Facultad de Estudios Superiores
Acatlán
Centro de Desarrollo Tecnológico
Laboratorio de Algoritmos para la Robótica
LAR

Guía de instalación y programación en ROS, para visualizar en Webots. Marzo-Julio 2023

Presentan:
Miriam Hernández, César Ricardo Zetina

Versión 1.0

Guía de instalación y programación en ROS, para visualizar en Webots.

Enero-Diciembre 2023

Miriam Hernández, César Ricardo Zetina

Objetivo

El presente documento tiene como objetivo, guiar al alumno a entender y configurar de forma correcta su sistema para el uso y manejo de ROS. ROS es el estándar de sistema operativo para los robots, el equipo del laboratorio lo utilizará principalmente para el desarrollo de la simulación de Autos Autónomos.

Se pretende participar dentro del Torneo Mexicano de Robótica, en la categoría AutoModelCar, la cuál actualmente cuenta con una subcategoría virtual.

A continuación se desarrollarán los siguientes puntos:

1. Requerimientos de sistema
2. Software de instalación
3. Proceso de instalación
4. Primeros pasos para la ejecución de ROS
5. Pasos para ejecutar Webots desde ROS

Requerimientos de sistema

ROS trabaja sobre el sistema Linux, aunque hay otras versiones de ROS que se pueden trabajar en Windows las que ocupamos en el laboratorio se trabajan en Linux, en específico la distribución Ubuntu. Aunque no especifica un mínimo de requerimientos para que funcione es recomendable que se cuente con un sistema reciente y una tarjeta gráfica, ya que las simulaciones que se realizan son en tiempo real.

Los siguientes requerimientos mínimos, son necesarios:

- **Hardware:**
  - Procesador: 2.5 GHz de doble núcleo o superior.
  - Memoria RAM: Se recomiendan al menos 2 GB de RAM, pero para un rendimiento óptimo, se sugieren 4 GB o más.
  - Espacio en disco: Se recomiendan al menos 20 GB de espacio libre en disco para la instalación y el desarrollo de paquetes.
  - Tarjeta gráfica: No se requiere una tarjeta gráfica dedicada, a menos que estés trabajando en aplicaciones de simulación y visualización 3D intensivas.

- **Sistema Operativo:**
    - Ubuntu 18.04 LTS (Bionic Beaver).

- **Red:**
    - Una conexión a Internet activa para descargar e instalar paquetes y dependencias.

- **Software:**
    - Un entorno de desarrollo (como el terminal) para ejecutar comandos.
    - Herramientas básicas de compilación (instaladas con “build essential”.
    - Python: ROS depende de Python, y en Ubuntu 18.04, Python 2.7 y Python 3.6+ están preinstalados.

- **Gráficos (opcional):**
    - Para aplicaciones que involucren visualización 3D o simulación, como RViz o Gazebo, una tarjeta gráfica compatible y controladores actualizados pueden mejorar la experiencia.

Hay que recordar que ROS es un sistema operativo modular, lo cual significa que se pueden instalar solo los paquetes que se requieran utilizar, sin embargo en esta guía se busca tener una instalación completa.

Software de instalación

La versión de ROS depende mucho de la versión de Ubuntu que se instale en el sistema, si es Ubuntu 18.04 le corresponde la versión de ROS melodic, mientras que para las computadoras con la versión de Ubuntu de 20.04 le corresponde la versión de ROS noetic.

Actualmente la comunidad de ROS ha creado la versión 2 del sistema, para los que cuentan con Ubuntu 22.04 se recomienda instalar ROS2 Humble, está versión viene optimizada generando algunos cambios y facilidades de uso, más adelante se darán algunos detalles respecto a esta versión.

Es importante mencionar que ROS es un sistema que va a permitir crear un ambiente de trabajo en el que el sistema robótico logrará generar todas sus acciones por medio de nodos que serán ejecutados de forma simultánea, paralela y concurrente.

Proceso de instalación

Una vez identificado la versión del software procede con la instalación siguiendo las siguientes instrucciones:

1. Instalado el sistema operativo de ubuntu, hay que asegurarse que el administrador de paquetes esté actualizado, así que, desde la terminal se escribirá la siguiente instrucción: “sudo apt update”.

2. Finalizada la actualización, hay que escribir la siguiente instrucción: “sudo apt install ros-(la versión que vamos a instalar)-desktop-full”. Si se requiere más información acerca de qué versiones de ROS hay se puede introducir lo siguiente: “apt search ros-(la versión de ROS de la cual se requiere más información).” En caso de tener duda respecto a la versión del ROS de interés se puede acudir directamente a la página web y copiar la instrucción de instalación.

3. Es posible que se requiera instalar algunas dependencias adicionales y herramientas útiles. Estas se pueden instalar con el siguiente instrucción: “sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential”.

Ejemplo de instalación de ROS, supongamos que tenemos instalado un Ubuntu 20.04 LTS, la versión de ROS que corresponde a este sistema es ROS noetic, en una terminal realicemos lo siguiente:

- ➢ Dentro de una terminal, primero aceptamos los paquetes de ROS
  - `sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`
- ➢ Después configuramos las claves con curl
  - `sudo apt install curl # if you haven't already installed curl`
  - `curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -`
- ➢ Iniciamos el proceso de instalación
  - `sudo apt update`
  - `sudo apt install ros-noetic-desktop-full`
- ➢ Configura tu entorno de desarrollo
  - `source /opt/ros/noetic/setup.bash`
  - `echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc`
  - `source ~/.bashrc`
- ➢ Instala las dependencias para el desarrollo de ROS con python
  - `sudo apt install python3-rosdep`

- ➢ Inicializa las dependencias
  - `sudo rosdep init`
  - `rosdep update`

Primeros pasos para la ejecución de ROS

Cuando el proceso de instalación finalice, hay que realizar un proceso de verificación del sistema, para eso se realizará lo siguiente:

- ➢ Verificar si las variables de entorno están disponible
  - `printenv | grep ROS`
- ➢ Activemos los comandos de ROS
  - `source /opt/ros/noetic/setup.bash`
- ➢ Iniciamos construyendo un catkin workspace:
  - `mkdir -p ~/catkin_ws/src`
  - `cd ~/catkin_ws/`
  - `catkin_make`
- ➢ Lo anterior genera la construcción de un proyecto, en donde hay un conjunto de carpetas de las cuales las principales son: src y devel. Hay que ingresar en devel y hacer lo siguiente:
  - `source devel/setup.bash`
- ➢ Para asegurarse de que su espacio de trabajo esté correctamente superpuesto por el script de configuración, asegúrese de que la variable de entorno ROS_PACKAGE_PATH incluya el directorio en el que se encuentra.
  - `echo $ROS_PACKAGE_PATH`
    `/home/youruser/catkin_ws/src:/opt/ros/noetic/share`
- ➢ Instalemos los paquetes del tutorial de ROS
  - `sudo apt-get install ros-noetic-ros-tutorials`

A continuación mostraremos algunas instrucciones de ROS

- ➢ rospack → Permite obtener información de los paquetes del proyecto
  - `rospack find roscpp`
  - Si está bien instalado regresará un resultado como: `/opt/ros/noetic/share/roscpp`
- ➢ roscd → nos permitira movernos entre directorios de ROS
  - `roscd roscpp`
  - `pwd`
- ➢ rosls → nos permitirá ingresar al directorio si colocar la ruta absoluta
  - `roscd roscpp_tutoriales/`

Pasos para ejecutar webots desde ROS

Supongamos que ya contamos con un ambiente desarrollado en ROS y requerimos probar su funcionamiento, como lo que se propone en la categoría AutoModelCar del TMR.

Es importante mencionar que es necesario tener instalado de forma previa webots. Instala desde terminal como:

- `sudo mkdir -p /etc/apt/keyrings`
- `cd /etc/apt/keyrings`
- `sudo wget -q https://cyberbotics.com/Cyberbotics.asc`
- `echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/Cyberbotics.asc] https://cyberbotics.com/debian binary-amd64/" | sudo tee /etc/apt/sources.list.d/Cyberbotics.list`
- `sudo apt update`
- `sudo apt install webots`

Para comenzar un proyecto que visualice su ejecución en webots, se debe utilizar la instrucción: roslaunch

Supongamos que se trabaja con el proyecto de AutoModelCar, donde el escenario y el robot están simulados en webots, es necesario realizar la ejecución desde el espacio de trabajo de ROS:

- Primero clonamos el proyecto en cuestión
  - git clone https://github.com/mnegretev/TMR-2022-AutoModelCar
  - cd TMR-2022-AutoModelCar
  - cd catkin_ws
  - catkin_make -j2 -l2
  - echo "source ~/TMR-2022-AutoModelCar/catkin_ws/devel/setup.bash" >> ~/.bashrc
  - source ~/.bashrc
- Ahora es necesario abrir el ambiente de trabajo:
  - roslaunch bring_up navigation_no_obstacles.launch

La instrucción roslaunch tiene indicado el nombre del paquete del proyecto a utilizar, y el nombre del archivo que contiene el escenario y robot de webots, la extensión de dicho archivo es .launch.

Cada instrucción de ROS indicada en la terminal que implique algún cambio en el comportamiento del robot se verá reflejada en la simulación de webots.

Las instrucciones de ROS están dadas por Tópicos que son en realidad funciones disponibles y creadas para el proyecto en cuestión.

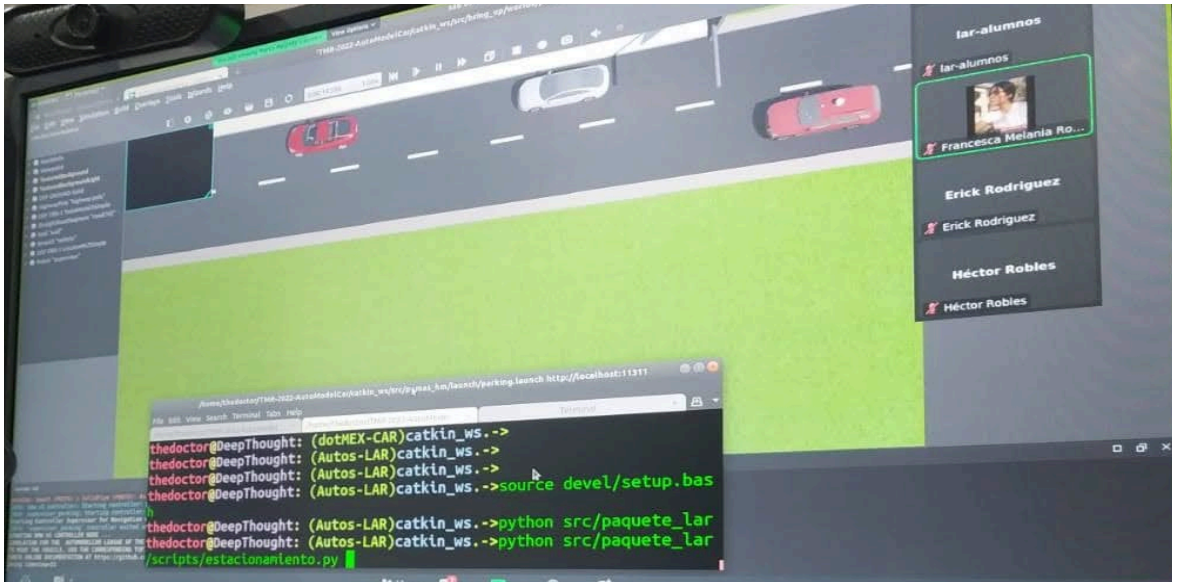
La programación en ROS sigue un comportamiento cliente servidor en la que existe un nodo maestro y el resto son esclavos, los esclavos son los clientes que solicitan servicios, manteniendo así comunicación entre los tópicos disponibles de forma paralela y concurrente.

Dentro del recurso del proyecto, una vez activo debe realizarse la ejecución de la solución propuesta, esta puede ser desarrollada en un paquete nuevo y en un archivo de python. Para ejecutarse y actuar en la simulación, la instrucción podría ser por ejemplo, un paquete generado llamado paquete_lar, en donde la solución es agregada dentro de una carpeta llamada scripts, el archivo con las instrucciones en python se llama estacionamiento.py.

La instrucción para ejecutar todo sería como sigue, estando dentro del proyecto catkin_ws:

- ➢ source devel/setup.bash
- ➢ python src/paquete_lar/scripts/estacionamiento.py

Una vez realizado el enter corre la ejecución del comportamiento en el simulador de webots.

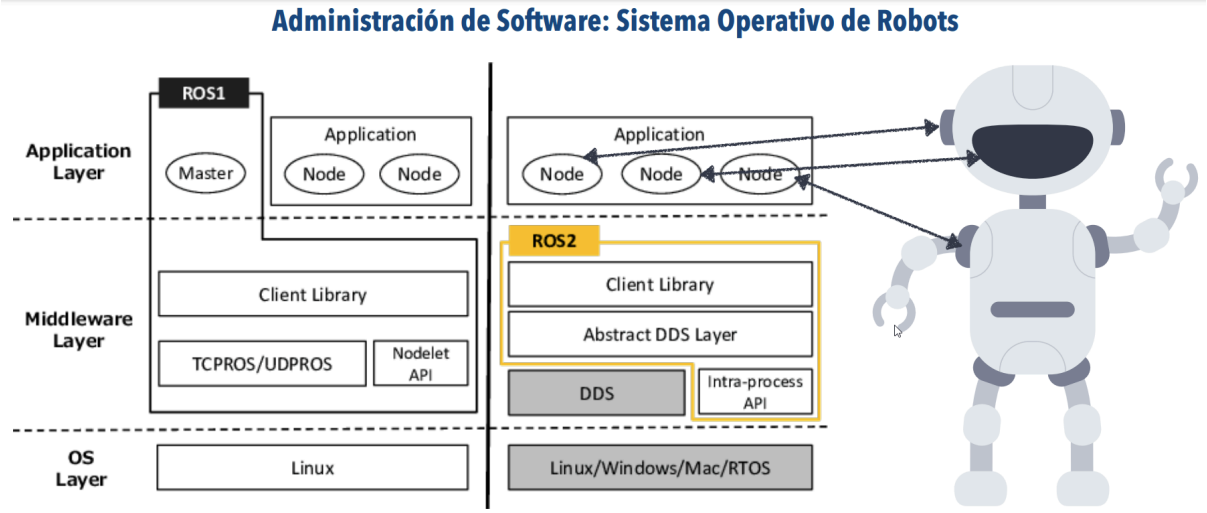


Ejecución de estacionamiento.py en ROS y webots.

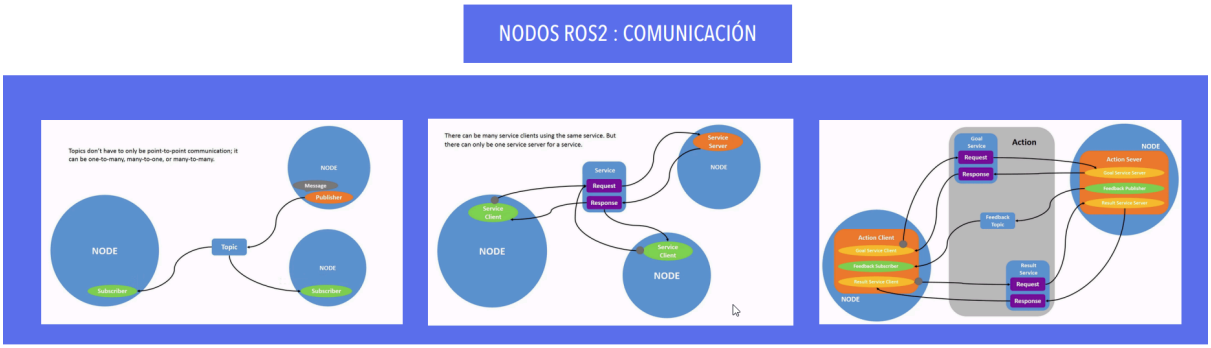
Actualización

En la actualidad se está utilizando una nueva versión de ROS y es ROS2, en esta se marca una diferencia respecto al esquema de trabajo de la versión actual. En esta la esencia es generar nodos super conectados, dejando de lado el comportamiento de maestro y esclavo. La idea es tener los servicios disponibles y

que cualquier nodo dentro del mismo ambiente pueda conectarse y resolver su problema en específico.



Comparativo entre ROS1 y versión de ROS2



Formas de comunicación en ROS2, dentro de la interfaz de desarrollo.

Realizar la actualización hace que el manejo de ROS para programar comportamientos en los robots cada vez sea más sencilla, en cuanto a configuraciones del sistema y más rápida en cuestiones de comunicación entre servicios entre solicitud y respuesta. Todo este sistema busca en general, realizar ejecuciones en tiempo real, para simular un comportamiento más activo-reactivo.

En esta versión se sigue utilizando los Tópicos, además de una definición más precisa entre la construcción de nodos de comunicación en las aplicaciones a desarrollar.

Conclusiones

ROS al ser un sistema desarrollado por una comunidad científica, y en busca de convertirse en un sistema estándar para el desarrollo de aplicaciones para robots de servicio, año con año y versión a versión de sistemas operativos Ubuntu, va cambiando y actualizando sus versiones, por lo que es importante estar pendientes de los cambios anuales.

Fuentes de consulta

- http://wiki.ros.org/noetic/Installation/Ubuntu
- http://wiki.ros.org/ROS/Tutorials
- http://wiki.ros.org/webots_ros
- https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html
- Diplomado, Robótica de Servicio, Dr. Héctor S Vargas Mtz Lab. Control y Robótica Facultad de Electrónica. (Otoño 2020)