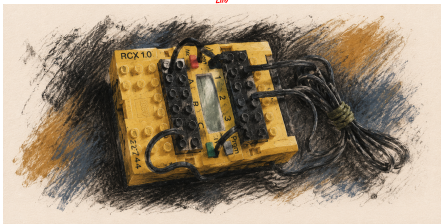
Entorno de desarrollo y primer sistema ROS 2

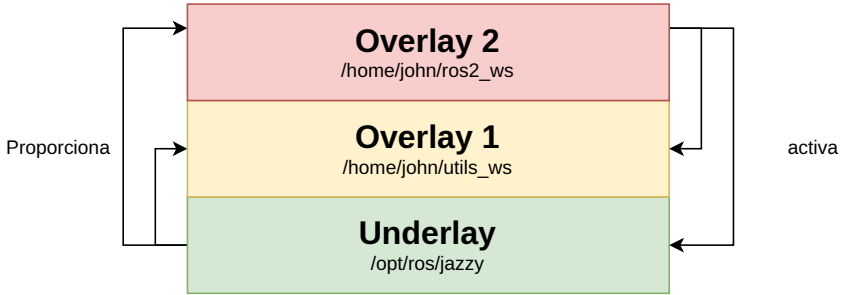
Teoría para las prácticas

Francisco Martín Rico y Rodrigo Pérez Rodríguez



Workspace, paquetes, underlay y overlay

- ROS 2 organiza el software en *workspaces* con código, compilación y ejecución.
- El paquete es la unidad básica de distribución en ROS 2.
- El **underlay** contiene los paquetes ROS 2 instalados en el sistema (/opt/ros/<distro>).
- El **overlay** es el workspace del usuario con paquetes propios o de terceros.



Activación del entorno y persistencia

- Activar ROS 2 significa configurar variables de entorno del sistema.
- La activación se realiza mediante el script `setup.bash` que se genera en el directorio `install` del workspace.
- Primero se activa el underlay de ROS 2 instalado.
- Después se activa el overlay del workspace del usuario.
- Sin activación, ROS 2 no puede localizar paquetes ni ejecutables.
- La activación puede hacerse persistente mediante el fichero `.bashrc`.

```
source /opt/ros/jazzy/setup.bash
source install/setup.bash
```

Compilación del workspace

- Un workspace es un directorio que contiene múltiples paquetes ROS 2 bajo su carpeta `src`.
- Los workspaces se compilan con la herramienta colcon.
- La compilación siempre se realiza desde el directorio raíz del workspace.
- La opción `-symlink-install` evita copias innecesarias de ficheros (recomendada)
- Crea el directorio `install` con los paquetes compilados y configurados.
- Tras compilar, es necesario recargar el entorno para activar los nuevos paquetes.
- Se limpia borrando `build`, `install` y `log` si es necesario.

```
colcon build --symlink-install
```

Herramientas de línea de comandos (ros2cli)

- `ros2cli` permite inspeccionar y controlar sistemas en ejecución.
- El comando base sigue el patrón mostrado a continuación.
- Cada comando se asocia a un dominio del sistema.
- Permite analizar sistemas sin modificar el código.
- El autocompletado con TAB facilita la exploración interactiva.

```
ros2 <comando> <verbo>

ros2 node list
ros2 topic list
ros2 topic echo /scan
```

Ejecución de nodos en ROS 2

- Un nodo es la unidad básica de ejecución en ROS 2.
- El subcomando **run** ejecuta un único nodo directamente.
- No es necesario conocer la ruta del ejecutable.
- El subcomando **launch** ejecuta múltiples nodos coordinados.
- Los launch files gestionan parámetros, remapeos y dependencias.

```
ros2 run <paquete> <ejecutable>
ros2 launch <paquete> <archivo.launch.py>
```

Interfaces y parámetros

- Las interfaces definen cómo se comunican los nodos.
- Incluyen mensajes, servicios y acciones.
- Los parámetros permiten configurar nodos sin modificar código.
- Cambiar parámetros altera el comportamiento en tiempo de ejecución.
- Comprender interfaces y parámetros es clave para analizar sistemas ROS 2.

Paquetes y ejecutables

- Un paquete puede contener nodos, librerías, interfaces y configuraciones.
- Mostrar paquetes disponibles y descubrir ejecutables.
- Es clave para explorar sistemas sin documentación previa.

```
ros2 pkg list
ros2 pkg executables <paquete>
```

Nodos y grafo de computación

- Los nodos se comunican mediante topics, servicios y acciones.
- Listar nodos activos e inspeccionar sus interfaces y parámetros.
- Permite entender el papel de cada nodo en el sistema.
- Es fundamental para analizar el grafo de computación del robot.

```
ros2 node list
ros2 node info <nodo>
```

Topics e inspección de datos

- Los topics son canales asíncronos de intercambio de datos.
- Listar topics, consultar tipos y ver datos en tiempo real.
- Es la forma más rápida de verificar que un nodo funciona.

```
ros2 topic list
ros2 topic type /scan
ros2 topic echo /scan
```

Interfaces y parámetros en práctica

- Inspeccionar interfaces y parámetros desde terminal.
- Los parámetros son esenciales en simulación y depuración.

```
ros2 interface list
ros2 interface show <interfaz>

ros2 param list
ros2 param get <nodo> <parametro>
ros2 param set <nodo> <parametro> <valor>
```

Flujo típico de trabajo en la práctica

- Activar underlay y overlay.
- Compilar el workspace.
- Recargar el entorno tras la compilación.
- Ejecutar nodos o simuladores.
- Analizar el sistema con `ros2cli` y RViz2.

```
source /opt/ros/jazzy/setup.bash
source install/setup.bash

colcon build --symlink-install
source install/setup.bash

ros2 launch <paquete> <archivo.launch.py>
```

Simulador del Kobuki

- El simulador se distribuye como paquetes ROS 2 que se compilan en el overlay del alumno.
- El código fuente se obtiene clonando el repositorio en `src`.
- Tras compilar, hay que recargar el overlay para que ROS 2 encuentre el paquete.
- Se lanza con un `launch` (con o sin interfaz gráfica).

```
git clone https://github.com/IntelligentRoboticsLabs/ASR-Software.git
colcon build --symlink-install
source install/setup.bash

ros2 launch kobuki simulation.launch.py
# Alternativa sin GUI (equipos con pocos recursos)
ros2 launch kobuki simulation.launch.py gui:=false
```

Visualización con RViz2

- RViz2 es la herramienta estándar de visualización en ROS 2.
- Permite inspeccionar sensores, mapas y estados del robot.
- Es necesario fijar correctamente el `Fixed Frame`.
- Los datos del láser se visualizan mediante `LaserScan`.
- Confirma visualmente que el sistema funciona correctamente.

Preguntas