

TUTORIAL 2: THE ROBOT OPERATING SYSTEM: BASICS 2

THE F1/10 TEAM

INTRODUCTION

This article contains the transcript of the Video Tutorial 2: The Basics of ROS (Part 2). This tutorial is based on tutorial 12 covered by the official ROS tutorials[1]. The aim of this tutorial is to familiarize you with the framework of the Robot Operating System by implementing your own simple publisher and subscriber nodes in python.

TRANSCRIPT OF VIDEO TUTORIAL

In this tutorial lets write our own publisher node that publishes a string hello world and a subscriber node that receives the string. We begin by creating a workspace. In your root folder, execute these commands:

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Catkin[2] is a ROS tool for compiling the different subsystems and modules used in ROS. Its like CMake, but over a larger scale and its specially designed keeping in mind some ROS requirements. Perform a `catkin_make` from the root of the workspace.

```
$ cd ..
$ catkin_make
```

The `devel` and `build` folders at the root of the workspace are where the linked libraries and the compiled code in machine language resides. Source the environment using

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

We will now create a package. In the `src` folder,type in the command

```
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

The `beginner_tutorials` is the name of the package and the `std_msgs`, `rospy`, `roscpp` are dependencies that we are defining. Use `catkin_make` from the root of the workspace. Source and confirm if your workspace is indeed added to your current environment by doing

```
$ echo ROS_PACKAGE_PATH
```

We will now use the `roscd` command to access a package within a workspace. If the environment containing the package is sourced, we should be able to naviagte to the pacakge by using `roscd` package name. here we can verify this by using

```
$ roscd beginner tutorials
```

We will create a new folder to hold all our code

```
$ mkdir scripts
$ cd scripts
```

Place the talker.py and listener.py files in the script folder just created.

Lets walk through the talker.py file first

- ‘from std_msgs.msg import String’ line imports the string data type from the std_msgs library that we included during the creation of beginner_tutorials package
- The rospy.Publisher function creates a new topic called chatter of type string
- rospy.init_node initializes the node called talker, and
- pub.publish(hello_str) publishes the string containing hello world with the system time.

The listener.py consists of a similar structure.

- Like in the talker node rospy.init_node initializes the listener node.
- But this node subscribes to the chatter topic published by the talker node using the rospy.Subscriber function
- The callback function is an interrupt function that is called every time some data is published to the chatter topic. Here we just print back the string that it received.

Before we can run the nodes we need to give executable permissions to these files using

```
$ sudo chmod +x talker.py listener.py
```

To build the nodes navigate to the root of the workspace and run

```
$ cd ~/catkin_ws
$ catkin_make
```

make sure the process ends without errors

Start roscore,

```
$ roscore
```

then in new terminals run the talker.py and listener.py nodes. Make sure you source the workspace using

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

every time you open a new terminal.

Run the nodes using


```
$ rosrn beginner_tutorials listener.py
```

and in a seperate terminal

```
$ source ~/catkin_ws/devel/setup.bash
$ rosrn beginner_tutorials talker.py
```

Go ahead and run all the `roscd`, `rostopic`, `rosmmsg` and `rqt_graph` commands to analyze the program. In the next tutorial you will learn how to configure and run ROS on the jetson.

REFERENCES

- [1] [ROS Beginner Tutorial 12](#) 
- [2] [What exactly is catkin](#) 