**Different Topics and message types in ROS**

**1./cmd\_vel**

* **Message Type**: geometry\_msgs/Twist
* **Description**: Used to send velocity commands to mobile robots. It typically includes linear and angular velocity components to control movement.

**2. /odom**

* **Message Type**: nav\_msgs/Odometry
* **Description**: Provides information about the robot's position and orientation in space, often used for navigation and localization.

**3. /scan**

* **Message Type**: sensor\_msgs/LaserScan
* **Description**: Contains LIDAR scan data, including ranges and angles for each reading, useful for obstacle detection and mapping.

**4. /camera/image\_raw**

* **Message Type**: sensor\_msgs/Image
* **Description**: Raw image data from a camera. This topic is used for computer vision tasks and image processing.

**5. /camera/camera\_info**

* **Message Type**: sensor\_msgs/CameraInfo
* **Description**: Contains intrinsic and extrinsic parameters of the camera, such as focal length and distortion coefficients, essential for image processing.

**6. /tf**

* **Message Type**: tf2\_msgs/TFMessage
* **Description**: Used for broadcasting and receiving transformations between different coordinate frames (e.g., robot base, camera, and sensors).

**7. /robot\_state**

* **Message Type**: std\_msgs/String (or a custom message type)
* **Description**: Represents the current state of the robot (e.g., idle, moving, error states). Custom messages can be used to include more detailed information.

**8. /joint\_states**

* **Message Type**: sensor\_msgs/JointState
* **Description**: Contains information about the position, velocity, and effort of each joint in a robot arm or manipulator.

**9. /move\_base/goal**

* **Message Type**: move\_base\_msgs/MoveBaseActionGoal
* **Description**: Used to send goals for navigation to the robot's base, enabling it to move to specified locations.

**10. /move\_base/status**

* **Message Type**: actionlib\_msgs/GoalStatusArray
* **Description**: Provides the status of goals sent to the move\_base action server, indicating whether the robot is active, succeeded, or failed.

**11. /battery\_state**

* **Message Type**: sensor\_msgs/BatteryState
* **Description**: Contains information about the robot’s battery status, such as voltage, current, and charge level.

**12. /imu/data**

* **Message Type**: sensor\_msgs/Imu
* **Description**: Provides data from an Inertial Measurement Unit (IMU), including orientation, angular velocity, and linear acceleration.

**13. /joint\_trajectory**

* **Message Type**: trajectory\_msgs/JointTrajectory
* **Description**: Used to specify a trajectory for robotic joints, allowing for complex movements in a sequence.

**14. /controller/state**

* **Message Type**: controller\_manager\_msgs/ControllerState
* **Description**: Provides the state of a controller managing the robot's joints or movements, useful for monitoring and debugging.

**Custom Messages**

In addition to standard messages, ROS allows users to define custom message types to suit specific applications. These messages can be composed of other messages and tailored to include specific fields relevant to the application.

**How Messages Work**

* **Publishing**: Nodes publish messages to a topic using a publisher object. The messages are sent out over the ROS network.
* **Subscribing**: Nodes subscribe to a topic using a subscriber object to receive messages.
* **Message Formats**: Messages are defined in .msg files, which specify the fields and types included in the message.

**Services and Actions**

**Services**

Services in ROS provide a way for nodes to make synchronous requests and receive responses. Each service is defined by a .srv file that specifies the request and response formats.

**1. /add\_two\_ints**

* **Service Type**: example\_msgs/AddTwoInts
* **Description**: A simple service that takes two integers as input and returns their sum. Often used as a basic example for service implementation.

**2. /clear**

* **Service Type**: std\_srvs/Empty
* **Description**: Clears the current state of the map or visualization (e.g., in RViz). It doesn't require any parameters.

**3. /reset**

* **Service Type**: std\_srvs/Empty
* **Description**: Resets the state of a node or system, typically used in simulation environments.

**4. /set\_parameters**

* **Service Type**: dynamic\_reconfigure/Config
* **Description**: Allows dynamic adjustment of parameters in real-time, enabling tuning of algorithms without restarting nodes.

**5. /move\_base/make\_plan**

* **Service Type**: move\_base\_msgs/MakePlan
* **Description**: Requests a path plan from the current position to a goal position. Useful in navigation tasks for mobile robots.

**Actions**

Actions in ROS are designed for tasks that can take time to complete. They allow for feedback during execution and can be preempted. Each action is defined by a .action file, specifying the goal, result, and feedback messages.

**1. /move\_base**

* **Action Type**: move\_base\_msgs/MoveBaseAction
* **Description**: Allows the robot to move to a specified goal location. Provides feedback during the movement process and allows for preemption.

**2. /grasp**

* **Action Type**: moveit\_msgs/Grasp
* **Description**: Controls a robotic arm to grasp an object. Provides feedback on the grasping process and can indicate success or failure.

**3. /follow\_joint\_trajectory**

* **Action Type**: control\_msgs/FollowJointTrajectoryAction
* **Description**: Sends a trajectory for a robot's joints to follow, providing feedback and allowing for adjustments during execution.

**4. /pick**

* **Action Type**: moveit\_msgs/PickAction
* **Description**: Controls the robot to pick up an object, providing feedback and allowing for retries in case of failure.

**5. /place**

* **Action Type**: moveit\_msgs/PlaceAction
* **Description**: Instructs the robot to place an object down at a specified location, similar to the pick action.