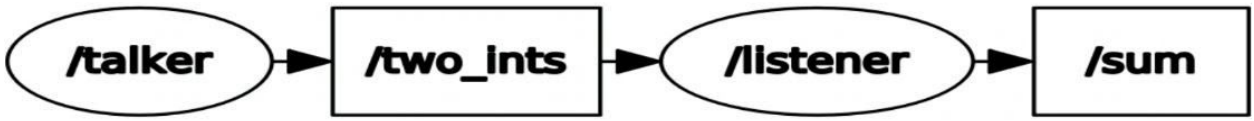
Activity 1

1. Problem description

In this activity, we will create two ROS nodes, a talker node and a listener node. The talker node is a publisher node which will broadcast two random integers at a specified rate on a selected ROS topic called “two_ints”. The listener node is a subscriber node which subscribes to that particular ROS topic and listens to the two integers published. Then it will publish the sum of the two integers on a different ROS topic called “sum”. The architecture of the problem is given below.



2. Creating a ROS package for activity 1

In the ROS-DS environment catkin_ws workspace is already available. The contents of a ROS workspace are organized as ROS packages. Each package is dedicated to a certain task. Each ROS package has a CMakeLists.txt file and package.xml file describing the package. Every ROS package has a CMakeLists.txt and a package.xml file which describes that package. To create a ROS package inside the workspace, browse to the src folder and use the catkin create pkg command as follows.

```
$ cd ~/catkin_ws/src
```

```
# catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

Here you have to set the name of the ROS package as activity_1 with the dependencies std_msgs, rospy and roscpp.

You can later add or modify package dependencies by modifying the package.xml file.

3. Cloning the Python files

Our activity_1 ROS package will consist of two ROS nodes, the publisher and the subscriber. The two incomplete python files can be cloned to the workspace by executing the following terminal commands. Python files are usually kept in the ‘scripts’ folder of the workspace.

```
$ git clone https://github.com/oshadajay/scripts.git
```

4. Creating the custom ROS message

To communicate the two integers first we should create a custom ROS message. ROS messages are stored in the .msg file format inside the 'msg' folder of the ROS package. Let's create the 'TwoInts.msg' file as follows.

```
$ cd ~/catkin_ws/src/activity_1

$ mkdir msg

$ cd msg

$ touch TwoInts.msg
```

ROS msg files are simple text files which contain the field types and field names per line. In our example the TwoInts.msg will contain two integers a and b with the int16 field type. Open the TwoInts.msg file in the editor and modify it accordingly.

When we create a custom ROS msg we should modify the CMakeLists.txt and package.xml files of the ROS package. Open the package.xml file in the editor and add the following two lines.

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

Then open the CMakeLists.txt file in the editor and do the following changes.

First modify the existing text to add message_generation before the closing parenthesis in find_package.

```
find_package(catkin REQUIRED COMPONENTS
   roscpp
   rospy
   std_msgs
   message_generation
)
```

Next find and uncomment the following section and add the filename of the created message.

```
add_message_files(
  FILES
  TwoInts.msg
)
```

Finally find and uncomment the following lines.

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

5. **Creating the publisher node**

   First let's create the publisher node. Open the two_int_talker.py file from the code editor and complete the missing lines and save it. Guidelines to fill the file are given as comments in the code.

6. **Creating the subscriber node**

   Then let's create the subscriber node. Open the two_int_listener.py file from the code editor and complete the missing lines and save it. Guidelines to fill the file are given as comments in the code.

7. **Make the two python files as executables**

   After creating the two python files we have to make those files as executables. This can be done by running the following terminal commands.

```
$ sudo chmod +x ~/catkin_ws/src/activity_1/scripts/two_int_talker.py
```

```
$ sudo chmod +x ~/catkin_ws/src/activity_1/scripts/two_int_listener.py
```

8. **Creating a ROS launch file**

   ROS launch files can be used to start multiple nodes at once. ROS launch files are written in the XML format. Let's create a ROS launch file to start the two_int_talker and the two_int_listener at once by running the following terminal commands. ROS launch files are usually stored in the 'launch' folder of the ROS package.

```
$ cd ~/catkin_ws/src/activity_1
```

```
$ mkdir launch
```

```
$ cd launch
```

```
$ touch execute.launch
```

   Then open the execute.launch file in the editor and add the following contents. Modify the pkg and type of each node by setting the package name to pkg and python script name to type. Save the file.

```
<launch>
    <node pkg="None"
          type="None"
          name="listener"
          output="screen"
    ></node>
    <node pkg="None"
          type="None"
          name="talker"
          output="screen"
    ></node>
</launch>
```

9. Building the workspace

We have now completed writing all the files required for our activity. Let's build the catkin workspace by running the following commands.

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

If there are no errors, the workspace should be built successfully. Afterwards to use the ROS packages inside our workspace we have to source the devel/setup.bash file by running the following terminal commands.

```
$ cd
```

```
$ source ~/catkin_ws/devel/setup.bash
```

10. Launching our application

Now it's the time to launch our application. You can execute ROS launch files by using roslaunch as follows.

```
$ roslaunch activity_1 execute.launch
```

If everything is done correctly, you should see the two random numbers transmitted from the two_int_talker node and the two_int_listener will listen to it and publish the sum of them.

To visualize the nodes and the topics in our application open a new shell and run rqt_graph. Open the graphical tools to see the graph. Remember you have to source the catkin_ws/devel/setup.bash file in every shell.

```
$ rqt_graph
```

You can list the topics available using the following command.

```
$ rostopic list
```

You can listen to any topic using the following command.

```
$ rostopic echo <topic_name>
```

You can visualize the values published in a topic in a graph using rqt_plot and opening the graphical tools.

```
$ rqt_plot
```

Viola! We have completed our first activity successfully.

Activity 2

The second activity is an extension of the first activity. In this activity, our goal is to send a video from a publisher node to a subscriber node on a ROS topic. The incomplete files needed for you to complete your activity are available at,

https://github.com/sakunaharinda/ROS-Handson-Session-1.2.git

Create a new ROSject on the constructsim and upload the contents inside src folder to the catkin_ws workspace. Complete the video_talker.py, video_listener.py and the execute.launch files. Run the execute.launch file and observe the graphical tools. You should see the received video. Take a screenshot of that. Run rqt_graph and take a screenshot too. Attach the two screenshots, talker and listener python files and the execute.launch files to an email and send it to ros.workshop.2021@gmail.com.