ENTREGABLE 21:

DESARROLLO DEL SISTEMA DE CONTROL Y COMUNICACIÓN DE LA SOLUCIÓN ROBÓTICA MÓVIL

(Octubre 2022)

ACTIVIDAD 3: DESARROLLO DE CAPACIDADES FORMATIVAS Y ENTORNOS DE EXPERIMENTACIÓN PARA LA INNOVACIÓN DIGITAL EMPRESARIAL

Contenido

1. OBJETIVO TAREA

En este informe se detalla el sistema de comunicación y control seleccionado y puesto en marcha para la solución robótica móvil. Se detallará cómo y porqué se ha implementado ROS, la estructura se comunicación entre dispositivos, la conexión con el simulador Gazebo, el visualizador de datos Rviz y posibilidades de desarrollo que ofrece estos sistemas.

2. ROS (ROBOT OPERATING SYSTEM)

ROS (Robot Operating System) es un kit de desarrollo de software de código abierto para aplicaciones de robótica. Ofrece una plataforma de software estándar que permite la creación de prototipos de forma más rápida a la tradicional. Esto y el cada vez mayor desarrollo y adopción de estos sistemas en la robótica tanto industrial como agrícola y, en especial, en los robots móviles autónomos o AMR (Autonomous Mobile Robots) ha conllevado la selección de ROS como plataforma de software sobre la que desarrollar el robot en este proyecto. Además, al ser open-source, existe una gran comunidad detrás de ROS que ayuda al desarrollo de aplicaciones de robótica siendo la plataforma por excelencia empleada en investigación. A continuación, se van a explicar algunos de los conceptos más importantes de un entorno ROS. Como conceptos fundamentales de ROS podemos describir los nodos, el maestro, el Servidor de Parámetros, los mensajes, los servicios y los topics.

Nodos: Los nodos son programas o procesos en los que se realizan cálculos o se ejecuta una determinada tarea. ROS es modular y un sistema de control para un robot puede contener muchos nodos. Para el robot que nos ocupa, por ejemplo, un nodo controla el LiDAR, otro nodo controla los motores de las ruedas, otro nodo la geolocalización, otro la planificación de la trayectoria, un nodo proporciona una interfaz gráfica, etc. En nuestro caso, para crear los nodos de ROS se va a usar la biblioteca rospy.

Mensajes: Los nodos se comunican entre sí mediante mensajes. Un mensaje es simplemente una estructura de datos, que comprende campos tipificados. Se admiten tipos de datos estándar como entero, coma flotante, booleano, etc., así como matrices.

Topics: Los mensajes se dirigen a través de un sistema de publicación/suscripción. Un nodo envía un mensaje publicándolo en un tema determinado. El tema es un nombre que se utiliza para identificar el contenido del mensaje. Un nodo que esté interesado en un determinado tipo de datos se suscribirá al tema correspondiente. Puede haber varios publicadores y suscriptores simultáneos para un mismo tema, y un mismo nodo puede publicar y/o suscribirse a varios temas. En general, los publicadores y suscriptores no conocen la existencia de los demás.

Servicios: El modelo de publicación/suscripción es un paradigma de comunicación muy flexible, pero su transporte unidireccional de muchos a muchos no es apropiado para las interacciones de solicitud/respuesta. La solicitud/respuesta se realiza a través de servicios, que se definen mediante un par de estructuras de mensajes, una para la solicitud y otra para la respuesta. Un nodo proveedor ofrece un servicio bajo un nombre y un cliente utiliza el servicio enviando el mensaje de solicitud y esperando la respuesta.

Master: El ROS Master permite que los nodos puedan encontrarse entre sí, intercambiar mensajes o invocar servicios.

Servidor de parámetros: El Servidor de Parámetros permite almacenar datos por clave en una ubicación central. Forma parte del Master.

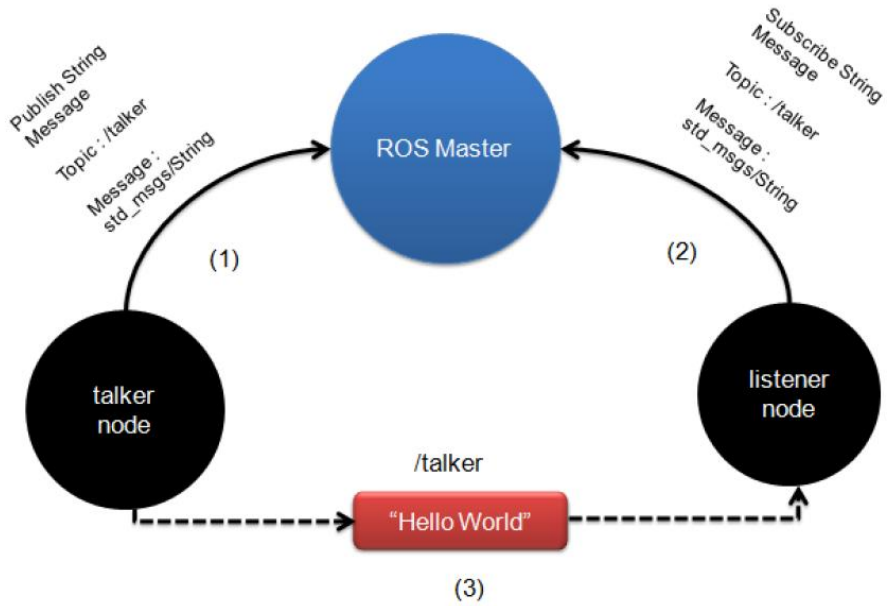

Figura 1. Ejemplo de comunicación en ROS entre 2 nodos a través de los topics

En la Figura 1 se muestran dos nodos, llamados publicador y suscriptor. El nodo publicador publica un mensaje tipo string con el contenido de 'Hola Mundo' en un topic llamado /talker, y el nodo suscriptor está suscrito a este topic. La secuencia es la siguiente:

1. Antes de ejecutar cualquier nodo en ROS, se inicia el ROS Master. Después de que se haya iniciado, esperará a los nodos. Cuando el nodo publicador comienza a funcionar, primero se conectará con el ROS master e intercambiará los detalles del topic. Esto incluye el nombre del topic, el tipo de mensaje y la URI del nodo publicador. El URI del maestro es un valor global, y todos los nodos pueden conectarse a él. El maestro mantiene tablas de los publicadores conectados a él. Cada vez que los detalles de un publicador cambian, la tabla se actualiza automáticamente.

2. Cuando se inicia el nodo suscriptor, éste se conectará con el master e intercambiará los detalles del nodo, como el topic al que se va a suscribir, su tipo de mensaje y el URI del nodo. El master también mantiene una tabla de suscriptores, similar a la del publicador.
3. Siempre que haya un suscriptor y un publicador para el mismo topic, el nodo maestro intercambiará el URI del publicador con el suscriptor. Esto ayudará a que ambos nodos puedan conectarse entre sí e intercambiar datos. Después de que se hayan conectado entre sí, el master no realiza ninguna función y los datos se intercambian sin pasar por el master.

En la solución robótica propuesta se ha empleado la distribución de Linux Ubuntu 18.04.6 LTS y la versión de ROS Melodic.

2.1 Estructura

El sistema de archivos en ROS se puede clasificar en metapaquetes, paquetes, manifiesto de paquetes, mensajes, servicios, códigos y archivos. A continuación, se presenta una breve descripción de cada componente:

- Metapaquetes: Los metapaquetes agrupan una lista de paquetes para una aplicación. Por ejemplo, en ROS, hay un metapaquete llamado navegación para robots móviles. Puede mantener la información sobre paquetes relacionados y ayuda a instalar esos paquetes durante su propia instalación.
- Paquetes: El software en ROS se organiza principalmente como paquetes ROS. Podemos decir que los paquetes ROS son la unidad de construcción atómica de ROS. Un paquete puede consistir en nodos/procesos ROS, conjuntos de datos y archivos de configuración, organizados en un único módulo.
- Manifiesto del paquete: Dentro de cada paquete habrá un archivo de manifiesto llamado package.xml. Este archivo consta de información como el nombre, la versión, el autor, la licencia y las dependencias necesarias del paquete. El archivo package.xml de un metapaquete consiste en los nombres de los paquetes relacionados.
- Mensajes (msg): ROS se comunica mediante el envío de mensajes ROS. El tipo de datos de los mensajes se puede definir dentro de un archivo con la extensión .msg. Estos archivos se llaman archivos de mensajes. Los archivos de mensajes se guardan en el directorio paquete/msg/archivos_de_mensajes.msg.
- Servicio (srv): Uno de los conceptos de nivel de gráfico de computación son los servicios. Se guardan en el directorio paquete/srv/archivos_servicio.srv.

El sistema de archivos usado en ROS puede verse en la Figura 2.

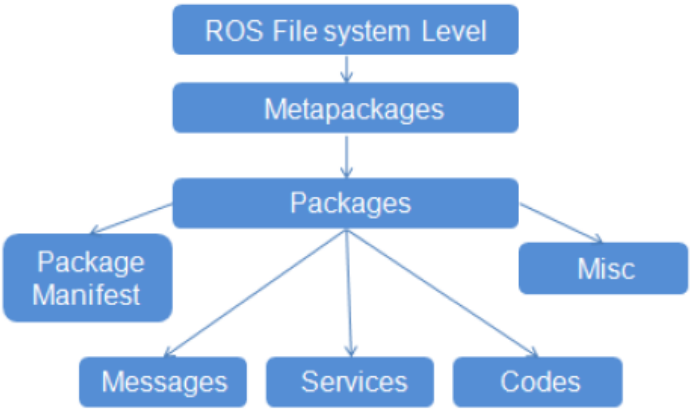

Figura 2. Sistema de archivos en ROS

El directorio principal o espacio de trabajo de ROS es *catkin_ws*. Dentro de este directorio se encuentran 3 carpetas: *build*, *devel* y *src*. La carpeta *build* es la ubicación por defecto donde *cmake* y *make* son llamados para configurar y construir los paquetes. La carpeta *devel* es donde los ejecutables y bibliotecas se sitúan antes de instalar los paquetes. En la carpeta *src* es donde se crea los paquetes y código del robot. En ella se creará un paquete, por ejemplo, '*robot_package*' que contendrá a su vez distintas carpetas (*config*, *launch*, *scripts*, etc.). En la carpeta scripts irán ubicados los archivos Python de programación de las funciones del robot. Dichos scripts serán lanzados mediante la ejecución de los archivos *.launch* presentes dentro de la carpeta *launch*.

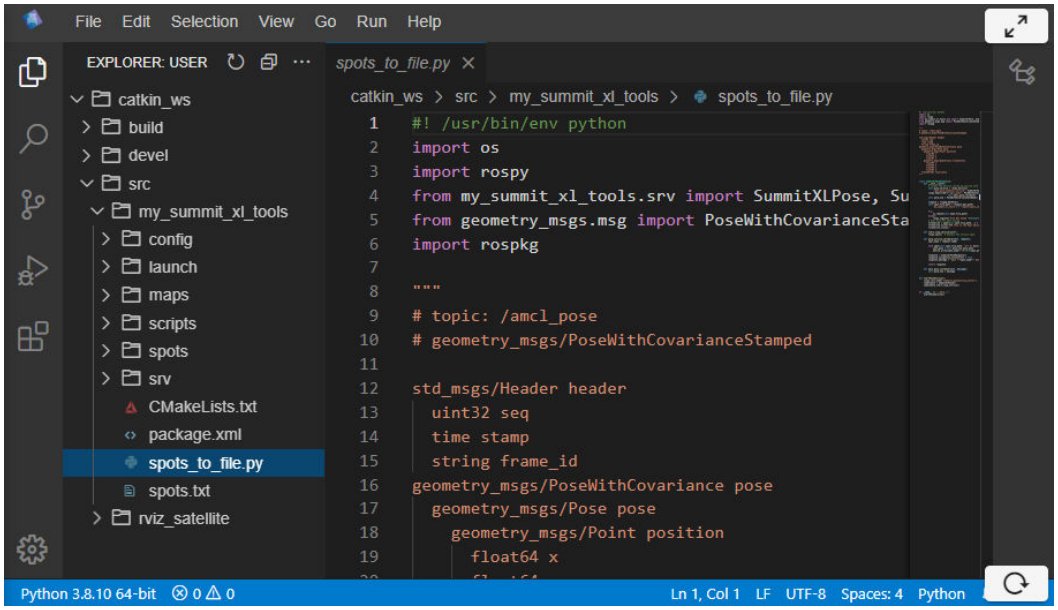

Figura 3. Aplicación en ROS empleando VS Code.

En la Figura 2 se observa a la izquierda el espacio de trabajo junto con el árbol de carpetas y subcarpetas comentado anteriormente. A la derecha se puede ver un

programa abierto realizado en Python. Para la ejecución de las diversas funciones del robot se utilizarán los topics mediante suscripción o publicación en el script.

/cmd_vel: publicando este mensaje de forma directa en el robot, este puede moverse tanto linear como angularmente. Por ejemplo:

```
rostopic pub -r1 /summit_xl_control/cmd_vel geometry_msgs/Twist
"linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.0"
```

Sin embargo, lo normal es emplear estos topics mediante un programa desarrollado en Python. Por ejemplo:

```
#! /usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

rospy.init_node('topic_publisher')
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
rate = rospy.Rate(2)
#count = Int32()
var = Twist()
var.linear.x = 0  # Move the robot with a linear velocity in the x
axis
var.angular.z = 0.5  # Move the with an angular velocity in the z
axis

while not rospy.is_shutdown():
    pub.publish(var)
    #var.data += 1
    rate.sleep()
```

2.2 Gazebo

Gazebo es el simulador 3D que incluye ROS y que se puede aplicar tanto a entornos interiores como exteriores, con soporte para múltiples robots. Permite una completa simulación dinámica y cinemática. Proporciona un renderizado realista de los entornos, incluyendo iluminación de alta calidad sombras y texturas. Puede modelar diferentes tipos de sensores para captar el entorno simulado, como LiDAR, cámaras, etc. Diferentes funcionalidades pueden ser ampliadas cargando los respectivos plugins.

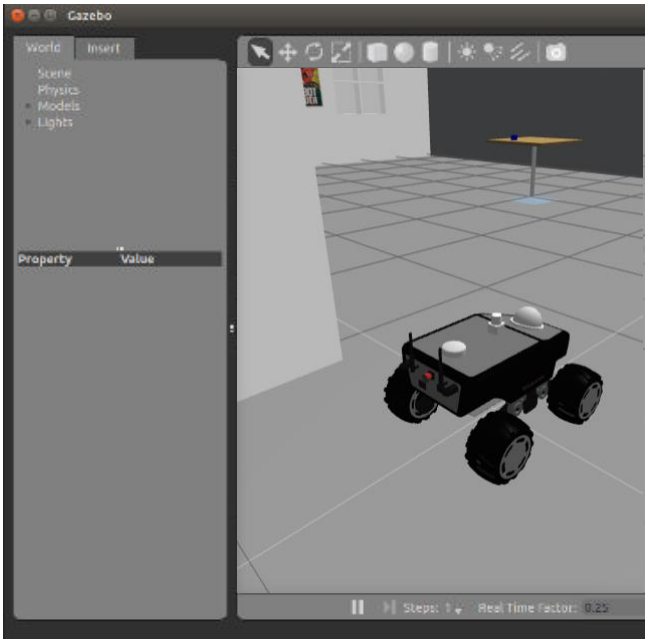

Figura 4. Simulación en Gazebo del robot

2.3 Rviz

Rviz es un visualizador de datos 2D y 3D de ROS donde es posible cargar datos de imágenes, nubes de puntos, datos de transformación 3D del robot (TF), modelos de robots, así como transformar datos, etc. en tiempo real.

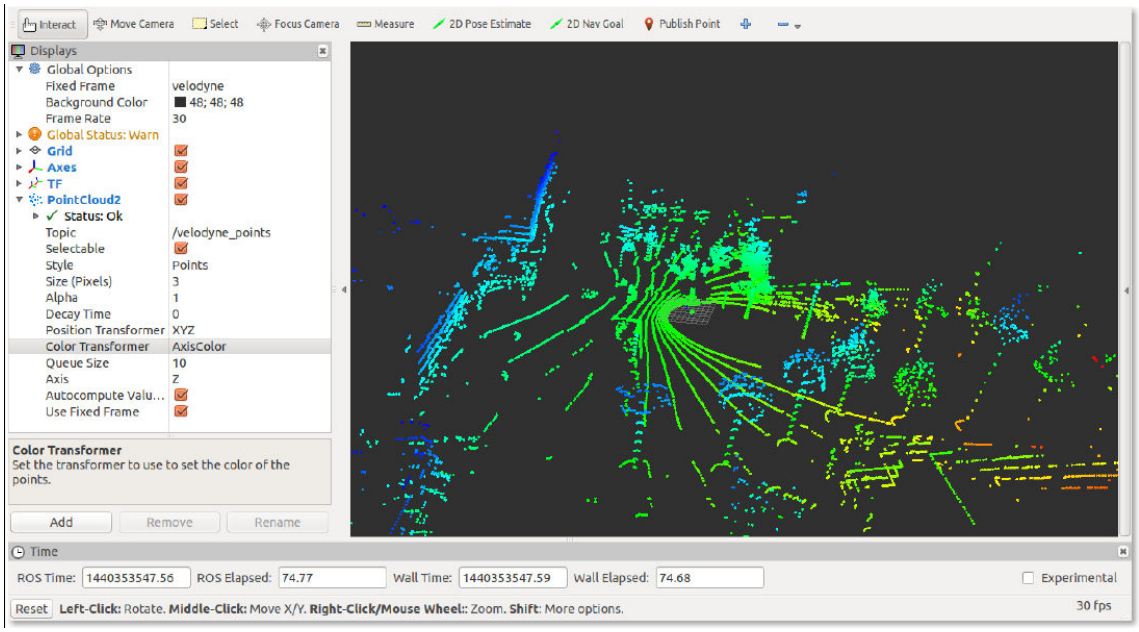

Figura 5. Visualizador de datos Rviz. Ejemplo de datos con LiDAR


Interreg
España - Portugal
Fondo Europeo de Desarrollo Regional
UNION EUROPEA
HIBA
HUB IBERIA AGROTECH
COMPETITIVIDAD


Interreg
España - Portugal
Fondo Europeo de Desarrollo Regional
UNION EUROPEA
COMPETITIVIDAD
HIBA
HUB IBERIA AGROTECH
Junta de Andalucía
Consejería de Agricultura, Ganadería, Pesca y Desarrollo Sostenible
UNIVERSIDAD D CÓRDOBA
Junta de Andalucía
Consejería de Transformación Económica, Industria, Conocimiento y Universidades
AXENCIA GALEGA DE INNOVACIÓN
CATEC
ADVANCED CENTER FOR AEROSPACE TECHNOLOGIES
FADA
ANDALUSIAN FOUNDATION FOR AEROSPACE DEVELOPMENT
INTEC
FUNDACIÓN EUROPEA PARA LA INNOVACIÓN Y APLICACIÓN DE LA TECNOLOGÍA
Finnova
GOBIERNO DE ESPAÑA
MINISTERIO DE DEFENSA
INTA
INSTITUTO NACIONAL DE TÉCNICA AEROESPACIAL
DiH DATAlife
FUNDECYT PCTEX
UNIVERSIDADE DE ÉVORA
cooperativas agro-alimentarias Andalucía
iniav
Instituto Nacional de Investigação Agrária e Veterinária, I.P.
UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Galileo Galilei
IPBeja
INSTITUTO POLITÉCNICO DE BEJA
SMART FARM COLAB
LABORATÓRIO COLABORATIVO PARA A INOVAÇÃO DIGITAL NA AGRICULTURA
INL
INTERNATIONAL IBERIAN NANOTECHNOLOGY LABORATORY
CONSULAI
www.consulai.com