# Turtlesim Matlab

## Package Summary

`turtlesim_matlab` contains Matlab code that imitates the "turtlesim" ROS package used to teach basic ROS principles.

### Dependencies

In order to run this code the ROS Toolbox is required. See the file "installing_ROSToolbox" for installing this toolbox. Matlab should request the installation of this toolbox when attempting to run the code without it.

### Nodes in Matlab

It is important to know how Matlab works with ROS Nodes. When a script creates a publisher or subscriber it continues running in the background until "rosshutdown" is called. So a "node" is a little more loosely defined. Nodes can be created as objects in a Matlab script or in the command window rather than just being the script. This package follows the standard node style as if writing nodes with C++ or Python. One script is equivalent to one node. If the node script contains a continuous "while" loop then the script will never end, so a second node cannot be run from the same Matlab window. A second instance of Matlab needs to be run in order to run an additional node. If running a second instance is not feasible, then all of the code must be written in a single script and all publishers and subscribers will be created in a single node.

## Getting Started

Start by running the main node to make sure the system works and the ROS Toolbox is installed. To do this, navigate to the folder containing the "turtlesim.m" file using the "Current Folder" window in Matlab or double-click the file in a file explorer and hit "Run" in the Editor once Matlab opens the file. Then select "Change Folder" if the option is given. The output should look like the following:

```
================================================================================
>> turtlesim
Initializing ROS master on …
Initializing global node … with NodeURI …
================================================================================
```

To check that the node is working correctly, print out the topics that are currently published and subscribed to. In order to perform this step a new instance of Matlab should be opened. Right-click on the Matlab icon and select "Open Additional Instance of Matlab". Once the new window has opened, that window needs to be connected to the ROS network. In the command window type: rosinit. Once connected to the ROS network, type the command: rostopic list.

```
=====================================================================
>> rosinit
Initializing global node … with NodeURI …
>> rostopic list
/rosout
/turtle/cmd_vel
/turtle/pose
=====================================================================
```

The topics should be listed in alphabetical order. To get the connections for a single topic type the command: rostopic info [topic_name].

```
=====================================================================
>> rostopic info /turtle/pose
Type: turtlesim/Pose

Publishers:
* /matlab_global_node_77871 ([node uri])

Subscribers:
>> rostopic info /turtle/cmd_vel
Type: geometry_msgs/Twist

Publishers:

Subscribers:
* /matlab_global_node_77871 ([node uri])
=====================================================================
```

Next write a quick publisher to test that the connection is working. Create a publisher for the command velocity (/turtle/cmd_vel). But before creating a publisher it is best to create the message that the publisher will be sending. The "/turtle/cmd_vel" topic uses a "geometry_msgs/Twist" type message. Create one using Matlab's "rosmessage('messagetype')" function. Replace "messagetype" with "geometry_msgs/Twist" and save the output of the function to a variable called "msg". Then create the publisher using Matlab's "rospublisher('topic_name', 'messagetype')" function. Replace "topic_name" with "/turtle/cmd_vel" and "messagetype" with "geometry_msgs/Twist" and save the output of the function to a variable called "pub". Set the value of the "Linear.X" variable to 20.0. Reference the member by using a period (the dot operator): msg.Linear.X = 20.0. Then send the message using the "send(msg)" function provided by the publisher object: pub.send(msg);.

```
=====================================================================
>> msg = rosmessage('geometry_msgs/Twist)
```

```
msg =

  ROS Twist message with properties:

    MessageType: 'geometry_msgs/Twist'
         Linear: [1×1 Vector3]
        Angular: [1×1 Vector3]

  Use showdetails to show the contents of the message

>> pub = rospublisher('/red_led', 'std_msgs/Bool')

pub =

  Publisher with properties:

          TopicName: '/turtle/cmd_vel'
         IsLatching: 1
    NumSubscribers: 0
       MessageType: 'geometry_msgs/Twist'

>> msg.Linear.X = 20.0;
>> pub.send(msg);
================================================================================
```

The robot will move forward a little bit indicating that it received the message. The command will only be followed for 1 second and then will stop. Now write a file called "control_node.m" that has a publisher to "/turtle/cmd_vel" and a subscriber to "/turtle/pose" that controls the robot.

## Nodes

### turtlesim.m

This node runs the GUI displaying the robot. The robot cannot leave the area and will follow the velocity commands received for 1 second. Continuously send a velocity command to keep it running. The 2D position and orientation are provided as feedback.

#### Published Topics

`/turtle/pose (turtlesim/Pose)`

Contains the current (x,y) position along with the orientation angle of the robot. Also provides the current linear and angular velocities.

#### Subscribed Topics

`/turtle/cmd_vel (geometry_msgs/Twist)`

The linear and angular command velocity of the robot. The robot will execute a

command velocity for 1 second and then time out. Twist.Linear.X is the forward velocity and Twist.Angular.Z is the angular velocity.

### control_node.m

This is the node written by the user to control the robot. An example is provided for reference.

#### Published Topics

`/turtle/cmd_vel (geometry_msgs/Twist)`

The linear and angular command velocity of the robot. The robot will execute a command velocity for 1 second and then time out. Twist.Linear.X is the forward velocity and Twist.Angular.Z is the angular velocity.

#### Subscribed Topics

`/turtle/pose (turtlesim/Pose)`

Contains the current (x,y) position along with the orientation angle of the robot. Also provides the current linear and angular velocities.