ROS Matlab Examples

Package Summary

`ros_matlab_examples` contains Matlab code for teaching ROS in Matlab.

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.

Section Descriptions

The "Getting Started" section explains how to run the code and starting up a second Matlab instance. The "Nodes" section explains the node scripts and their topics. The "Support Files" explains any files that are not used as nodes. Code examples are separated by lines of equal signs "=====". Any commands intended to be run in the command window will be preceded by the Matlab prompt ">>".

Folder Structure

ros_matlab_examples
|----control_node.m
|----example_ROS_LED_si.m
|----example_ROS_LED.m
|----installing_ROSToolbox.pdf
|----ROS Matlab Example-hw.pdf
|----ROS_examples.m

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 "example_ROS_LED.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:

```
================================================================
>> example_ROS_LED
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
/blue_led
/green_led
/keypress
/red_led
/rosout
================================================================
```

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 /blue_led
Type: std_msgs/Bool

Publishers:

Subscribers:
* /matlab_global_node_37895 ([node uri])
>> rostopic info /keypress
Type: std_msgs/Char

Publishers:
* /matlab_global_node_37895 ([node uri])

Subscribers:
================================================================
```

Next, write a quick publisher to test that the connection is working. Create a publisher for the red LED. But before creating a publisher it is best to create the message that the publisher will be sending. The LED topics use a “std_msgs/Bool” type message. Create one using Matlab’s “rosmessage(‘messagetype’)” function. Replace “messagetype” with “std_msgs/Bool” and save the output of the function to a variable called “msg”. Then create the publisher using Matlab’s “rospublisher(‘topicname’, ‘messagetype’)” function. Replace “topicname” with “/red_led” and “messagetype” with “std_msgs/Bool” and save the output of the function to a variable called “pub”. Set the value of the message to “true” by storing it in the “Data” member of “msg”. Reference the member by using a period: msg.Data = true. Then send the message using the “send(msg)” function provided by the publisher object: pub.send(msg);.

```
=================================================================
>> msg = rosmessage(‘std_msgs/Bool’)

msg =

  ROS Bool message with properties:

    MessageType: 'std_msgs/Bool'
           Data: 0

  Use showdetails to show the contents of the message

>> pub = rospublisher(‘/red_led’, ‘std_msgs/Bool’)

pub =

  Publisher with properties:

         TopicName: '/red_led'
        IsLatching: 1
    NumSubscribers: 0
       MessageType: 'std_msgs/Bool'

>> msg.Data = true;
>> pub.send(msg);
=================================================================
```

The red LED symbol should turn to a brighter red color. That means it has been turned on. Now fill in the “control_node.m” file with a subscriber and publishers to turn the LEDs on and off using the keyboard.

Nodes

example_ROS_LED.m

This node generates the GUI that contains the simulated LEDs. The LEDs should be controlled by completing the "control_node.m" file.

Published Topics

`/keypress (std_msgs/Char)`

The ASCII number of the key pressed by the keyboard. Note that 'r' = 114, 'g' = 103', and 'b' = 98.

Subscribed Topics

`/red_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the red LED.

`/green_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the green LED.

`/blue_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the blue LED.

control_node.m

The node that handles processing the keystrokes and turning on or off the LEDs.

Published Topics

`/red_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the red LED.

`/green_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the green LED.

`/blue_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the blue LED.

Subscribed Topics

`/keypress (std_msgs/Char)`

The ASCII number of the key pressed by the keyboard. Note that 'r' = 114, 'g' = 103', and 'b' = 98.

Published Topics

`/red_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the red LED.

`/green_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the green LED.

`/blue_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the blue LED.

Subscribed Topics

`/keypress (std_msgs/Char)`

The ASCII number of the key pressed by the keyboard. Note that 'r' = 114, 'g' = 103', and 'b' = 98.

example_ROS_LED_si.m

This node is a replication of the "example_ROS_LED.m" file but with a section that must be filled in with the contents of the "control_node.m" content. This node allows all of the operation to be performed in a single node in order to avoid opening an additional instance of Matlab.

Published Topics

`/keypress (std_msgs/Char)`

The ASCII number of the key pressed by the keyboard. Note that 'r' = 114, 'g' = 103', and 'b' = 98.

Subscribed Topics

`/red_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the red LED.

`/green_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the green LED.

`/blue_led (std_msgs/Bool)`

Reads whether to turn ON (true) or OFF (false) the blue LED.

Support Files

ROS_examples.m

This file contains examples for how to write ROS components using the Matlab syntax.