Create a ROS python project

OVERVIEW

Level: *Basic*
Goal: *learn how to structure your python code using catkin package manager*
Requirements:

- Quick start with coding on QTrobot
- Basic understanding of ROS framework

This tutorial guides you towards creating and structuring your python code for QTrobot. You will learn, how to create a ROS package for your python code, how to build it and how to run your code using `rosrun` and `roslaunch`. We will also show you how to configure your code using ROS parameters and launch file.

What is catkin?

Catkin is the official build system of ROS. In some simple words, catkin helps you to better structure you code, its dependencies within an isolated workspace known as catkin workspace (e.g. `~/catkin_ws` folder). Catkin offers different commands to create a ROS package, build and manage its dependencies. You can **read more about catkin** on official ROS website.

Structure of catkin workspace folder

Catkin folder usually consists of `src`, `devel` and `build` folder. The `src` folder is the place where the your codes are hosted. the `devel` and `build` folder are automatically created and updated by catkin commands to configure and build your codes.

```
~/catkin_ws
    - src
    - devel
    - build
```

REMINDER

- Ensure that you have already read carefully the **`Quick start guide`** for coding and have your setup ready to follow this tutorial on QTPC.
- Both QTRP and QTPC has its own catkin folder which are located in the home folder: `~/catkin_ws`! We are running this tutorial (and the most following tutorials) on the QTPC.

Our first python project

In the following steps, we will create a package called `my_tutorial` with the following information. Then step by step we fill it with our python code.

- *package name*: `my_tutorial`

- *package dependencies*: `std_msgs, rospy, and` roscpp`
- *package description*: `My first python tutorial`
- *package version*: `1.0.0`
- *package author name*: `my name`

Create project's structure

Open a terminal on QTPC, switch to `~/catkin_ws/src` folder and run the following commands:

```
cd ~/catkin_ws/src
catkin_create_pkg my_tutorial std_msgs rospy roscpp -D "My first python
tutorial" -V "1.0.0" -a "my name"
```

That creates a new package called `my_tutorial` in the `src` folder of our catkin workspace on QTPC:

```
Created file my_tutorial/package.xml
Created file my_tutorial/CMakeLists.txt
Created folder my_tutorial/include/my_tutorial
Created folder my_tutorial/src
Successfully created files in /home/qtrobot/catkin_ws/src/my_tutorial. Please
adjust the values in package.xml.
```

You can open open the `package.xml` file and update its content such as description, version, authors name, emails etc.

Create our main python file

The above command only creates an empty structure for our project. We need to add our python files by ourselves as follows:

```
cd my_tutorial/src
touch my_tutorial_node.py
chmod +x my_tutorial_node.py
```

first we created `my_tutorial_node.py` and then made it executable using `chmod +x` permission setting command.

open the `my_tutorial_node.py` file using your favorite editor[1] and add the following content to it:

```
#!/usr/bin/env python
import sys
import rospy

if __name__ == '__main__':
    rospy.init_node('my_tutorial_node')
    rospy.loginfo("my_tutorial_node started!")

    try:
```

```
        rospy.spin()
    except KeyboardInterrupt:
        pass

    rospy.loginfo("finsihed!")
```

The above code simply starts a ROS node called *my_tutorial_node* does nothing until we press `<CTRL+C>` to stop it.

Build and configure our python project

Before adding an actual action to our python code, let's just first build and run it and check if everything is fine. To build a catkin package, we need to run `catkin_make` command within catkin workspace.

NOTE

The `catkin_make` command must be run within the top most level of catkin workspace, that is in `~/catkin_ws` and **NOT** in `~/catkin_ws/src`.

```
cd ~/catkin_ws
catkin_make
```

the above command configure and builds our *my_tutorial* package along with other packages in catkin source directory.

TIP

To build a single package (e.g. *my_tutorial*) from source directory, you can call `catkin_make --pkg my_tutorial`!

Run and check our python project

There are different way to run a python project in ROS. You can simply run the python file from command line like any other python scripts. However, the most standard way of running a python project in ROS is to use `rosrun` or `roslaunch` command. Let's start with `rosrun` command. The interesting fact of these two commands is that you do not need to be in the same folder or use the absolute path to run your ROS python script. You can run it from anywhere on QTPC as follows:

```
rosrun my_tutorial my_tutorial_node.py
```

The first parameter of *rosrun* command is the name of our package and the second on is the name of our python script file. That should creates a new ROS node called `my_tutorial_node` in the ROS network. we can see it using `rosnode list` command from another terminal:

```
/controller
/my_tutorial_node       <---
/qt_motor
```

```
/qt_nuitrack_app
/qt_robot_interface
/qt_setting_interface
/qt_vosk_app
/rosapi
/rosauth
/rosbridge_websocket
/rosout
```

To stop the code, just press `<CTRL+C>` in the same terminal where we run our code.

Adding parameters to our python project

ROS framework offers parameter Server where strings, integers, floats, Booleans, lists and etc. can be stored as key-value objects. The parameter server can be accessed via command line and within all ROS supported programing languages such as our python code. We can also sets the parameters within our code. Lets first modify our python code and add a line to read a parameters called `param1`

```
param1 = rospy.get_param('~param1', 'default_value')
rospy.loginfo("value of param1 is %s", param1)
```

The above code looks for `param1` key within our node namespace. Notice that, we already specified the node namespace name in our python code when we were initializing it: `rospy.init_node('my_tutorial_node')`

Since we have not set any value for `param1` in parameter server yet, the above code should print the `default_value`! Now let's set a value for `param1` using `rosparam` command from command line. open a terminal and type:

```
 rosparam set /my_tutorial_node/param1 "QT"
```

This set a `param1` key specifically for our `my_tutorial_node`. Now if we re-run our python code, it will get the value (i.e. "QT") of the `param1` and print it out. You can **read more about rosparam in python** on official ROS website.

Preparing ROS launch file

Launch files are very common in ROS. They provide a convenient way to start up multiple nodes, setting up parameters for nodes and etc. Let's create our first launch file. First we need to create the `launch` folder within our python project folder. Catkin does not do this automatically. Then we need to create a launch file:

```
cd ~/catkin_ws/src/my_tutorial
mkdir launch
cd launch
touch my_tutorial.launch
```

now open `my_tutorial.launch` and add the following content:

```
<launch>
  <node name="my_tutorial" pkg="my_tutorial" type="my_tutorial_node.py"
output="screen">
  </node>
</launch>
```

This is the very basic setting of a ROS launch file. we just indicated that we would like to run one node which is located in `my_tutorial` package. Within this package, we want to run our python script which is given as `type="my_tutorial_node.py"`. The optional output parameter indicate where the python log output should be printed.

Running our python project using launch file

Now we have our launch file ready we can launch it from everywhere on QTPC using the `roslaunch` command:

```
roslaunch my_tutorial my_tutorial.launch
```

Setting parameters using launch file

Modify `my_tutorial.launch` file as follow to set a value for `param1` key.

```
<launch>
  <node name="my_tutorial" pkg="my_tutorial" type="my_tutorial_node.py"
output="screen">
    <param name="param1" value="QTrobot :)" />
  </node>
</launch>
```

CONGRATULATIONS! 😉

You have just created your first python project in ROS. In the next tutorial, you will learn how to extend this tutorial to **`Command QTrobot via ROS Publishers`**.

---

1. you can simply use Ubuntu gedit text editor or install Atom or Visual Studio Code whichever comes as your favorite.↩