 - ➞ `std_msgs/Int32`
 - ▷ Para poder consultar el formato de un mensaje
 - ➞ `rosmmsg show std_msgs/Int32`
- Debemos crear un paquete con las dependencias `roscpp` y `std_msgs`
- Creamos el fichero `.cpp` en la carpeta `src`



Estructura del programa

```
#include <ros/ros.h>
#include <std_msgs/Int32.h>

int main(int argc, char ** argv){
    ros::init(argc,argv, "simple_node");
    ros::NodeHandle nh;
    ros::Publisher pub=nh.advertise<std_msgs::Int32>("contador", 1);/*creamos
    un publicador empleando la función advertise. Esta función es una plantilla
    (template) por lo que tenemos que decirle el tipo de mensaje que va a
    utilizar. Los argumentos son el nombre del topic y el tamaño de la cola*/

    std_msgs::Int32 cont;//El tipo de mensaje que queremos publicar
    cont.data=0; //La estructura del mensaje la tenemos que consultar con rosmg show
    ros::Rate loop_rate(1);

    while(ros::ok()){
        pub.publish(cont);//Publicamos en el topic
        cont.data++;
        loop_rate.sleep();
    }
    return 0;
}
```

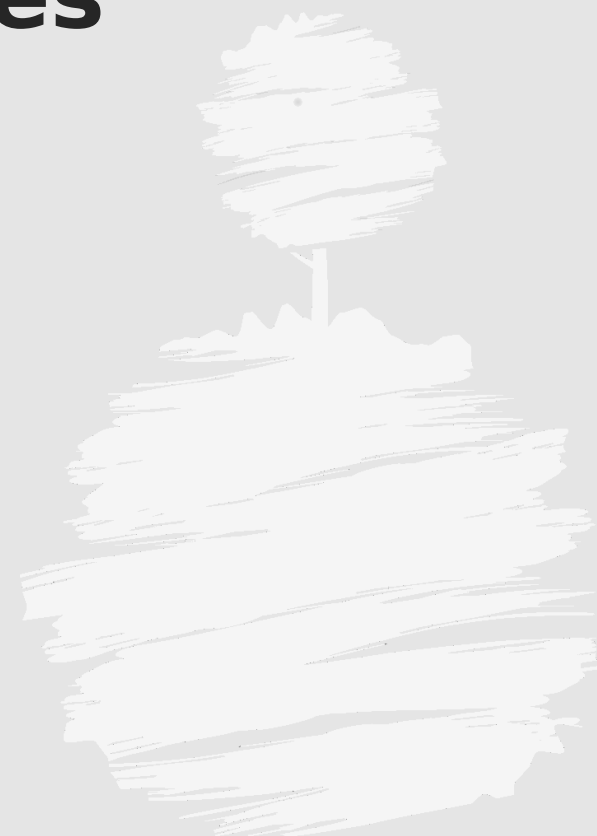
Ejecución del publicador

- Modificamos el CmakeLists.txt (sección `##build##`)
 - ▷ `add_executable`
 - ▷ `add_dependencies`
 - ▷ `target_link_libraries`
- Creamos la carpeta launch y un fichero de inicio (opcional)
 - ▷ El atributo `type` debe de coincidir con el nombre del ejecutable
- Compilar con:
`catkin_make --only-pkg-with-deps "nombre_paquete"`
- Lanzar el nodo con el comando `roslaunch` (opcional)
o
- Ejecutar `roscore` y luego lanzar el nodo con:
`roslaunch "nombre_paquete" "nombre_ejecutable"`
- Consultar el topic (desde otra terminal)

➡ `rostopic echo contador`

ROS y C++. Topics - Subscriptores

Plataformas de Software en Robótica



Crear un subscritor básico

- En esta presentación vamos a mostrar como crear un nodo subscritor básico que lee del topic contador creado anteriormente
- Consultamos los topics disponibles y el tipo de mensaje que publican, en caso de no conocerlo
- Creamos un proyecto (paquete) nuevo
 - ▷ Necesitará tener las dependencias de C++ (roscpp) y la de std_msgs
- Creamos un fichero .cpp en la carpeta src del paquete creado

Estructura del programa

```
✓ #include<ros/ros.h>
  #include<std_msgs/Int32.h>

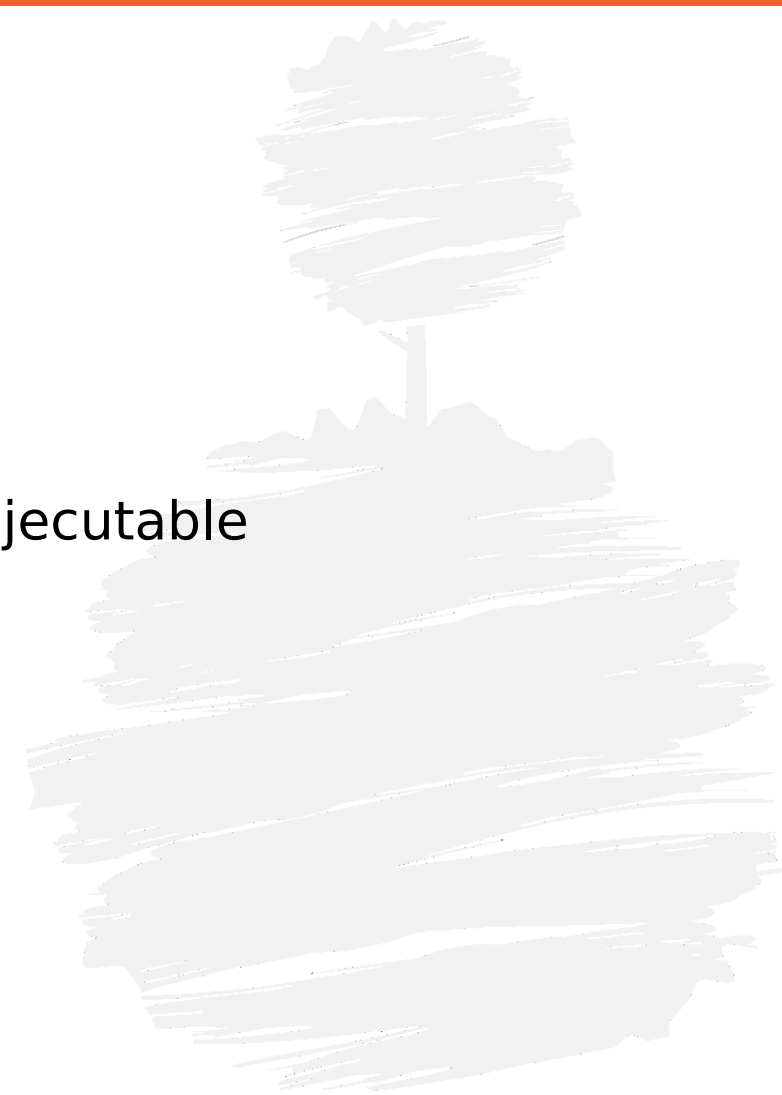
✓ void contadorCallback(const std_msgs::Int32ConstPtr & msg){
    ROS_INFO("CallBack contador: %i",msg->data);
}

✓ int main (int argc, char ** argv){
    ros::init(argc, argv, "subscriptor");
    ros::NodeHandle nh;
    ros::Subscriber sub=nh.subscribe<std_msgs::Int32>("contador",10,contadorCallback);

    ros::spin();
    return 0;
}
```

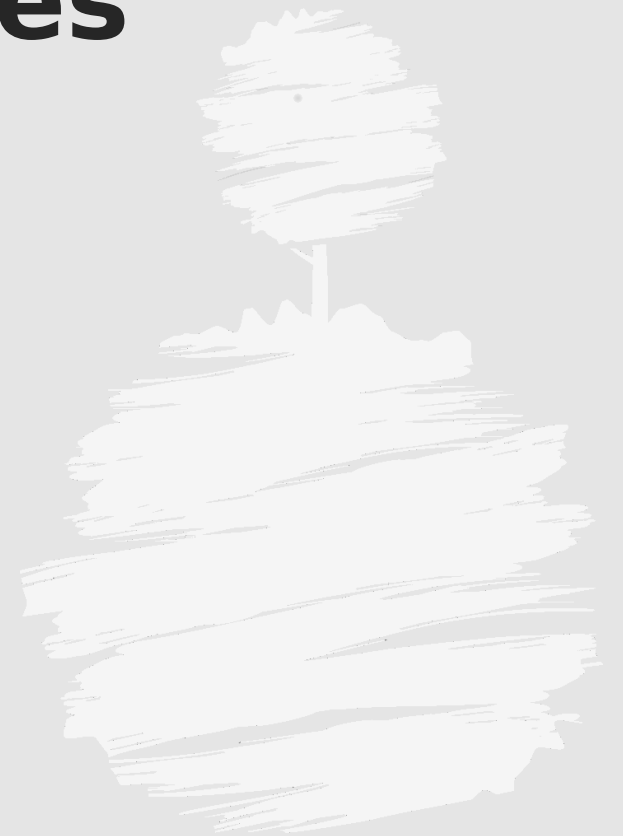
Ejecución del subscriptor

- Modificamos el CmakeLists.txt (sección ##build##)
 - ▷ add_executable
 - ▷ add_dependencies
 - ▷ target_link_libraries
- Creamos la carpeta launch y un fichero de inicio
 - ▷ El atributo type debe de coincidir con el nombre del ejecutable
- Compilar con catkin_make
- Lanzar el publicador para crear el topic
- Lanzar el subscriptor



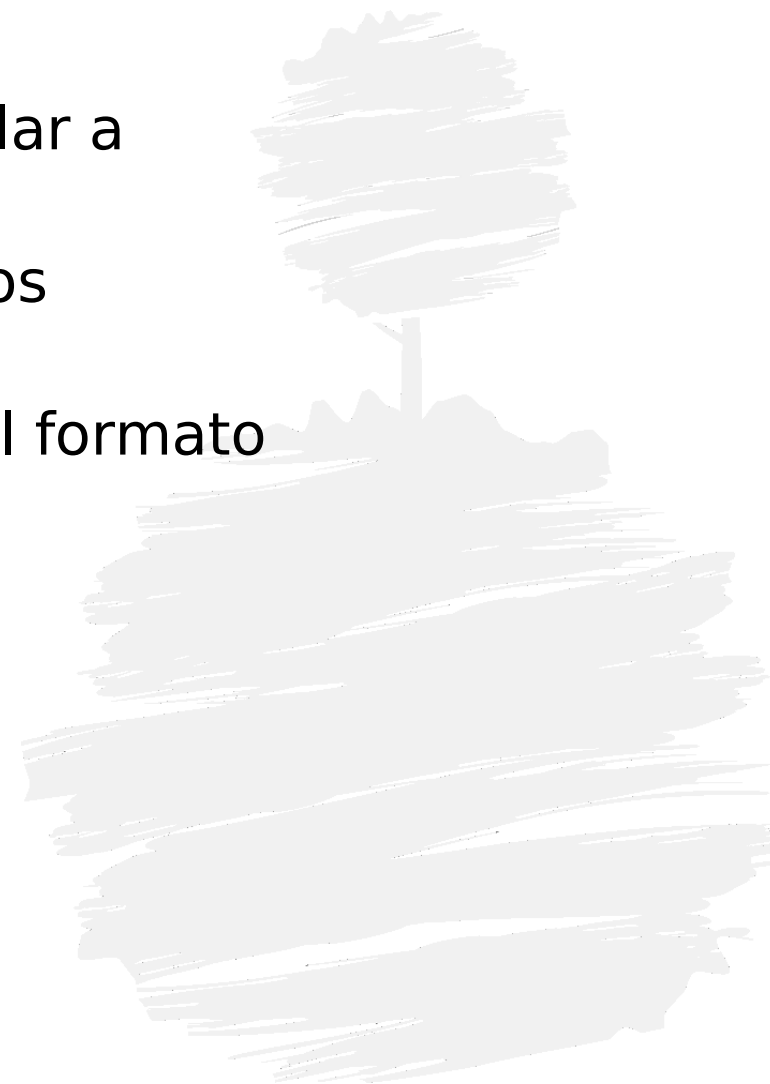
ROS y C++. Topics - Mensajes

Plataformas de Software en Robótica



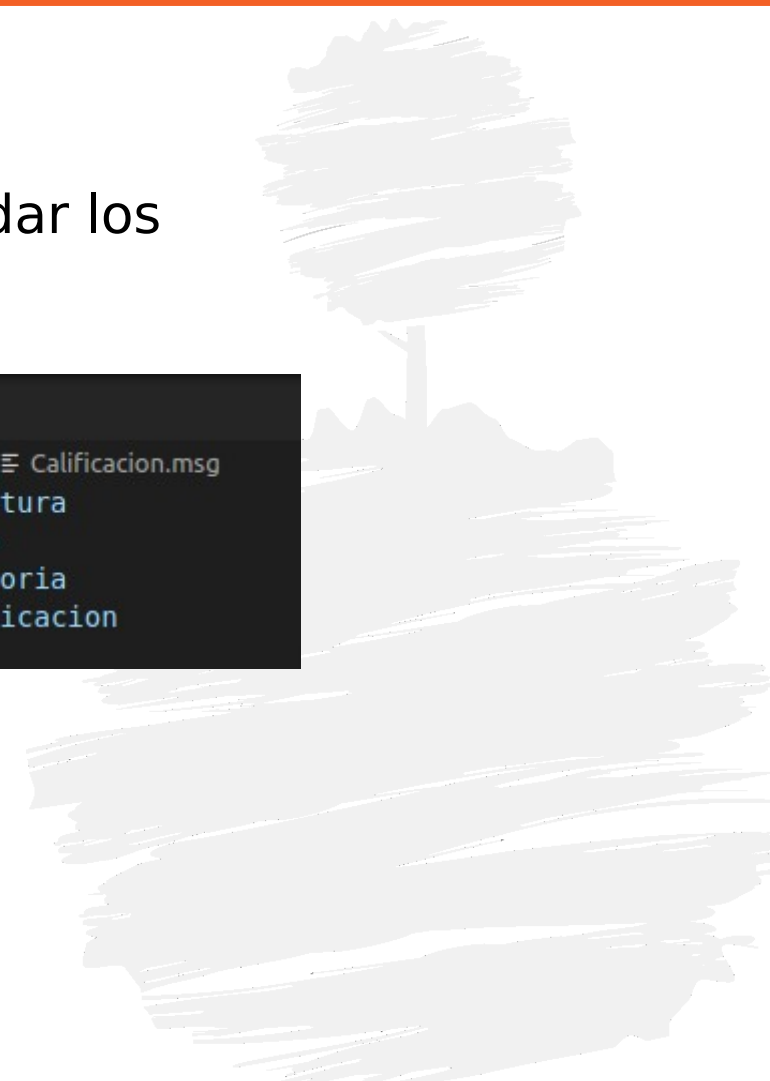
Crear un mensaje personalizado

- La forma de crear un mensaje personalizado es similar a Python, ya que no se emplea código C++
- Creamos un paquete en el que guardaremos nuestros mensajes personalizados
- Necesitamos crear una carpeta msg que contenga el formato del mensaje
- Creamos el fichero con el mensaje personalizado
 - ▷ <nombre_mensaje>.msg



Crear un mensaje personalizado

- Ejemplo de un mensaje
 - ▷ Suponiendo un mensaje en el que tenemos que guardar los siguientes datos:
 - ⇒ Nombre de una asignatura
 - ⇒ Nombre de un alumno
 - ⇒ Calificación
 - ⇒ Convocatoria



```
Calificacion.msg x
src > b1_ej4_pkg > msg > Calificacion.msg
1  string asignatura
2  string alumno
3  int8 convocatoria
4  float32 calificacion
5
```

Crear un mensaje personalizado

- Acciones necesarias para compilar el mensaje y tenerlo accesible
- Modificación del CmakeLists.txt
 - ▷ Añadir el paquete message_generation a la sección find_package

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  #std_msgs ##No es necesario, ya que viene por defecto
  message_generation
)
```

- ▷ Descomentar la sección “add_message_files”

```
add_message_files(
  FILES
  nombre_mensaje.msg
)
```

Crear un mensaje personalizado

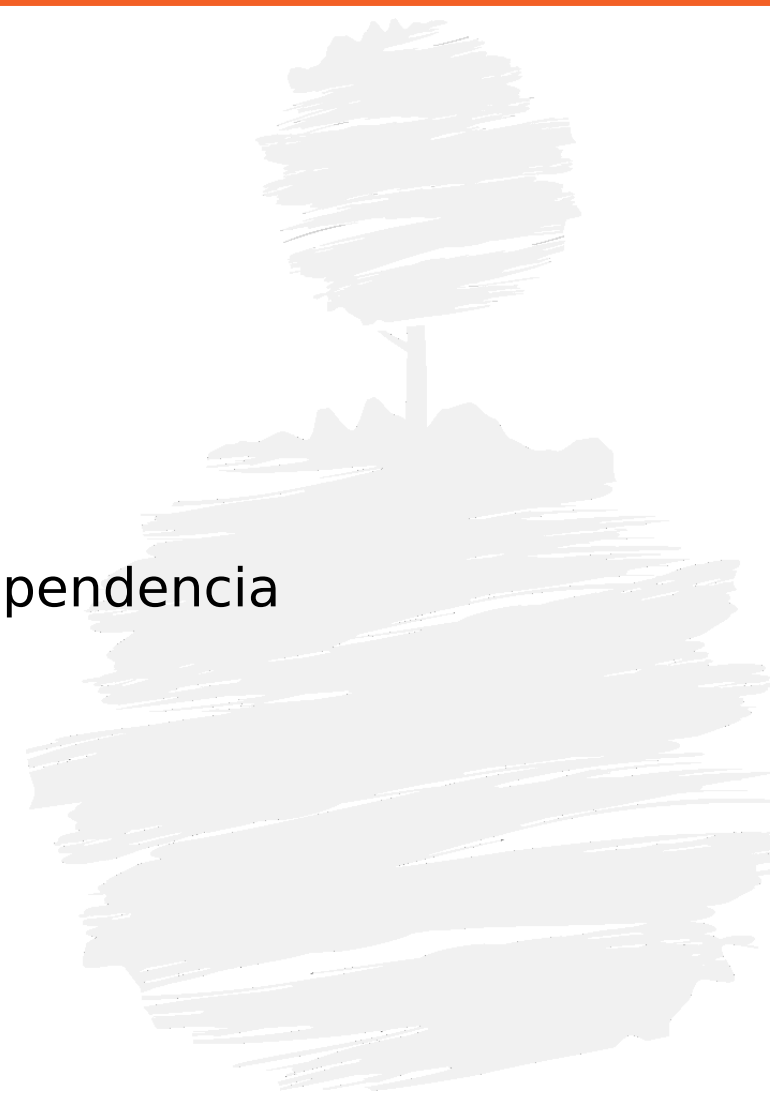
- Modificación del CmakeLists.txt

- ▶ Descomentar la sección generate_messages

```
generate_messages(  
  DEPENDENCIES  
    std_msgs  
)
```

- ▶ Modificar la sección catkin_package añadiendo la dependencia message_runtime

```
catkin_package(  
  CATKIN_DEPENDS roscpp message_runtime  
)
```



Crear un mensaje personalizado

- Modificación del package.xml
 - ▷ Añadir las siguientes líneas
 - ➞ `<build_depend>message_generation</build_depend>`
 - ➞ `<exec_depend>message_runtime</exec_depend>`
- Una vez realizadas las modificaciones, es necesario compilar el paquete que contiene los mensajes
- Para emplearlo en otro proyecto podemos simplemente agregar el paquete con los mensajes como una dependencia al nuevo paquete
 - ▷ `catkin_create_pkg <nuevo_pkg> roscpp std_msgs <my_own_msgs_pkg>`

Consultar un mensaje personalizado

- Podemos buscar nuestro mensaje personalizado, una vez compilado, con:
rosmmsg list
- Podemos buscar dentro de la lista de mensajes empleando utilidades de línea de comandos:
rosmmsg list | grep nombre_mensaje
- Para consultar los campos del mensaje:
rosmmsg show nombre_mensaje