Tutorial: Howto Publish to a ROS Topic in the Happy Artist RMDMIA:

In this tutorial we teach Java programmers thow to Publish to a ROS Topic.

System Requirements:

- *ROS*
- *ROS Turtlesim (Configuration instructions)*
- *Java 1.6 or greater*
- ***RMDMIA vpr1_v7 slipstream or greater*

Prerequisites:

- RMDMIA Client Configuration for ROS Turtlesim Tutorial

Overview:

This tutorial explains how to publish to a ROS Topic with the Happy Artist RMDMIA. The ROS Turtlesim is used in this example to demonstrate publishing to the /turtle1/cmd_vel topic.

The following code demonstrates how to obtain a Java reference to a topic publisher, and publish a message:

```
// Get the ROSNode from RCSMManager.
ROSNode rosNode = (ROSNode)getController().getRCSM().getProviderByName("ros");
// Get the topic Publisher
CommunicationSenderInterface publisher=rosNode.getPublisher("/turtle1/cmd_vel");
// Define the message GAMEPAD_LEFT
static byte[] GAMEPAD_LEFT=Movements.getLeftMovement();
// Send message (Publish message) – This will spin the turtle to the left.
sender.send(GAMEPAD_LEFT);
```

```
// The Movements class contains pre-defined topic /turtle1/cmd_vel movement messages
class Movements
{
        public static byte[] getLeftMovement()
        {
            byte[] gamepad_left=new byte[52];
            gamepad_left[0] = (byte)(0x30);
            gamepad_left[1] = (byte)(0x00);
            gamepad_left[2] = (byte)(0x00);
            gamepad_left[3] = (byte)(0x00);
            // Append linear x movement event to byte[].
            gamepad_left[4] = (byte)(0x00);
            gamepad_left[5] = (byte)(0x00);
            gamepad_left[6] = (byte)(0x00);
            gamepad_left[7] = (byte)(0x00);
            gamepad_left[8] = (byte)(0x00);
            gamepad_left[9] = (byte)(0x00);
            gamepad_left[10] = (byte)(0x00);
            gamepad_left[11] = (byte)(0x00);
            // Append linear y movement event to byte[].
            gamepad_left[12] = (byte)(0x00);
            gamepad_left[13] = (byte)(0x00);
            gamepad_left[14] = (byte)(0x00);
            gamepad_left[15] = (byte)(0x00);
            gamepad_left[16] = (byte)(0x00);
            gamepad_left[17] = (byte)(0x00);
            gamepad_left[18] = (byte)(0x00);
```

```
 gamepad_left[19] = (byte)(0x00);
 // Append linear theta z to byte[]
 gamepad_left[20] = (byte)(0x00);
 gamepad_left[21] = (byte)(0x00);
 gamepad_left[22] = (byte)(0x00);
 gamepad_left[23] = (byte)(0x00);
 gamepad_left[24] = (byte)(0x00);
 gamepad_left[25] = (byte)(0x00);
 gamepad_left[26] = (byte)(0x00);
 gamepad_left[27] = (byte)(0x00);

// Append angular x movement event to byte[].
 gamepad_left[28] = (byte)(0x00);
 gamepad_left[29] = (byte)(0x00);
 gamepad_left[30] = (byte)(0x00);
 gamepad_left[31] = (byte)(0x00);
 gamepad_left[32] = (byte)(0x00);
 gamepad_left[33] = (byte)(0x00);
 gamepad_left[34] = (byte)(0x00);
 gamepad_left[35] = (byte)(0x00);
 // Append angular y movement event to byte[].
 gamepad_left[36] = (byte)(0x00);
 gamepad_left[37] = (byte)(0x00);
 gamepad_left[38] = (byte)(0x00);
 gamepad_left[39] = (byte)(0x00);
 gamepad_left[40] = (byte)(0x00);
 gamepad_left[41] = (byte)(0x00);
 gamepad_left[42] = (byte)(0x00);
```

```
            gamepad_left[43] = (byte)(0x00);

            // Append angular theta z to byte[]

            gamepad_left[44] = (byte)(0x00);

            gamepad_left[45] = (byte)(0x00);

            gamepad_left[46] = (byte)(0x00);

            gamepad_left[47] = (byte)(0x00);

            gamepad_left[48] = (byte)(0x00);

            gamepad_left[49] = (byte)(0x00);

            gamepad_left[50] = (byte)(0x00);

            gamepad_left[51] = (byte)(0x40);

            return gamepad_left;

        }

    }
```