Faculty 3ME
DELFT UNIVERSITY OF TECHNOLOGY

ROS Exercise 1
2017-2018

Priadi Teguh Wibowo (4625862)

December 6, 2017

1 Intro

This report explains exercise in Chapter 3: Writing ROS programs from a set of exercises for the book "A Gentle Introduction to ROS". The exercise asks the student to create a new package containing new two nodes.

- The first node is `vel_printer` which subscribes to geometry/Twist messages and prints them on the console.
- The second node is `vel_filter` which subscribes to the messages published by `pubvel` node and then republishes the filtered messages which only has positive angular velocity component.

2 Implementation

The program that creates `pubvel` node can be found in Listing 3.4 from the book. Meanwhile, the program of `vel_printer` node is similar to subscriber example program in Listing 3.5. The differences are only the topic that it subscribes (`geometry/Twist`) and the way it prints the messages (we print x-axis linear velocity and z-axis angular velocity).
Basically, the implementation of `vel_filter` node combines both publisher and subscriber program. First, it creates an object to subscribe the messages published by `pub_vel` node which has topic `turtle1/cmd_vel` then it creates an object to publish the filtered message which has topic `geometry/Twist`. It is shown by code snippet in Listing 1.

Listing 1: Publisher and Subscriber in `vel_filter`

```
// Create a subscriber object.
ros::Subscriber sub = nh.subscribe("turtle1/cmd_vel", 1000,
    &twistMessageReceived);
```

```
   // Create a publisher object.
  ros::Publisher pub =
      nh.advertise<geometry_msgs::Twist>("geometry/Twist",1000);
```

The subscriber object has call back function `twistMessageReceived`. This call back function assigns the z-axis angular velocity and x-axis linear velocity from `turtle1/cmd_vel` messages into global variable `angular_vel` and `linear_vel` respectively. It is shown by code snippet in Listing 2.

Listing 2: Call back function in `vel_filter`

```
// angular velocity
double angular_vel = 0;
double linear_vel = 0;

// A call back function . Executed each time a new
    turtle1/cmd_vel message arrives.
void twistMessageReceived(const geometry_msgs::Twist& msg)
{
  angular_vel = msg.angular.z;
  linear_vel = msg.linear.x;
}
```

Afterwards, the code snippet in Listing 3 shows how to republish filtered messages. Because there is another work other than the callback function, `ros::spinOnce()` is used instead `ros::spin()`. Then the conditional statement `(angular_vel >= 0)` filters the messages and make the program only publishing the messages with positive angular velocity.

Listing 3: Loop in `vel_filter`

```
while(ros::ok()) {
 // Let ROS take over
 ros::spinOnce();
 // only publish when angular velocity is positive
 if (angular_vel >= 0)
 {
  // message for republish
  geometry_msgs::Twist msg;
  msg.linear.x = linear_vel;
  msg.angular.z = angular_vel;
  // Publish the message.
  pub.publish(msg);
 }
 // Wait until it's time for another iteration.
 rate.sleep();
}
```

3 Result

Figure 1 shows the visualization of publish-subscribe relationship between the three nodes.

- The `pubvel` node is represented by `/publish_velocity` node which publish messages with topic `/turtle1/cmd_vel`.
- The `vel_filter` node is represented by `/filter_twist` node which subscribes messages with topic `/turtle1/cmd_vel` and publishes messages with topic `/geometry/Twist`.
- `vel_printer` node is represented by `/subscribe_geometry_Twist` node which subscribes messages with topic `/geometry/Twist`.

Messages `/geometry/Twist` is the filtered version of messages `/turtle1/cmd_vel`.

Figure 1: Relationship between 3 nodes

Figure 2 shows the result of filtering the messages. The left terminal is `vel_printer` node and the right terminal is `pubvel` node. As we can see, the left terminal only prints the positive angular velocity. Meanwhile, the right terminal prints the original random velocity which includes both positive and negative angular velocity.

4 Conclusion

This exercise introduces several fundamental concepts in ROS, which are :

- A ROS package which is a collection of files, executables and other supporting files.
- Nodes which is a running instance of ROS program.
- Topics and messages which are basic entities in communication between ROS nodes.
- Publisher and Subscriber programs which enables communicating between nodes via messages.

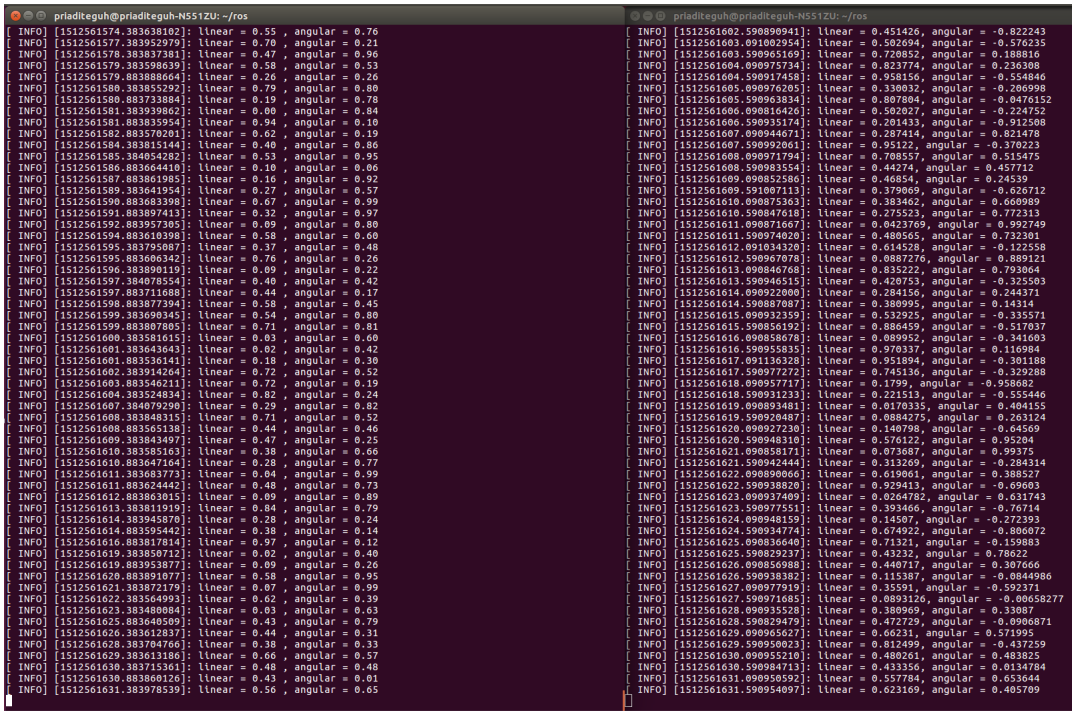

Figure 2: Result comparison between `pubvel` node and `vel_printer` node