ROS

roscd ⇒ changes the directory to catkin_ws workspace.

ros core ⇒ Runs the ROS Master node

rosnode list ⇒ list all the ROS Node

rostopic list ⇒ " " " topics

rosrun Packagename {eg-turtlesim} {To run a ros node} tab 2x

example rosrun turtlesim turtlesim_node

rostopic echo topicname {echo's what is being published in the topic}

Introduction to ROS and its Package Management {Mastering ROS for Robotics Programming}

⇒ The basic building blocks of the ROS Software framework are ROS packages.

↳ Create a Wiki page for our package on the ROS website to contribute to the ROS community.

★ Why should we learn ROS?

Robot application development platform that provides various features such as:-

- Message passing
- Distributed computing
- Code reusing etc...

⇒ ROS Comes with ready to use Capabilities, for example, SLAM & AMCL.

{Simultaneous Localization and Mapping}

{Adaptive Monte Carlo Localization}

MoveIt → {Package for motion planning of robot manipulator}

⇒ ROS is packed with tons of tools for debugging (rqt-gui), Visulizing (RViz) and Performing Simulation (Gazebo).

⇒ ROS is packed with device drivers and interface Packages for various sensors & actuators in Robotics.

⇒ The ROS message-passing middleware allows Communicating between different nodes. These nodes can be programmed in any language that has Ros client libraries. (C, C++, Python, Java)

⇒ ROS can easily handle Concurrent resource.

⇒ ROS has Active Community.

★ **Reasons for not Prefering ROS for Robots**

⇒ It is difficult to learn ROS.

⇒ To get started with Gazebo is not an easy task.

⇒ Difficulties in robot modeling.

↳ Robot modeling in ROS is performed using URDF, which is an XML based robot description.

↳ There is a SolidWorks plugin to convert a 3D model from SolidWorks to URDF.

⇒ We always need a computer to run ROS.

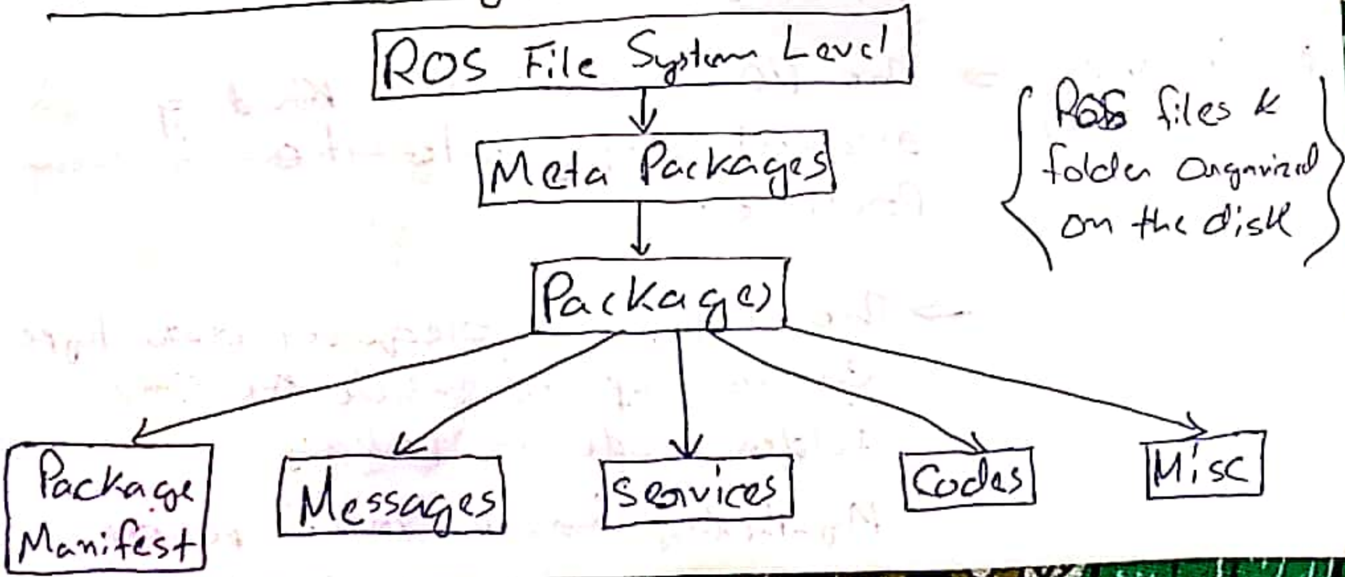
↳ Small robots that work completely on microcontroller don't require a ROS system.

↳ ROS is only required when we want to perform high-level functionalities such as autonomous navigation and motion planning.

⇒ When we deploy ROS on a commercial product, a lot of things need to be taken care of. One thing is code quality.

↳ Code quality is a loose approximation of how long-term useful and long-term maintainable the code is:

★ Understanding the ROS file System level



Packages ⇒ It contains ROS runtime process (nodes), libraries, configuration files & so on. which are organized together as a single unit.

Package manifest ⇒ It is inside a pakage that contains information about the Package, auther, licence, dependencies, compilation flags and so on.

(Package.xml file)

Meta package ⇒ Term meta package is used for a group of package for a Special purpose.

Meta Package manifest ⇒ Similar to package manifest, difference are that it might include Packages inside it as runtime dependencies and declare an export tag.

Messages (.msg) ⇒ The ROS messages are a type of Information that is Sent form one ROS process to the other.

Services (.srv) ⇒ The ROS service is a Kind of request/reply interaction between Processes.

→ The reply and request data type can be defined inside the srv folder inside the pakag.

My_Package/srv/MyServiceType.srv

Repositories ⇒ Most of the ROS packages are maintained using a Version Control System (VCS) such as Git.

⇒ The package in the repositories can be released using a Catkin release automation tool called bloom.

★ ROS packages

Commands to Create, modify and work with the ROS packages.

Catkin_Create_pkg ⇒ This command will Create new pakage.

ros pack ⇒ This command is used to get information about the package in the file System.

Catkin_make ⇒ This command is used to build the Package in the workspace.

ros dep ⇒ This Command will install the System dependencies required for this pakage.

To work with pakages, ROS provides a bash-like command called rosbash, which is used to navigate and manipulate the ROS package. Some of the rosbash Commands:

roscd ⇒ This Command is used to change the Package folder.

roscp ⇒ This command is used to Copy a file from a package.

rosed ⇒ This Command is used to edit a file.

rosrun ⇒ This Command is used to run an executable inside a package.

* ROS Meta packages

⇒ Specialized package in ROS that only Contain one file, that is a package.xml file.

⇒ Meta packages simply group a set of multiple Packages as a single logical package.

★ Understanding the ROS Computation graph level

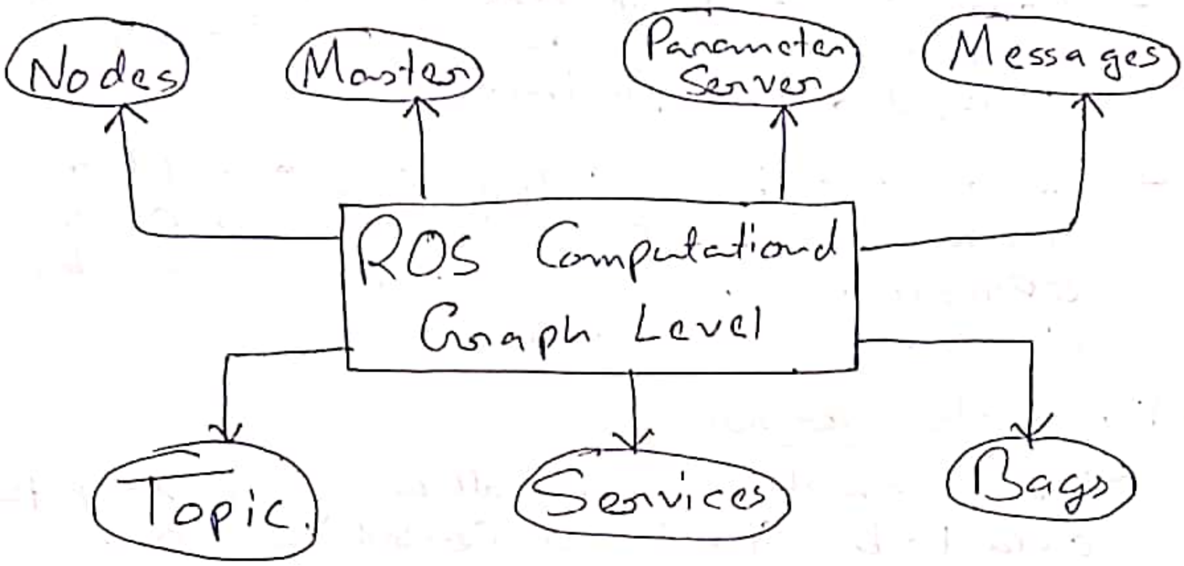
⇒ The Computation in ROS is done using a network of process called ROS nodes.

⇒ The main Concepts in the Computation graph are ROS Nodes, Master, Parameter Server, Messages, Topic, Services and Bags.

⇒ The ROS Communication related packages including Core client libraries, such as roscpp and rospython and the implementation of concepts such as topics, nodes, Parameters and Services are included in a Stack called ros_comm.

↳ This stak also consists of tools such as rostopics, rosparam, rosservice and rosnode.

⇒ Ros_Comm stack contains the ROS communication middleware packages and these packages are collectively called <u>ROS Graph layer</u>.



<u>Nodes</u>

→ Nodes are the process that perform computation.

→ Each ROS node is written using ROS Clint libraries such as <u>roscpp</u> and <u>rospy</u>.

→ In a robot, there will be many nodes to perform different kinds of tasks.

→ One of the aim of ROS nodes is to build simple Processes rather than a large process with all functionality.

Master

→ The ROS Master provides name registration and lookup to the rest of the nodes.

→ Nodes will not be able to fine each other, exchange messages or invoke Services without a ROS Master.

→ In a distributed System, we should run the master on one Computer and other remote nodes can find each other by Communicating with the master.

Parameter Server

→ The Parameter Server allows you to keep the data to be stored in a Central location.

→ All nodes can asses and modify these values

→ Parameter Server is a part of ROS Master.

Messages

→ Nodes Communicate with each other using messages.

Topics

→ Each message in ROS is transported using named buses called topics.

→ When a node sends a message through a topic, then we can say that node is publishing a topic.

→ When a node receives a message through a topic, then we can say that the node is subscribing to a topic.

→ The publishing node and Subscribing node are not aware of each other's existence.

→ Each topic has a Unique name, and any node can access this topic and Send data through it as long as they have the right message type.

Services

→ In some robot applications, Publish/Subscribe model will not be enough if it needs to request/response interaction.

→ ROS Services are used in these case.
↳ We can define a Service defination that contains two part:
→ Requests
→ Responses.

→ Using ROS Services, we can Write a Server node and client node.

Bags

→ Bags are a format for saving and playing back ROS message data.

→ Bags are very useful features when we work with complex robot mechanisms.

★ Understanding ROS Nodes

⇒ ROS nodes are a process that perform Computation using ROS clint libraries such as roscpp and rospy.

⇒ One node communicate with other nodes using ROS Topics, Services & Parameters.

⇒ All running nodes should have a name assigned to identify them from the rest of the System.

⇒ There is a rosbash tool to introspect ROS nodes :-

```
# rosnode info [node_name]
```

{This will print info about the node}

```
# rosnode kill [node_name]
```

{This will kill a running node}

```
# rosnode list
```

{This will list all the running nodes}

```
# rosnode machine [machine_name]
```

{This will list the nodes running on a Particular machine or a list of machine}

```
# rosnode ping
```

{This will check the connectivity of a node}

```
# rosnode cleanup
```

{This will purge the registration of Unreachable nodes}

★ ROS messages

⇒ ROS nodes communicate with each other by Publishing message to a topic.

⇒ Message are a Simple data Structure Containing field types.

⇒ ROS has inbuilt tools called rosmsg to get information about ROS messages:-

```
# rosmsg show [message]
```

{This shows the message description}

```
# rosmsg list
```

{This lists all message}

```
# rosmsg md5 [message]
```

{This displays md5 sum of a message}

{MD5 Checksum comparision is used to confirm whether the publisher and subscriber exchange the Same message data types.}

```
# rosmsg Package [Package_name]
```

{This lists message in a Package}

```
# rosmsg Packages [Package-1] [Package-2]
```

{This lists packages that Contain messages}

★ ROS topics

⇒ ROS topics are named buses in which ROS nodes exchange messages.

⇒ ROS nodes are not interested to know which node is publishing the topic or subscribing topics, it only looks for the topic name and whether the message types of publisher and subscriber are matching.

⇒ ROS node communicate with topics using TCP/IP based transport known as <u>TCPROS</u>. (Default)

↳ Another type is <u>UDPROS</u>.

⇒ The rostopic command can be used to get information about ROS topics:-

rostopic bw /topic

{Displays the bandwidth used by a given topic}

rostopic echo /topic

{Points the content of a given topic}

rostopic find /message-type

{This command will find topics using the given message type}

rostopic info /topic

{Points information about the topic}

rostopic list

{lists all the active topics in ROS Systems}

rostopic pub /topic message-tye args

{This Command is used to publish a Value to a topic with a message type}

rostopic type /topic

{This will display type of message for the given topic}

★ ROS Services

⇒ When we need a request/response kind of Communication in ROS, we have to use the ROS Services.

↳ Mainly used in distributed System.

⇒ WE have to define a request datatype and a response datatype in a srv file.

rosservice call /service args

{This tool will Call the Service using the given arguments}

rosservice find Service_type

{find Services in the given Service type}

rosservice info /Services

{This will print information about the given Service}

rosservice list

{list active service running on the System}

rosservice type /service

{This .Command will print the service type of a given Service}

rosservice uri /service

{This tool will print the Service ROSRPC URI}

★ ROS bags (·bag)

⇒ A bag file in ROS is for storing ROS message data from topics and services.

⇒ Bag files are created, using the rosbag command, which will subscribe one or more topics & store the message's data in a file as it's received.

⇒ The main application of rosbag is datalogging. The robot data can be logged and can visualize and process offline.

rosbag record [topic_1] [topic_2] -o [bag_name]

{This command will record the given topic into a bag file}

rosbag play [bag_name]

{This will playback the existing bag file}

★ Understading ROS Master

⇒ ROS Master is much like DNS server
↓
(Domain name System)

⇒ When any node starts in the ROS System, it will start looking for ROS Master and register the name of the node in it.

↳ Master has the details of all nodes currently running on the ROS System.

★ Using the ROS Parameter

⇒ While programming a robot, we might have to define robot parameters such as robot Controller gain such as P, I and D.

⇒ ROS provides a parameter Server, which is a shared server in which all ROS nodes can access parameters from this Server.

↳ A node can read, write, modify and delete Parameter values from the parameter Server.

⇒ The rosparam tool used to get and set the ROS parameter from the Command line:-

```
# rosparam set [Parameter_name] [value]
```

{To set a value in the given parameter}

```
# rosparam get [Parameter_name]
```

{To retrieve a value from the given parameter}

```
# rosparam load [YAML file]
```

{ROS parameter can be saved into a YAML file and it can load to the parameter server using this Command}

```
# rosparam dump [YAML file]
```

{Dump existing ROS parameters to a YAML file)

```
# rosparam delete [Parameter_Name]
```

{To delete a given Parameter)

```
# rosparam list
```

{List the existing parameter name}

★ Understanding ROS Community level

- Distributions
  ↳ ROS distributions are a collection of Versioned meta package that we Can install.
- Repositories
  ↳ Where different institutions can develop and release their own robot Software components.
- ROS Wiki
  ↳ Main forum for documenting information about ROS
- ROS Answer
  ↳ To ask questions related to ROS
- Blog
  ↳ The ROS blog updates with news, Photo, and Videos related to the ROS Community.

★ Creating a new package in ROS

⇒ Packages are Created is Src folder of Catkin workspace (Catkin_ws)

① Catkin_Create_Pkg   Package name   dependencies

↓

{eg ROS_Tutorial}   {std_msgs, rospy, roscpp}

② go to Catkin work directory

③ Catkin_make. {To Compile the workspace}

★ To run a node

rosrun Packagename nodename.