ROS Tutorial - Session 2: Expanding Your ROS Skills

Welcome to Session 2 of our ROS Tutorial series! Building on the foundations laid in Session 1, this session will dive deeper into ROS capabilities, focusing on creating and debugging Python nodes, setting up a ROS workspace, understanding the node graph, and obtaining node data and information. We'll also explore what ROS is and what you can achieve with it.

Overview

- **Setting Up Your Development Environment:** Installing Visual Studio Code extensions to enhance your coding experience.
- **Creating a ROS Workspace:** Steps to initialize and build your workspace.
- **Developing ROS Packages:** How to create and configure a ROS package.
- **Programming with Python in ROS:** Guidelines for writing ROS nodes in Python.
- **Understanding and Utilizing the ROS Node Graph:** Tools and commands for visualizing and interacting with the graph.
- **Session Objectives:**
    - Install necessary tools and extensions for ROS development.
    - Create and build a ROS workspace.
    - Develop and debug Python-based ROS nodes.
    - Visualize and analyze the ROS node graph.
    - Retrieve data and information about ROS nodes.

Getting Started

Install Visual Studio Code Extensions

To enhance your ROS development experience, we recommend installing the following Visual Studio Code extension:

- **Python by Microsoft:** This extension provides comprehensive support for Python programming, including features such as debugging, linting, IntelliSense, and more, making it essential for ROS development.
    - Python - VS Marketplace Link
- **ROS by Microsoft:** Specifically designed for ROS projects, this extension facilitates ROS development in Visual Studio Code by offering features like ROS command integration, debugging, and environment configuration.
    - ROS - VS Marketplace Link

Setting Up a ROS Workspace

Create your ROS workspace to organize your projects and packages:

```
mkdir -p ~/classROSws/src
cd ~/classROSws
catkin_make
```

Remember to source the setup script to make your terminal aware of the new workspace:

```
source ~/classROSws/devel/setup.bash
```

To automatically source it in every new terminal session, add it to your `.bashrc` file:

```
echo "source ~/classROSws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Creating a ROS Package

Initialize a new ROS package with necessary dependencies:

```
cd ~/classROSws/src
catkin_create_pkg testSubPub std_msgs rospy roscpp
```

Adjust the `package.xml` to include details like the author, dependencies, and more.

Writing a Python Node in ROS

When writing Python scripts for ROS, ensure they're executable:

```
chmod +x myscript.py
```

When developing a Python node, such as a publisher, it's important to follow a structured template to ensure compatibility with ROS. Additionally, to prevent any version-related errors, especially in systems with multiple Python versions installed, it's essential to use the Python version that was used during the ROS installation process. For example, Ubuntu 20.04 typically uses Python 3.8.10 by default.

Here's a basic template for a ROS publisher node in Python:

```
#!/usr/bin/env python3
# The shebang line above ensures that the script is executed using the
Python interpreter specified,
# which is particularly important in environments with multiple Python
versions.

import rospy  # Import the rospy client library for Python.

if __name__ == '__main__':
    rospy.init_node('publisher_node', anonymous=True)  # Initialize the ROS
node with a unique name.
```

```
    rospy.loginfo('Hello world')  # Log information to the console.
    # Additional functionality for publishing messages would go here.
```

To run your Python node:

```
rosrun testSubPub pub_python.py
```

Visualizing the ROS Node Graph

Understanding the connections between nodes is crucial. Use `rqt_graph` to visualize the node graph, and `rosnode info /nodename` to get detailed information about specific nodes.

Looking Ahead: Session 3

Prepare for an exciting continuation as we go deeper into robot modeling with URDF, simulation with Gazebo, and controlling nodes with RViz in Session 3. Stay tuned for more in-depth exploration of ROS functionalities!

✒️ Crafted by

Sadegh-Kalami

🔗 **Connect with me:**

- **GitHub Profile**: collaborations at Sadegh-Kalami.
- **Personal Site**: My projects and portfolio at Sadegh-Kalami's GitHub Page.

Your journey through my digital craftsmanship begins here.
