EE405A
Robotics Operating System (ROS) - 3

(TA) Daegyu Lee
School of Electrical Engineering
KAIST

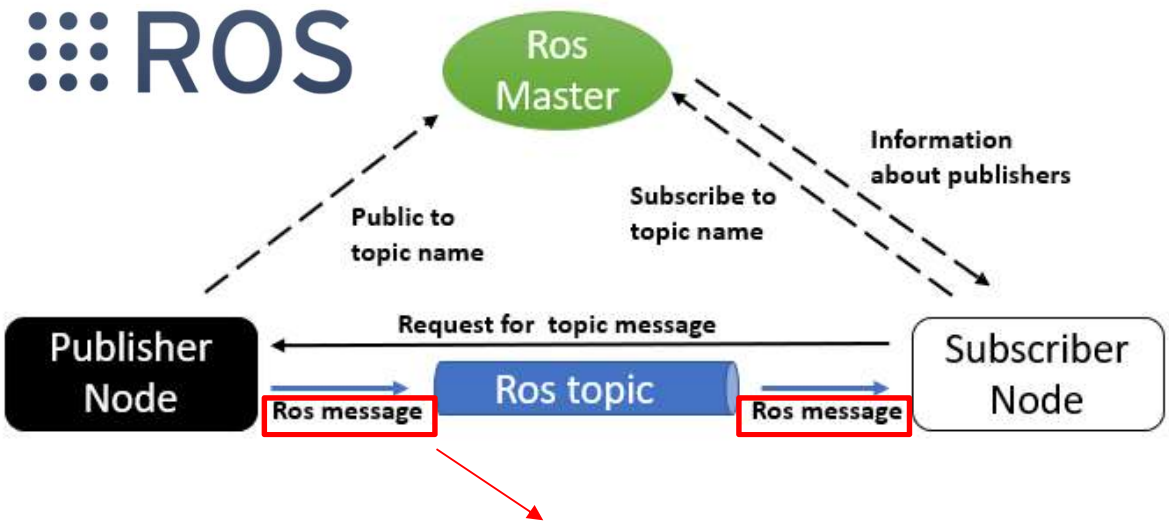
March 24, 2021

lee.dk@kaist.ac.kr

Experiment Objectives

In this week, you will do the following:

- Understand ROS messages
- Make your Custom ROS message.



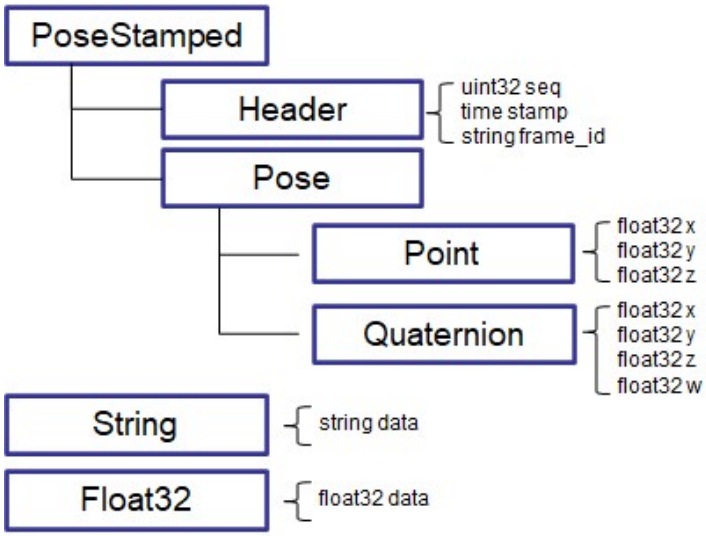
What is the ROS messages?

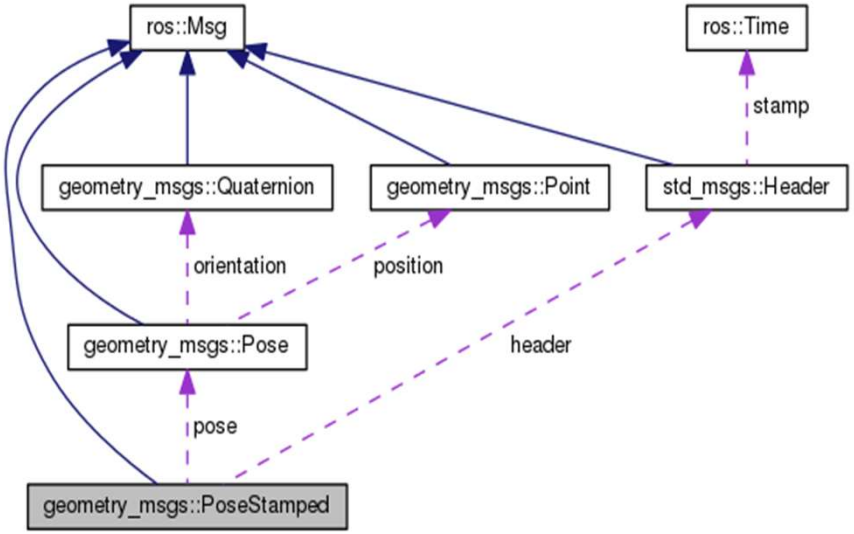
ROS Message Type

ROS Message Type

➢ ROS Message Type

- ❑ A message is a simple data structure, comprising typed fields.
- ❑ In message type, standard primitive types (Integer, Floating point, Boolean, etc.) are supported, as are arrays of primitives types.
- ❑ Messages can be arbitrary nested structures and arrays (much like C structs).





Reference : http://api.kittcar.com/kitt_platform/html/classgeometry__msgs_1_1PoseStamped.html

ROS Message Type

➢ ROS Message Type

- ❑ Hierarchical characteristics of the ROS message type.

geometry_msgs/PoseStamped Message

File: geometry_msgs/PoseStamped.msg

Raw Message Definition

```
# A Pose with reference coordinate frame and timestamp
Header header
Pose pose
```

Compact Message Definition

```
std_msgs/Header header
geometry_msgs/Pose pose
```

geometry_msgs/Pose Message

File: geometry_msgs/Pose.msg

Raw Message Definition

```
# A representation of pose in free space, composed of position and orientation.
Point position
Quaternion orientation
```

Compact Message Definition

```
geometry_msgs/Point position
geometry_msgs/Quaternion orientation
```

geometry_msgs/Point Message

File: geometry_msgs/Point.msg

Raw Message Definition

```
# This contains the position of a point in free space
float64 x
float64 y
float64 z
```

Compact Message Definition

```
float64 x
float64 y
float64 z
```

geometry_msgs/Quaternion Message

File: geometry_msgs/Quaternion.msg

Raw Message Definition

```
# This represents an orientation in free space in quaternion form.

float64 x
float64 y
float64 z
float64 w
```

Compact Message Definition

```
float64 x
float64 y
float64 z
float64 w
```

Reference : http://docs.ros.org/en/melodic/api/geometry_msgs/html/msg/PoseStamped.html

ROS Message Type

➢ Accessing ROS Message Type Information

- ❑ 'rosmsg show' command
  - ✓ Add '-r' option to display the raw msg definition
- ❑ Visit or Google ROS message documentation
  - ✓ http://wiki.ros.org/common_msgs

```
sw@sw-Blade:~$ rosmsg show geometry_msgs/PoseStamped
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

➢ Discovering the Message Type of a ROS Topic

- ❑ 'rostopic info' command
  - ✓ Type
  - ✓ Publishing Nodes
  - ✓ Subscribing Nodes

```
sw@sw-Blade:~$ rostopic info /scout/mavros/vision_pose/pose
Type: geometry_msgs/PoseStamped

Publishers:
 * /scout/tf_listener (http://localhost:37495/)

Subscribers: None
```

what is this topic named "/scout/mavros/vision_pose/pose"?

Reference : http://docs.ros.org/en/melodic/api/geometry_msgs/html/msg/PoseStamped.html

ROS Messages

ROS Messages

- **ROS messages(rosmsg)**

Reference : http://wiki.ros.org/msg

  - ROS uses a simplified messages description language for describing the data values.
  - This description makes it easy for ROS tools to automatically generate source code for the message type in several target languages.
  - The format of this language is simple: a message description is **a list of data** field descriptions

Reference : http://docs.ros.org/en/diamondback/api/std_msgs/html/index-msg.html

Built-in types

Primitive Type	Serialization	C++	Python2	Python3
bool (1)	unsigned 8-bit int	uint8_t (2)	bool	
int8	signed 8-bit int	int8_t	int	
uint8	unsigned 8-bit int	uint8_t	int (3)	
int16	signed 16-bit int	int16_t	int	
uint16	unsigned 16-bit int	uint16_t	int	
int32	signed 32-bit int	int32_t	int	
uint32	unsigned 32-bit int	uint32_t	int	
int64	signed 64-bit int	int64_t	long	int
uint64	unsigned 64-bit int	uint64_t	long	int
float32	32-bit IEEE float	float	float	
float64	64-bit IEEE float	double	float	
string	ascii string (4)	std::string	str	bytes
time	secs/nsecs unsigned 32-bit ints	ros::Time	rospy.Time	
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration	

common ROS messages

- std_msgs
- geometry_msgs
- nav_msgs
- visualization_msgs

custom ROS messages

- my_msgs
- your_msgs
- ee405_msgs
- kaist_robot_msgs

ROS Messages

Reference : http://docs.ros.org/en/diamondback/api/geometry_msgs/html/index-msg.html

➢ geometry_msgs

- ❑ geometry_msgs provides messages for common geometric primitives such as points, vectors, and poses. These primitives are designed to provide a common data type and facilitate interoperability throughout the system.

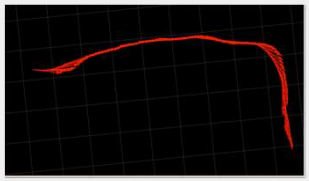
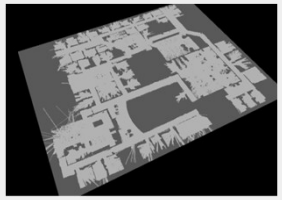
Message	Basic Information	Description	rviz
geometry_msgs/ PoseStamped	Header header Pose pose geometry_msgs/Point position geometry_msgs/Quaternion orientation	A Pose with reference coordinate frame and timestamp	
geometry_msgs/Quaternion	float64 x float64 y float64 z float64 w	This represents an orientation in free space in quaternion form.	-
geometry_msgs/Point	float64 x float64 y float64 z	This contains the position of a point in free space	-
geometry_msgs/Twist	Vector3 linear Vector3 angular	This expresses velocity in free space broken into it's linear and angular parts.	-

ROS Messages

Reference : http://docs.ros.org/en/diamondback/api/nav_msgs/html/index-msg.html

➢ nav_msgs

- ❑ nav_msgs defines the common messages used to interact with the navigation stack.

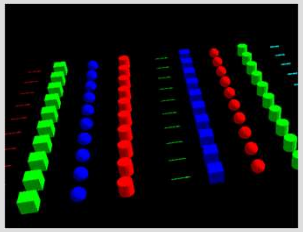
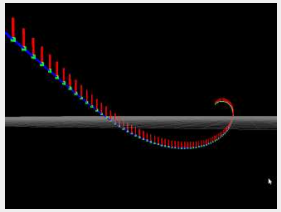
Message	Basic Information	Description	rviz
nav_msgs/Path	Header header geometry_msgs/PoseStamped[] poses	An array of poses that represents a Path for a robot to follow	
nav_msgs/Odometry	Header header string child_frame_id geometry_msgs/PoseWithCovariance pose geometry_msgs/TwistWithCovariance twist	This represents an estimate of a position and velocity in free space.	
nav_msgs/MapMetaData	time map_load_time float32 resolution uint32 width uint32 height geometry_msgs/Pose origin	This hold basic information about the characterists of the OccupancyGrid	-
nav_msgs/OccupancyGrid	Header header MapMetaData info int8[] data	This represents a 2-D grid map, in which each cell represents the probability of occupancy.	

ROS Messages

Reference : http://docs.ros.org/en/melodic/api/visualization_msgs/html/index-msg.html

➢ visualization_msgs

- ❑ visualization_msgs is a set of messages used by higher level packages, such as rviz, that deal in visualization-specific data.

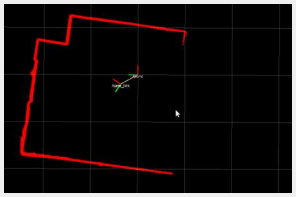
Message	Basic Information	Description	rviz
Visualization_msgs/Marker	std_msgs/Header header string ns int32 id int32 type int32 action geometry_msgs/Pose pose geometry_msgs/Vector3 scale std_msgs/ColorRGBA color duration lifetime bool frame_locked geometry_msgs/Point[] points std_msgs/ColorRGBA[] colors string text string mesh_resource bool mesh_use_embedded_materials	The Markers display allows programmatic addition of various primitive shapes to the 3D view by sending a visualization_msgs/Marker or visualization_msgs/MarkerArray message.	
Visualization_msgs/MarkerArray	visualization_msgs/Marker[] markers	For array of the Marker type.	

ROS Messages

Reference : http://docs.ros.org/en/melodic/api/sensor_msgs/html/index-msg.html

➢ sensor_msgs

- ❑ sensor_msgs is a set of messages made for delivering various input information, ranging from camera, lidar, imu and even joystick.

Message	Basic Information	Description	rviz
sensor_msgs/CompressedImage	std_msgs/Header header string format uint8[] data	Header timestamp should be acquisition time of image Header frame_id should be optical frame of camera origin of frame should be optical center of camera +x should point to the right in the image +y should point down in the image +z should point into to plane of the image	
sensor_msgs/LaserScan	std_msgs/Header header float32 angle_min float32 angle_max float32 angle_increment float32 time_increment float32 scan_time float32 range_min float32 range_max float32[] ranges float32[] intensities	Single scan from a planar laser range-finder	

Make Custom Messages

Make Custom Messages

- **Building .msg Files**

Reference : http://wiki.ros.org/msg

  - The ROS Client Libraries implement message generators that translate .msg files into source code.
  - These message generators must be invoked from your build script(package.xml, CMakeList.txt)

Open package.xml, and make sure these two lines are in it:

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>
```

Description in the wiki is old version

Note that at build time, we need "message_generation", while at runtime, we only need "message_runtime".

Open CMakeLists.txt in your favorite text editor (rosed from the previous tutorial is a good option).

Add the message_generation dependency to the find package call which already exists in your CMakeLists.txt so that you can generate messages. You can do this by simply adding message_generation to the list of COMPONENTS such that it looks like this:

```
# Do not just add this line to your CMakeLists.txt, modify the existing line
find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs message_generation)
```

You may notice sometimes your project builds fine even if you did not call find_package with all dependencies. This is because catkin combines all your projects into one, so if an earlier project calls find_package, yours is configured with the same values. But forgetting the call means your project can easily break when build in isolation.

Also make sure you export the message runtime dependency.

```
catkin_package(
  ...
  CATKIN_DEPENDS message_runtime ...
  ...)
```

- **Open** package.xml **and add these lines :**
  <build_depend>message_generation</build_depend>
  <build_export_depend>message_runtime</build_export_depend>
  <exec_depend>message_runtime</exec_depend>

- **Open** CMakeLists.txt **and add** "message_generation" **in the list of** find_package()

- **Open** CMakeLists.txt **and add** "message_runtime" **in the list of** catkin_package()

Make Custom Messages

- **Building .msg Files** (Cont')

Reference : http://wiki.ros.org/msg

Find the following block of code:

```
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )
```

Uncomment it by removing the # symbols and then replace the stand in `Message*.msg` files with your `.msg` file, such that it looks like this:

```
add_message_files(
  FILES
  Num.msg
)
```

Find the following block of code:

```
# generate_messages(
#    DEPENDENCIES
#    std_msgs  # Or other packages containing msgs
# )
```

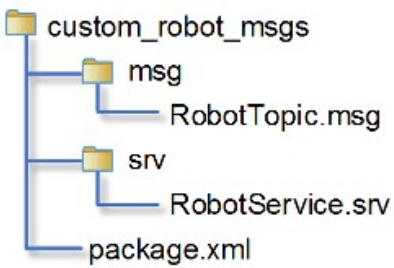
Uncomment it by removing the # symbols and then replace `std_msgs` with the messages your messages depend on, such that it looks like this:

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

- **Write your custom messages**
  Message1.msg
  Message2.msg
  which are stored in **msg** directory.

- **uncomment**
  add_message_files()
  by removing #

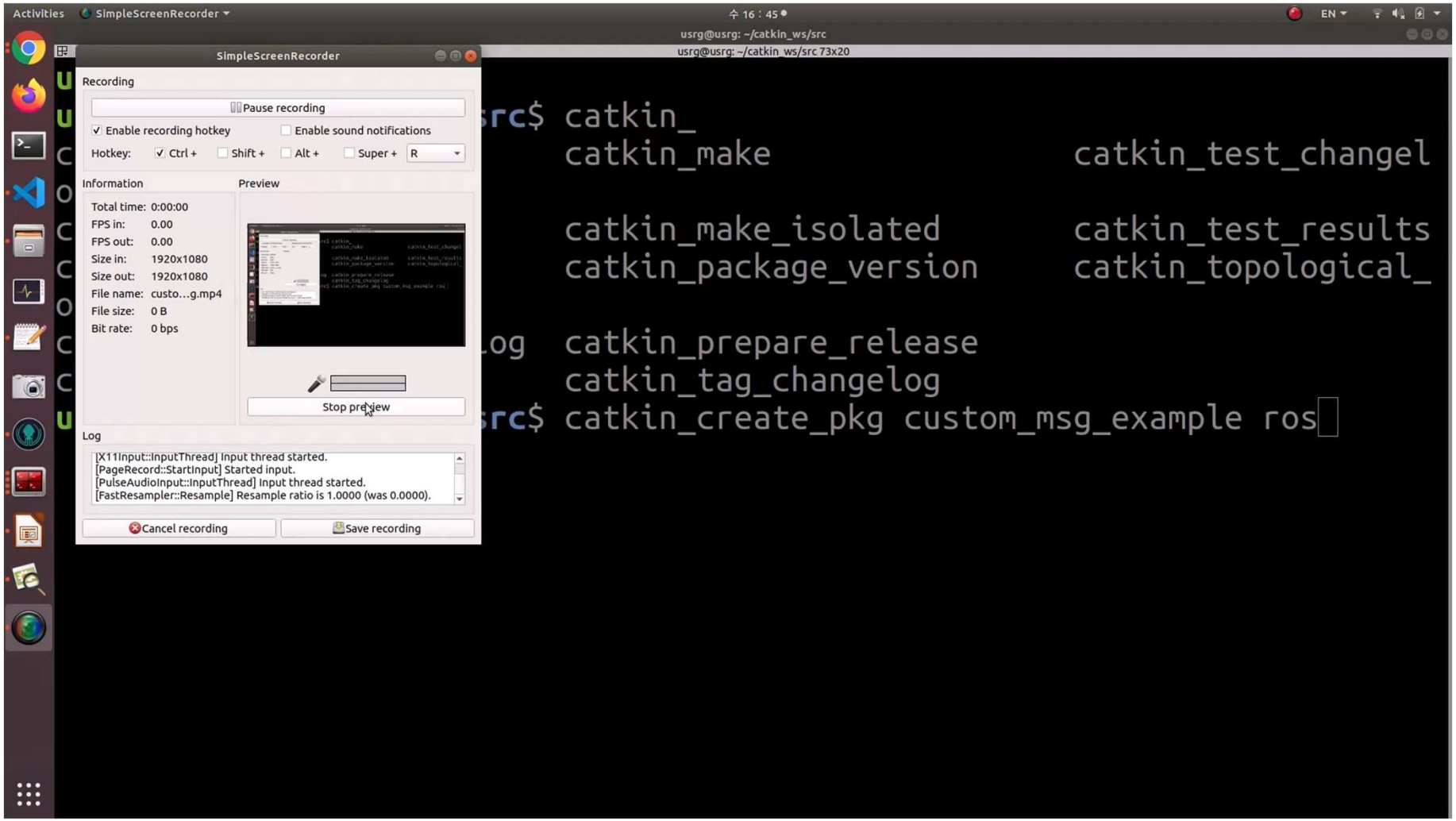
- **uncomment**
  generate_messages()
  by removing #



- **Add your custom messages**
  Message1.msg --> here, RobotTopic.msg
  in **msg** directory.

Make Custom Messages

- **Building .msg Files** (Cont')



Q & A

Feel free to ask me

Email : lee.dk@kaist.ac.kr