ROS

ROBOTICS CLUB

SCIENCE AND TECHNOLOGY COUNCIL
IIT KANPUR

Contents

- What is ROS?
- Why ROS?
- Some applications
- ROS from the Terminal
- Launch files

Robotics Club IITKanpur

What is ROS?

ROS is an open-source, meta-operating system for your robot. It is a medium of communication between the hardware and software of the robot.

Why ROS?

- ROS is an open source environment, it has brought all of robotics under one roof.
- It saves us the trouble of writing standard pieces of code like path planning and localization algorithms as it makes them available in user-editable packages.
- It makes working with sensors, their data and its integration much simpler through messages and topics.

Robotics Club IITKanpur

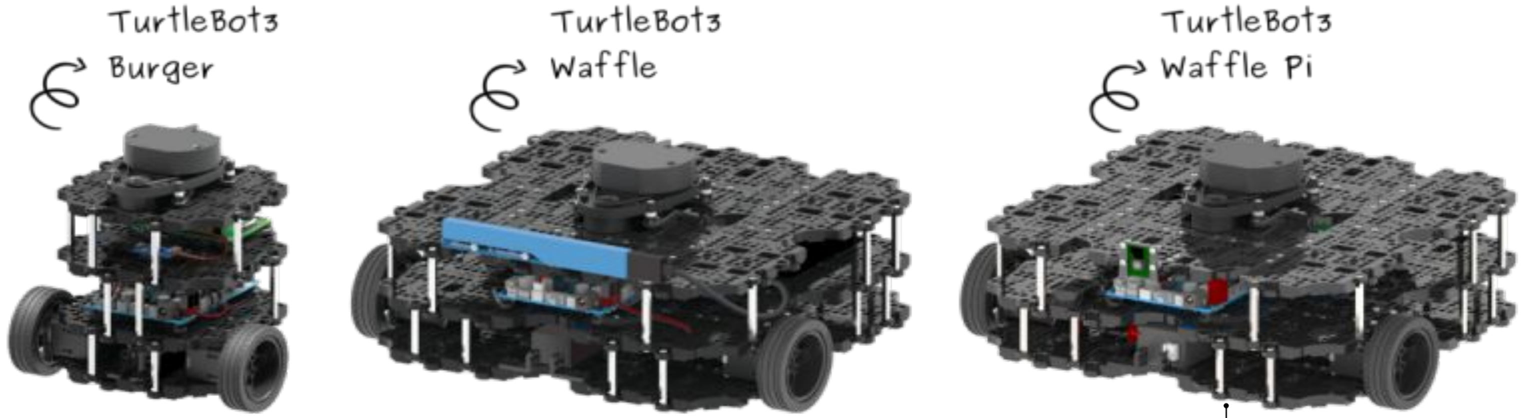
Why ROS?

- Standard ROS programmable robots are available with working simulations.
  For example, Turtlebot 3, Pioneer 3dx.
- It provides us with robust simulation environment called Gazebo and many other functionalities like Rviz, some of which we will discuss later.

Robotics Club IITKanpur

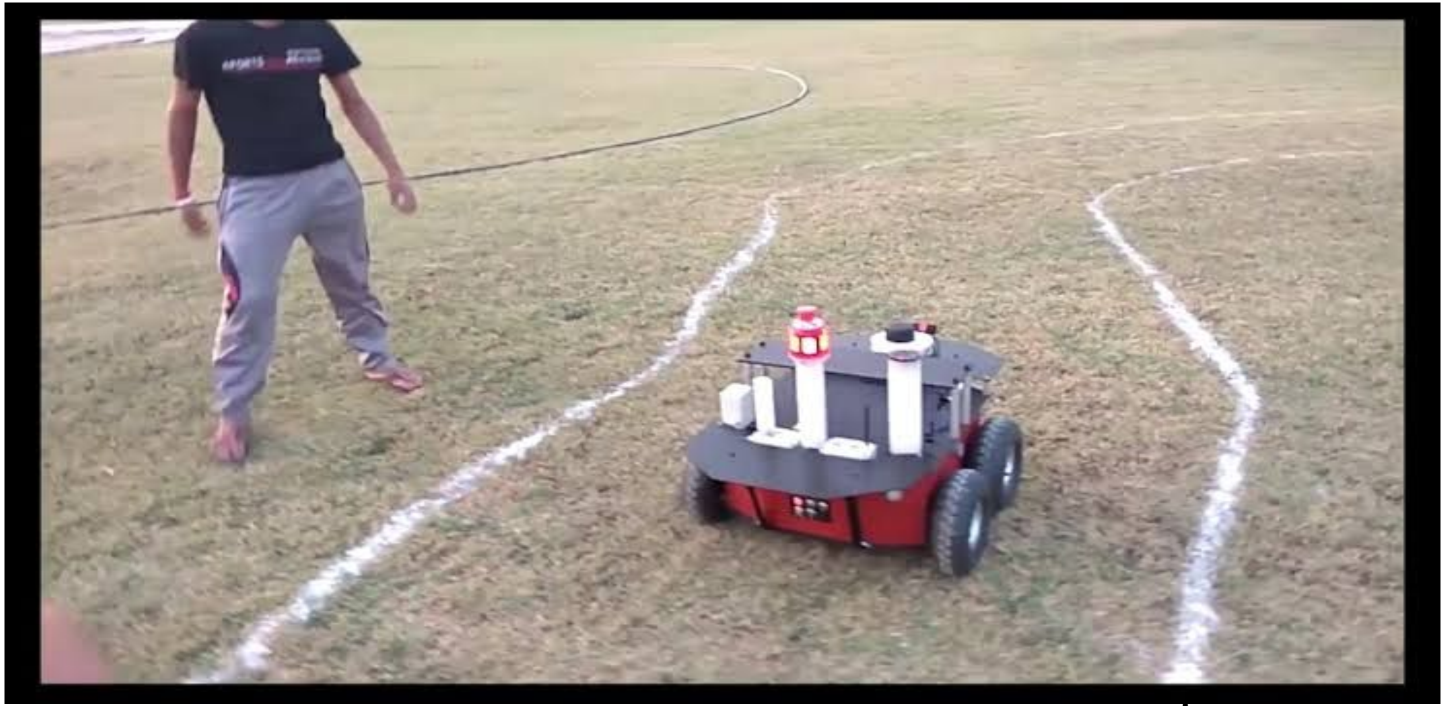
Turtlebot

TURTLEBOT3



Robotics Club IITKanpur

Pioneer 3dx (Team IGVC)



Robotics Club IITKanpur

Gazebo

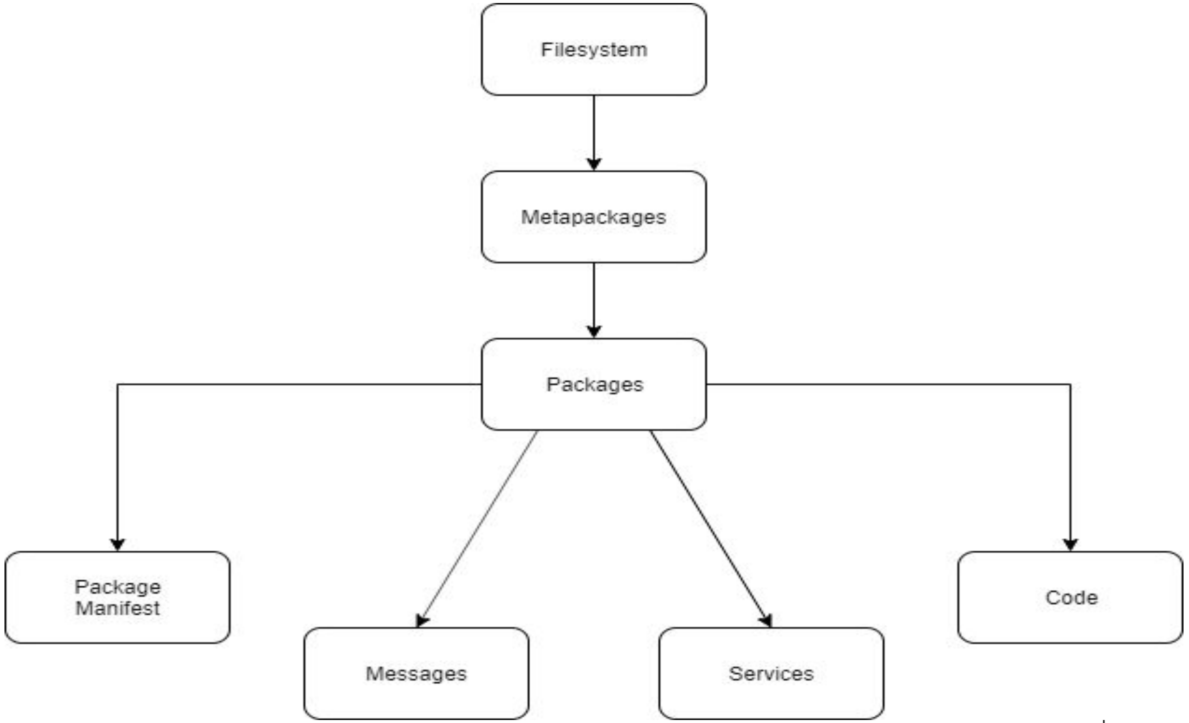
Gazebo is simulation environment that works hand in hand with ROS.

- It allows to import custom CAD models.
- It allows to define the physics and dynamics of our environment.
- It allows us to test our code on a real world like environment.

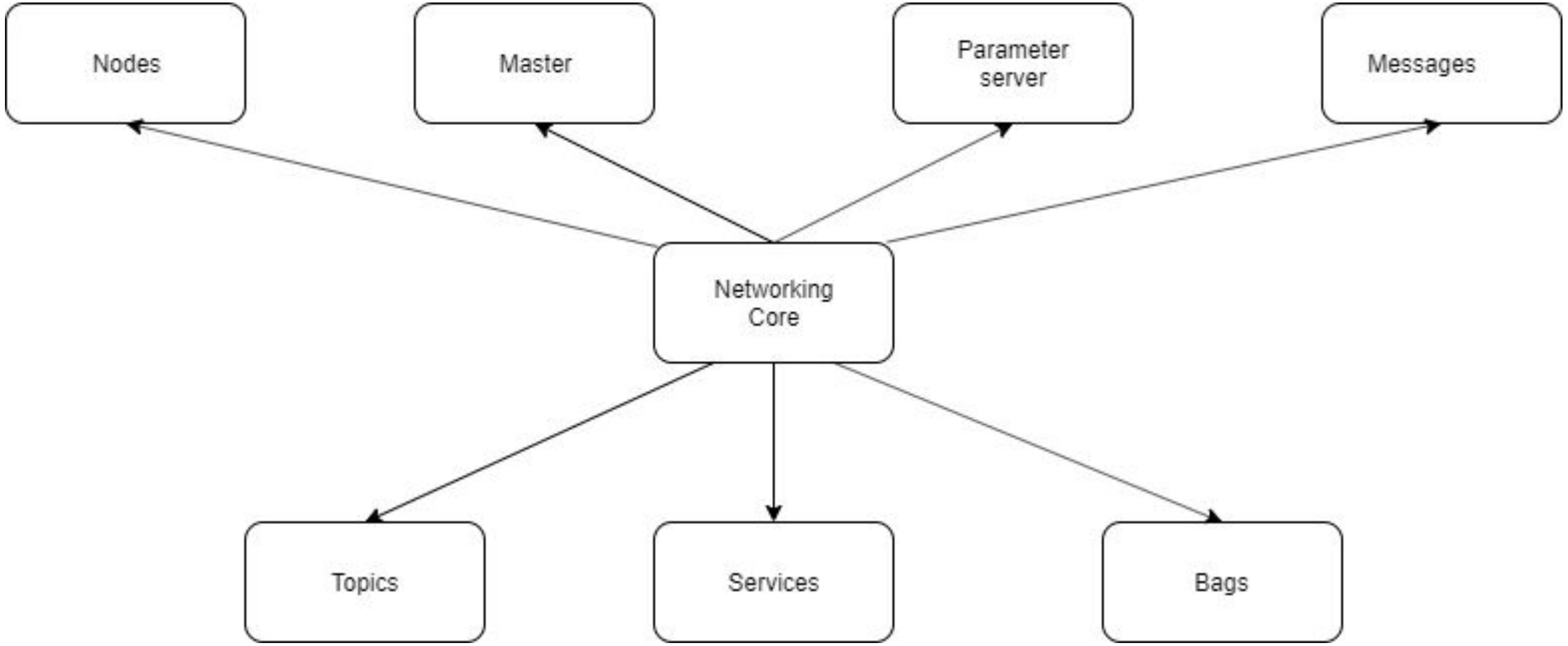
Turtlebot modeled in Gazebo

Robotics Club IITKanpur

Understanding the ROS filesystem



Understanding the ROS Network



Robotics Club IITKanpur

Understanding the ROS Network

- Nodes are processes where computations happen, many nodes connected to the ROS network can interact with each other.
- Master sets up communication between nodes, messages and services. It registers the names of for components of our system.
- Parameter server stores our parameters.
- Messages are used by nodes to communicate with each other.

Understanding the ROS Network

- Topics : Messages are routed to the ROS network and hence other nodes on particular topics.
- Services are used to request or get an answer from a node.
- Bags are a format to save and play back ROS message data.

Robotics Club IITKanpur

Catkin

- Catkin is a build system used with ROS
- Build system - Automatically creates executables from your code
- Runs using g++, Python and CMake - more on this later
- Needs properly structured workspaces to function

Catkin Workspaces

Every catkin workspace has five main folders:

- **build** - this is where catkin works to make the executables
- **devel** - everything needed to develop the executable is kept here
- **src** - all the source code goes here - this is where you will work
- **logs** - to store error logs
- **install** - for testing the built executables - (out of scope for us)

Never edit any files in the build and devel folders.

Robotics Club IITKanpur

Creating Catkin Workspaces

- Can convert any directory to a catkin workspace by:

```
catkin init
```

- Provided there is an src subfolder - rest directories automatically made
- To check the status of a catkin workspace, use:

```
catkin config
```

- To build a ROS package using catkin,

```
catkin build [package_name]
```

- To clean your workspace i.e delete build and devel use

```
catkin clean
```

Robotics Club IITKanpur

Git - A Brief Introduction

- git is a version control system
- It allows you to keep track of each version of your code.
- GitHub is a website that maintains your code using git
- GitHub is the hub of open-source development
- ROS is also open-source - so you can find a lot of ROS packages on GitHub
- Downloading a Git repository is termed cloning

Install git: sudo apt-get install git
To clone a repository: git clone [repository_url]

Robotics Club IITKanpur

Bash Cheatsheet

- cd [DIR_PATH]
- ls [DIR_PATH]
- mkdir [DIR_NAME]
- touch [FILE_NAME]
- pwd
- ~
- ..
- .
- /

- export [VAR_NAME]=[VALUE]
- source [SCRIPT_NAME]
- rm [FILE_PATH]
- rmdir [DIR_PATH]
- mv [SRC_PATH] [DST_PATH]
- cp [SRC_PATH] [DST_PATH]

Time for a tutorial!

- First let's create a catkin workspace in the home directory(~)

```
mkdir -p ~/tutorial_ws/src
cd ~/tutorial_ws
catkin init
```

- Now, let's clone a repository **into the src folder**

```
cd ~/tutorial_ws/src
git clone https://github.com/ashwin2802/ROS-WC19
```

- Next let's build the package

```
catkin build turtlesim
```

Robotics Club IITKanpur

Few Things to note

- catkin will automatically search for the package in all subdirectories of the src folder
  - We had turtlesim at src/ROS-WC19/turtlesim

- You cannot have two packages of the same name.
  - catkin searches by package name, this will confuse it

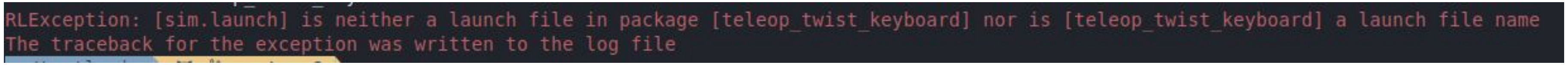
- You can execute catkin build from any subdirectory of the workspace
  - As long as the package you want to build is inside the src folder

Robotics Club IITKanpur

A Very Important Command

source [WS_PATH]/devel/setup.bash

- Basically tells the system about the executables you just built
- If you miss this, you will get errors like these:



```
RLException: [sim.launch] is neither a launch file in package [teleop_twist_keyboard] nor is [teleop_twist_keyboard] a launch file name
The traceback for the exception was written to the log file
```

- Has to be done every time you open a new terminal
- **Hax:** Put the command in your **~/.bashrc** file
  - All commands in the ~/.bashrc file are run when a new terminal is created

Tutorial 1

In three separate terminals:

Starts up ROS

```
roscore
```

```
rosrun turtlesim turtlesim_node
```

```
rosrun turtlesim turtle_teleop_key
```

Starts up ROS nodes

General Syntax: ros[cmd] [package_name] [file_name]

Robotics Club IITKanpur

ROS Terminal Commands

- rosnode: list, info, kill, ping
- rostopic: list, info, type, find, echo, pub, hz
- rosservice: list, info, type, find, call
- rosparam: list, get, set
- rosmsg: show
- rossrv: show

When in doubt, press TAB

Robotics Club IITKanpur

Some more ROS commands

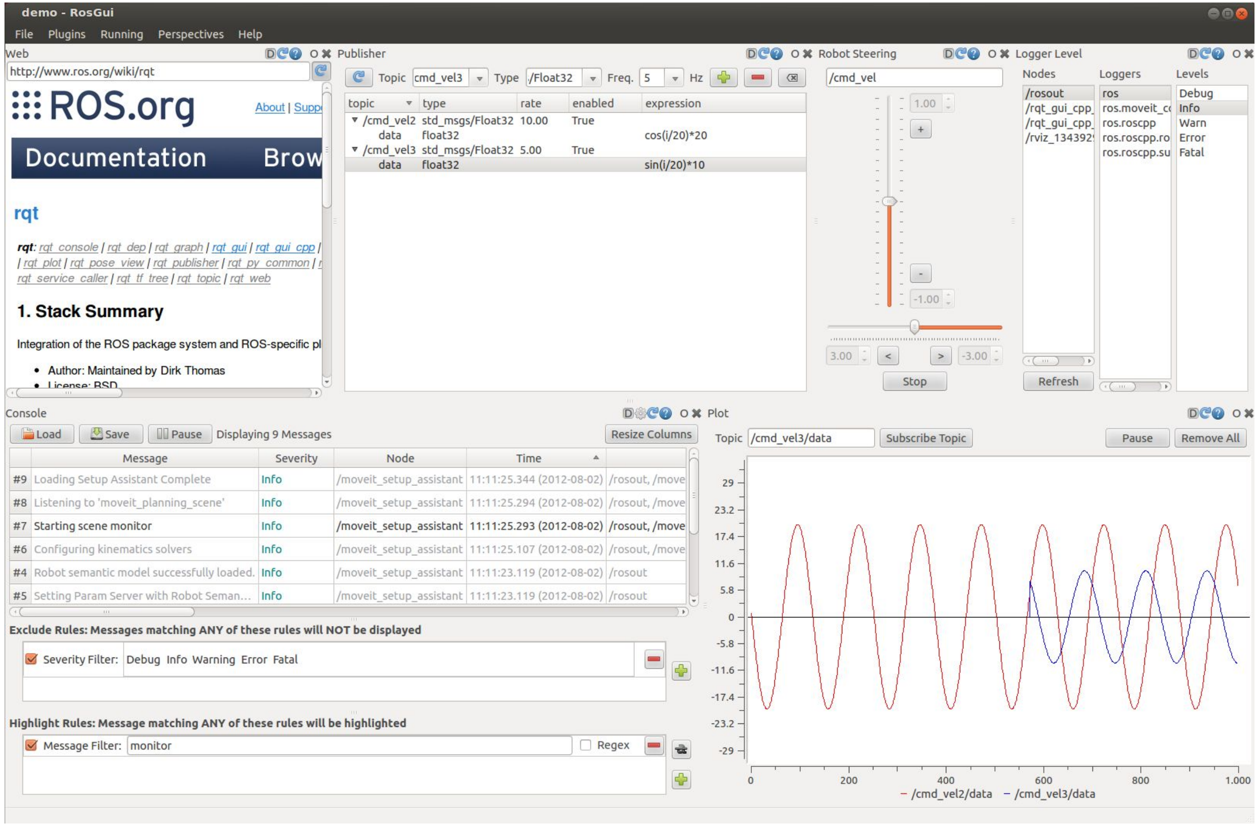
Here's a cheat sheet for important commands you will regularly use.

- roscd : used for switching between various ROS package directories.
- rospack find [package_name] : gives exact path to a ROS package.
- rosls : allows you to ls directly in a ROS package.
- rosed : allows you to edit any file in any ROS package.
- roscore : starts up ros
- rosrun : to run a node.

Don't forget to press TAB

Robotics Club IITKanpur

RQt



demo - RosGui

File Plugins Running Perspectives Help

Web

http://www.ros.org/wiki/rqt

ROS.org About | Supp

Documentation Brow

rqt

rqt: rqt_console | rqt_dep | rqt_graph | rqt_gui | rqt_gui_cpp | | rqt_plot | rqt_pose_view | rqt_publisher | rqt_py_common | rqt_service_caller | rqt_tf_tree | rqt_topic | rqt_web

1. Stack Summary

Integration of the ROS package system and ROS-specific pl

- Author: Maintained by Dirk Thomas
- License: BSD

Publisher

Topic: cmd_vel3 Type: /Float32 Freq.: 5 Hz

topic	type	rate	enabled	expression
▼ /cmd_vel2	std_msgs/Float32	10.00	True	
data	float32			cos(i/20)*20
▼ /cmd_vel3	std_msgs/Float32	5.00	True	
data	float32			sin(i/20)*10

Robot Steering

/cmd_vel

1.00 + - -1.00

3.00 < > -3.00

Stop

Logger Level

Nodes	Loggers	Levels
/rosout	ros	Debug
/rqt_gui_cpp_	ros.moveit_c	Info
/rqt_gui_cpp_	ros.roscpp	Warn
/rviz_134392	ros.roscpp.ro	Error
	ros.roscpp.su	Fatal

Refresh

Console

Load Save Pause Displaying 9 Messages Resize Columns

	Message	Severity	Node	Time	
#9	Loading Setup Assistant Complete	Info	/moveit_setup_assistant	11:11:25.344 (2012-08-02)	/rosout, /move
#8	Listening to 'moveit_planning_scene'	Info	/moveit_setup_assistant	11:11:25.294 (2012-08-02)	/rosout, /move
#7	Starting scene monitor	Info	/moveit_setup_assistant	11:11:25.293 (2012-08-02)	/rosout, /move
#6	Configuring kinematics solvers	Info	/moveit_setup_assistant	11:11:25.107 (2012-08-02)	/rosout, /move
#4	Robot semantic model successfully loaded.	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout
#5	Setting Param Server with Robot Seman…	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout

Exclude Rules: Messages matching ANY of these rules will NOT be displayed

☑ Severity Filter: Debug Info Warning Error Fatal

Highlight Rules: Message matching ANY of these rules will be highlighted

☑ Message Filter: monitor ☐ Regex

Plot

Topic /cmd_vel3/data Subscribe Topic Pause Remove All

- GUI for ROS
- Has many useful plugins
- We will look at some of them

Robotics Club IITKanpur

RQt Plugins

General Syntax: rosrun rqt_[plugin_name] rqt_[plugin_name]

- rqt_graph: Shows you the whole ROS network
  - **most useful** for debugging errors
- rqt_plot: used to plot data published on topics
- rqt_top: shows amount of resources used by each ROS node
- rqt_image_view: used to view images published on topics
- rqt_publisher: used to publish messages to topics
- rqt_service_caller: used to call services
- rqt_gui: can display multiple plugins

Robotics Club IITKanpur

Launch files

- Used to launch multiple nodes in one single go
- No need to launch roscore too - it does it automatically
  - No need to open multiple terminals
  - But also - all the nodes will write messages to same terminal
- All launch files go in **launch** directory - inside your package
- Launch files must end in **.launch**
- Written in XML (e**X**tensible **M**arkup **L**anguage)
  - Just like HTML - tags, nesting, attributes etc.

Robotics Club IITKanpur

Writing a launch file

```
cd turtlesim
mkdir launch
cd launch
```

- Create the launch directory in the turtlesim package
- Create a launch file **main.launch** and open it
- First line: `<?xml version="1.0" encoding="UTF-8"?>` specifies that file is in XML
- Second line: `<launch>` specifies that file is a launch file
- Last line: `</launch>` ends the launch tag
- Add node using:

```
<node pkg="[PACKAGE_NAME]" type="[EXECUTABLE_NAME]" name="[NODE_NAME]" output="screen"/>
```

Notice / at the end - This is a more compact form
Can be used since there are no tags to be nested inside node **yet**

Robotics Club IITKanpur

Sample launch file

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
   <include file="$(find mavros)/launch/px4.launch" />
   <node pkg="inter_iit_uav_fleet" type="planner" name="planner" output="screen"/>
   <node pkg="inter_iit_uav_fleet" type="router" name="router"/>
   <node pkg="inter_iit_uav_fleet" type="detector" name="detector">
       <remap from="image" to="camera/image_raw"/>
   </node>
</launch>
```

Two new tags here

Had to close with </node> here

include tag: used to include other launch files in current launch file

remap tag: used to change name of topics just before launch

- will be useful in debugging - fixing wrong connections

Robotics Club IITKanpur

Rosbags

- Used to store message data from topics
- We can store a bag and then play it later on any other system
- Very useful for in-depth analysis in case of any failures
- To record a bag,

```
rosbag record -o [BAG_NAME] [TOPIC_NAME1] [TOPIC_NAME2] [...]
```

- To play a rosbag

This will play the bag on loop till you stop it using **Ctrl+C**

```
rosbag play [BAG_NAME].bag --loop
```

- When you play a bag, no nodes are created, only the messages are published

Robotics Club IITKanpur

- Remember, ROS will give you a tough time at the start but the documentation will be a life saver, learn to navigate it well and also learn to not get intimidated by it.
- You do not need to memorize much in ROS, feel free to use the documentation whenever you get stuck. This is something we still do to this day!
- Watching someone do it is way different than doing it yourself. So do try your best at the assignments.

Contact us if you have any problem/suggestion:

Abhay Varshney	**9559015388**
Madhur Deep Jain	**8894051687**
Neil Shirude	**9850892135**

roboticsclubiitkanpur@gmail.com

http://students.iitk.ac.in/roboclub/

https://www.facebook.com/roboclubiitkanpur

https://www.youtube.com/c/RoboticsClubIITKanpur

Robotics Club IITKanpur