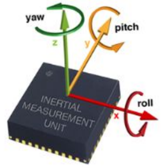
Contents

What Makes a Robotic System ?

Camera

IMU



Laser scanner

Robot

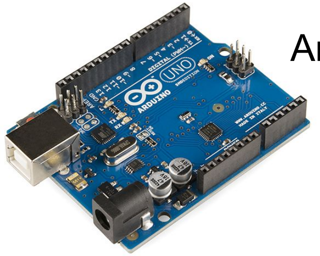
Motor & Encoder

GPS

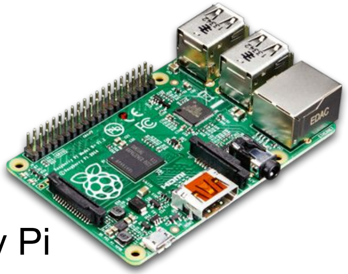
A cooperative system of sensors and actuators...

Intel NUC

Arduino



Robot

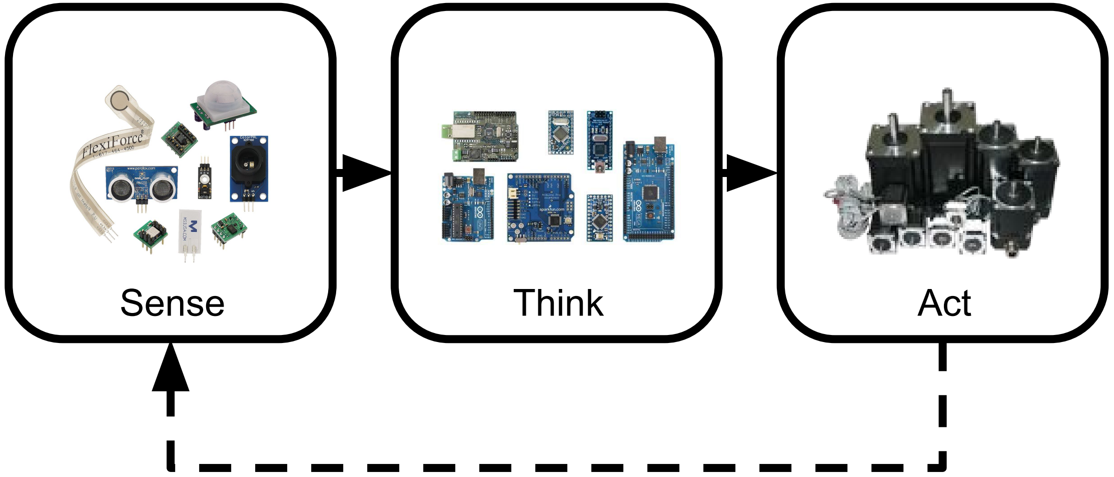


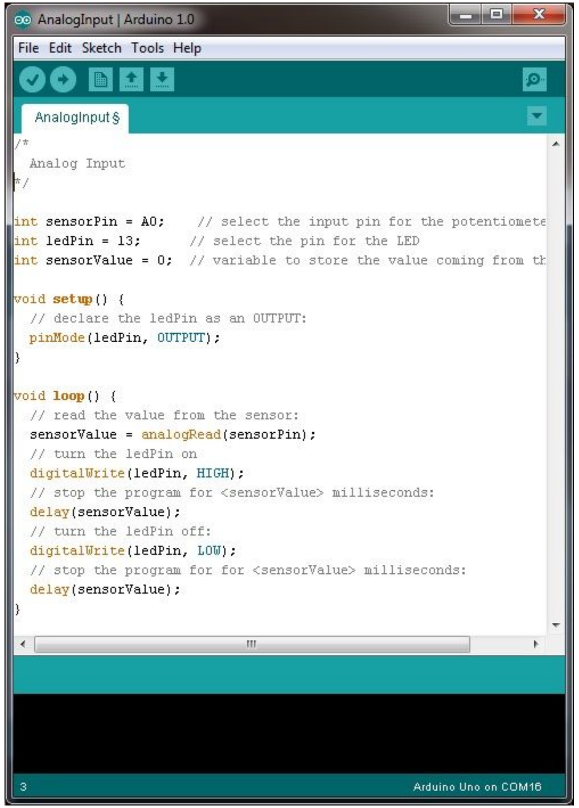
Raspberry Pi

ODROID

...and processors to help in this cooperation

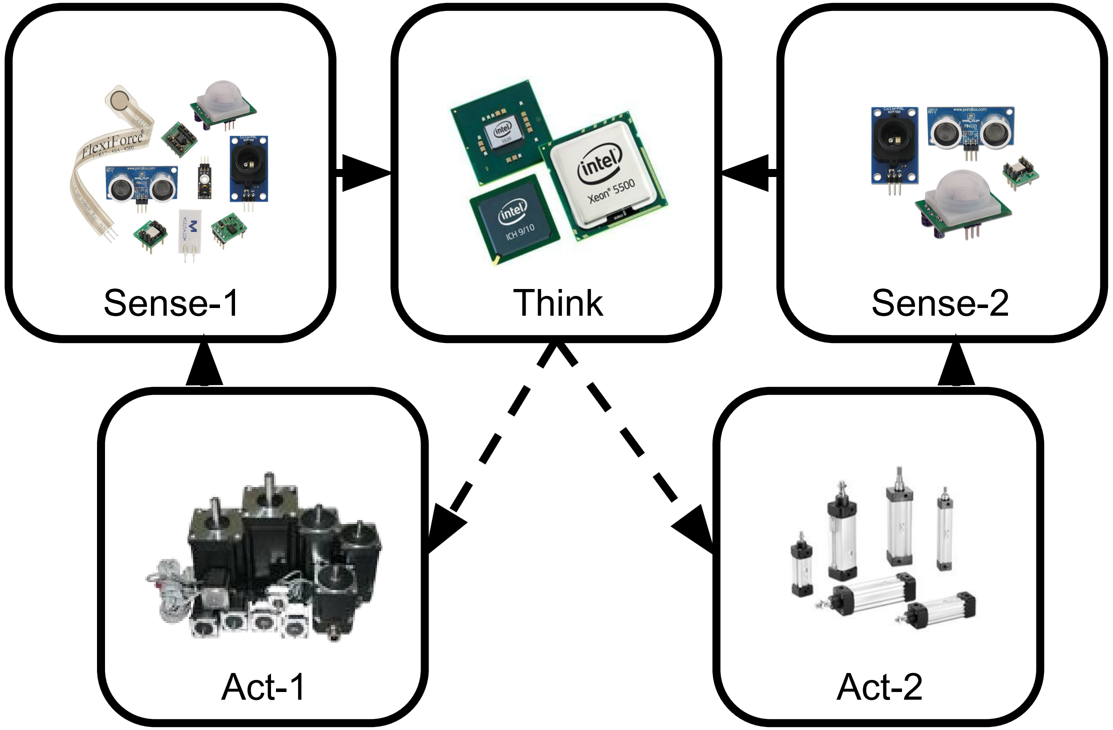
Robotic System: Sequential workflow

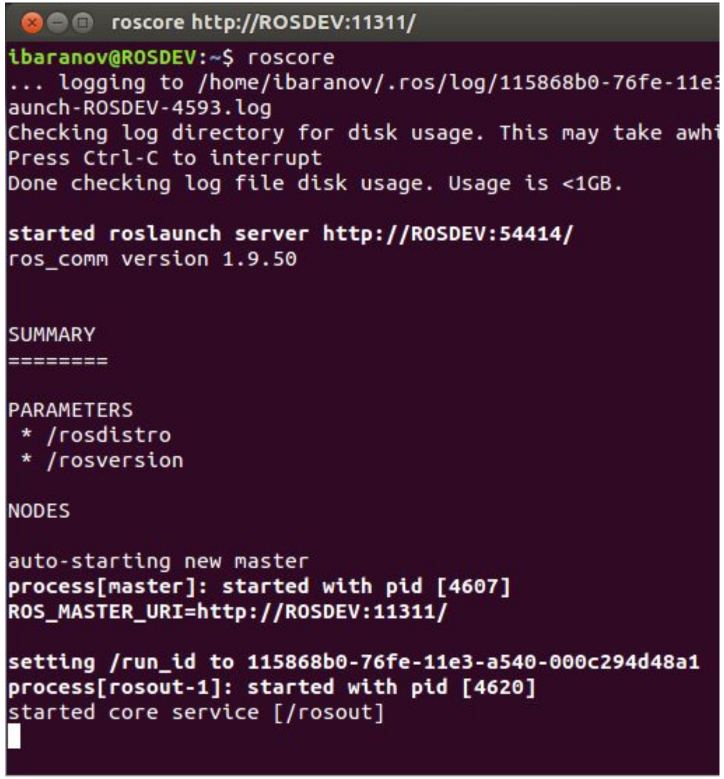




```
/*
  Analog Input
*/

int sensorPin = A0;    // select the input pin for the potentiomete
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from th

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

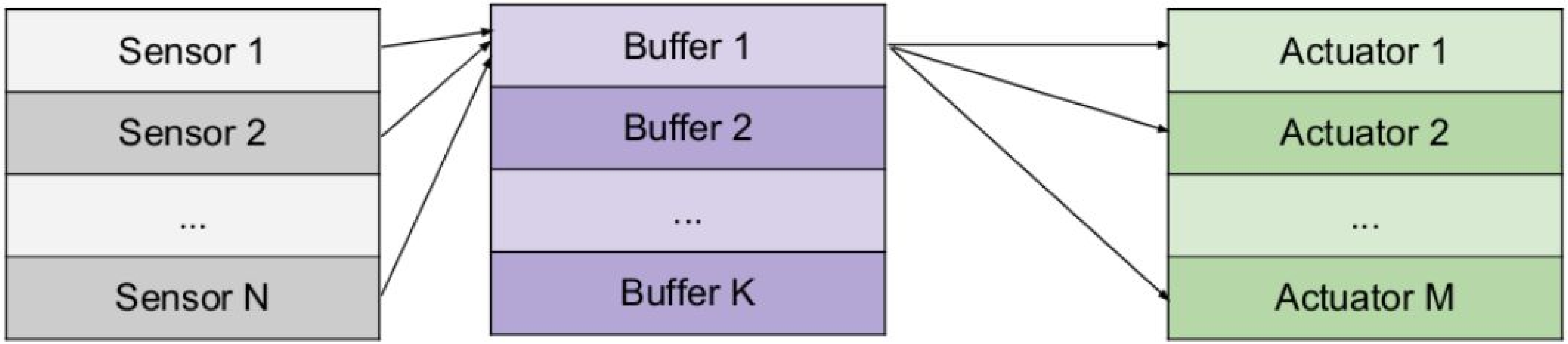
Robotic System: Parallel workflow





```
ibaranov@ROSDEV:~$ roscore
... logging to /home/ibaranov/.ros/log/115868b0-76fe-11e3
aunch-ROSDEV-4593.log
Checking log directory for disk usage. This may take awhi
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ROSDEV:54414/
ros_comm version 1.9.50


SUMMARY
========

PARAMETERS
 * /rosdistro
 * /rosversion

NODES

auto-starting new master
process[master]: started with pid [4607]
ROS_MASTER_URI=http://ROSDEV:11311/

setting /run_id to 115868b0-76fe-11e3-a540-000c294d48a1
process[rosout-1]: started with pid [4620]
started core service [/rosout]
```

Implementing a Robotic System in Parallel Mode



So, How do we do this ? We all have the used a software that does all this.

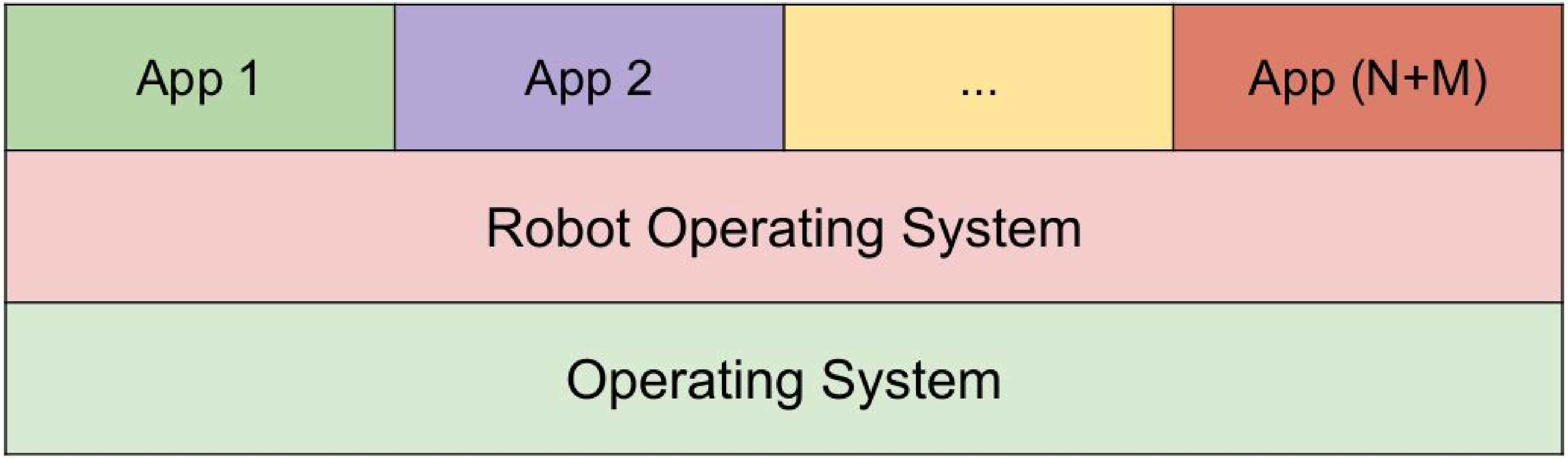
Source : Lakshay Garg

Think

The Operating System

ROS to the rescue

Why not let the OS handle the tedious task.



Source : Lakshay Garg

What is ROS ?

- **ROS** or **Robot Operating System**.
- **Framework** for **robotic software development** providing Operating System Like functionality, including **hardware abstraction**, **low-level-device control**, **message-passing** between processes, and **package management**.
- The origins lie in Stanford Artificial Intelligence Lab and was further developed at Willow Garage.

Stanford
ARTIFICIAL
INTELLIGENCE

Willow
Garage

- Available for all major operating systems
- Massively growing user base.

ROS 10 years 11 Distros

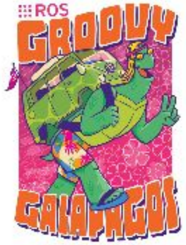
2010

2010

2011

2011

2012



2012

2013

2014

2015

2016

2017

open robotics

ROS is more than just a “middleware”.

Tools	**Capabilities**	**Plumbing**	**Ecosystem**
● Simulation ● Visualization ● GUI ● Data Logging ● Debugging ● Testing	Libraries for ● Mobility ● Perception ● Manipulation ● Control	● Process Management ● Message Passing Interface ● Device Drivers	● Large community of Developers and organizations. ● Documentation ● Tutorials

Credit : ETHZ RSL

Philosophy of ROS

- **Peer to Peer**
  - Individual programs (nodes) communicate over ROS API (messages, etc)
- **Distributed**
  - Nodes can communicate over a network.
- **Multilingual**
  - Native support for C++, Python and Lisp, Experimental support exist for Java and Lua too. Client Libraries for Matlab etc.
- **Light Weight**
  - Doesn’t slow the programs or even hinder their ability to work with other frameworks.
- **Free and Open Source**

Credit : ETHZ RSL

What ROS isn’t ?

- An actual Operating System
- A programming Language
- A programming environment/IDE
- A hardware.

Credit : ETHZ RSL

ROS Architecture

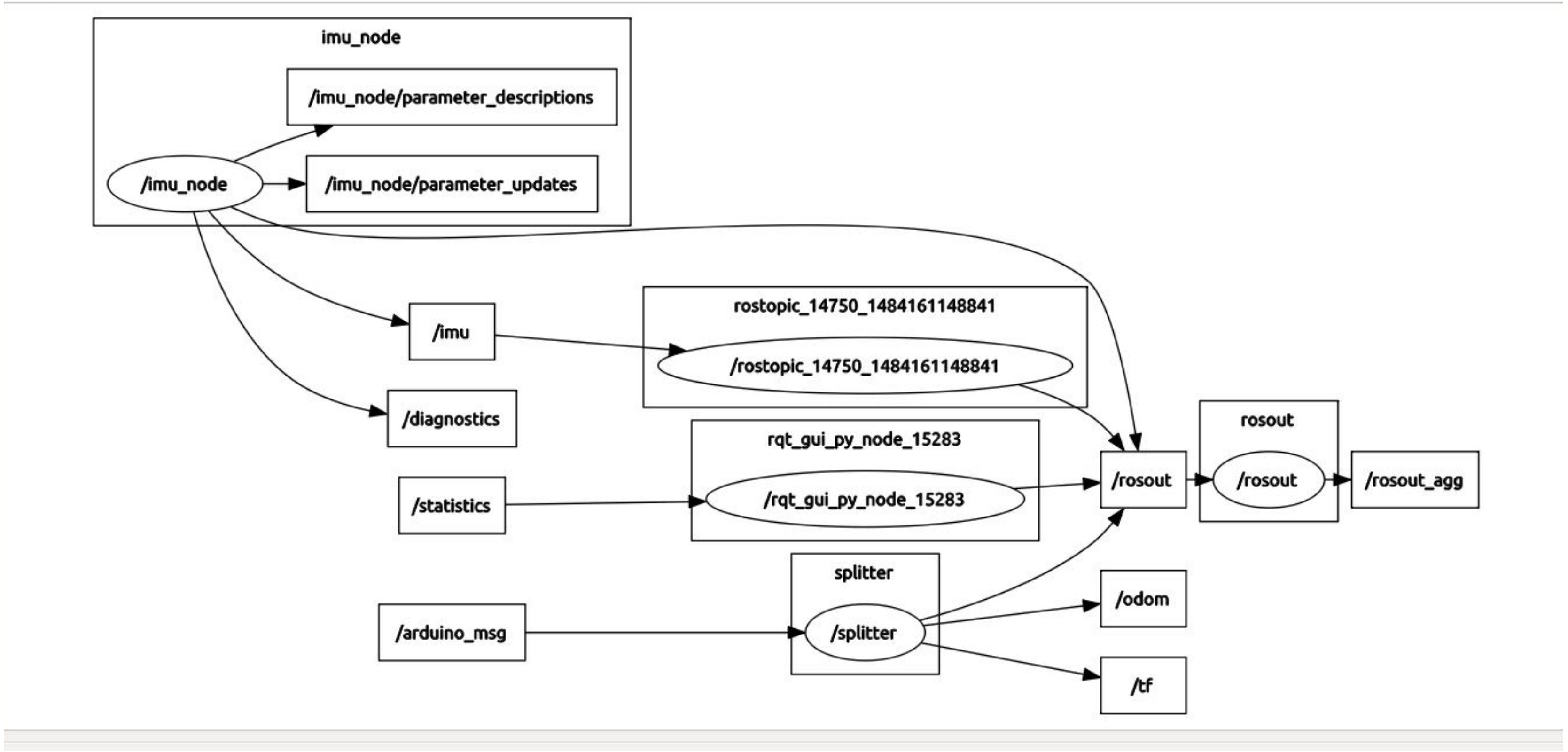
ROS Communication Layer : ROS Core

- **ROS Master**
  - Centralized Communication Server based on XML and RPC
  - Negotiates the communication connections
  - Registers and looks up names for ROS graph resources
- **Parameter Server**
  - Stores persistent configuration parameters and other arbitrary data.
- ***`rosout`***
  - Network based *`stdout`* for human readable messages.

ROS Communication Layer : Graph Abstraction

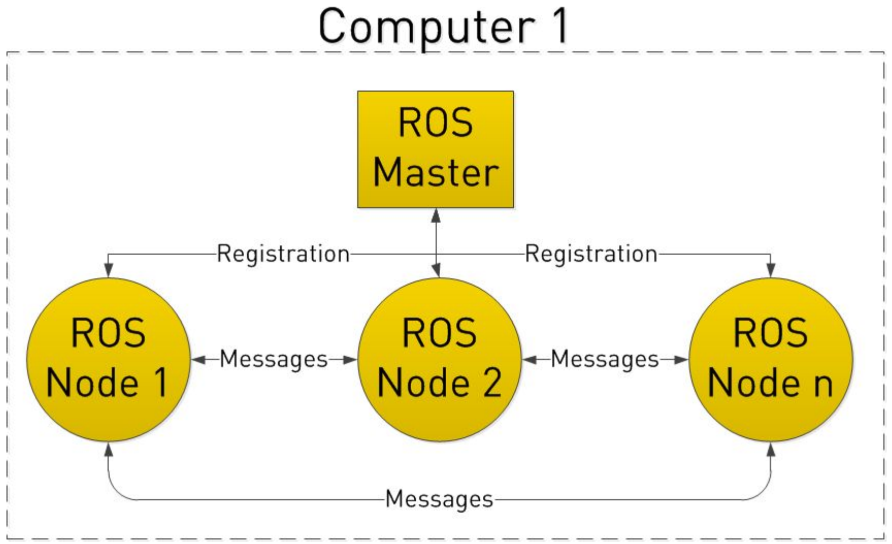
- **Nodes**
  - Processes distributed over the network.
  - Serves as source and sink for the data sent over the network
- **Parameters**
  - Persistent data such as configuration and initialization settings, i.e the data stored on the parameter server. e.g camera configuration
- **Topics**
  - Asynchronous many-to-many communication stream
- **Services**
  - Synchronous one-to-many network based functions

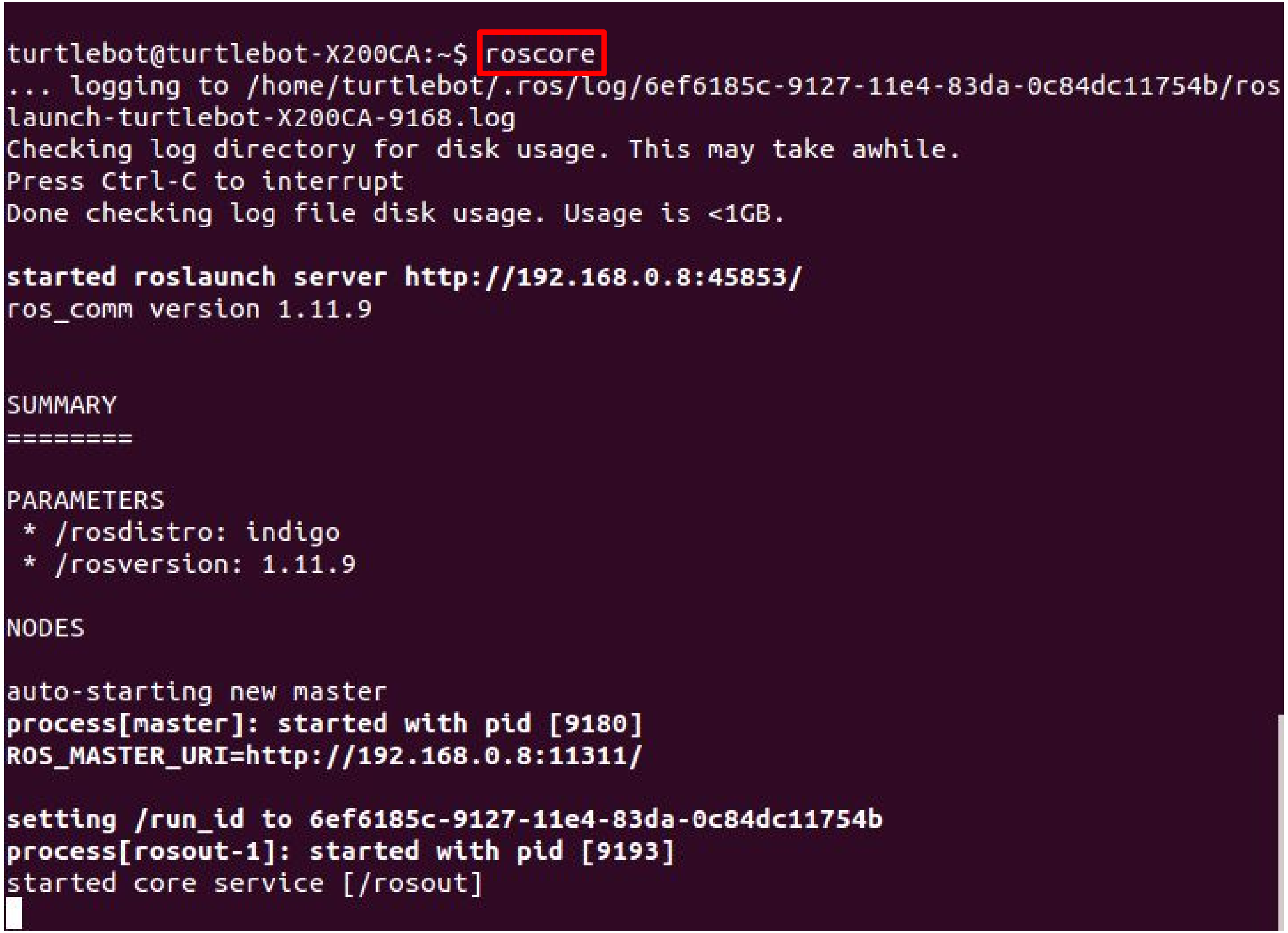
ROS Communication Layer : Graph Abstraction



#1: ros::roscore

- ROS master process is called roscore
- Allows intercommunication between *nodes* (processes using ROS framework)
- *Syntax:* `$ roscore`

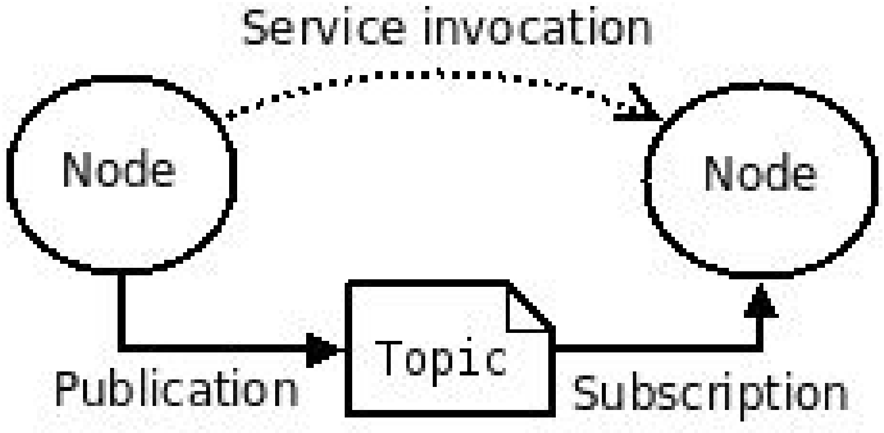


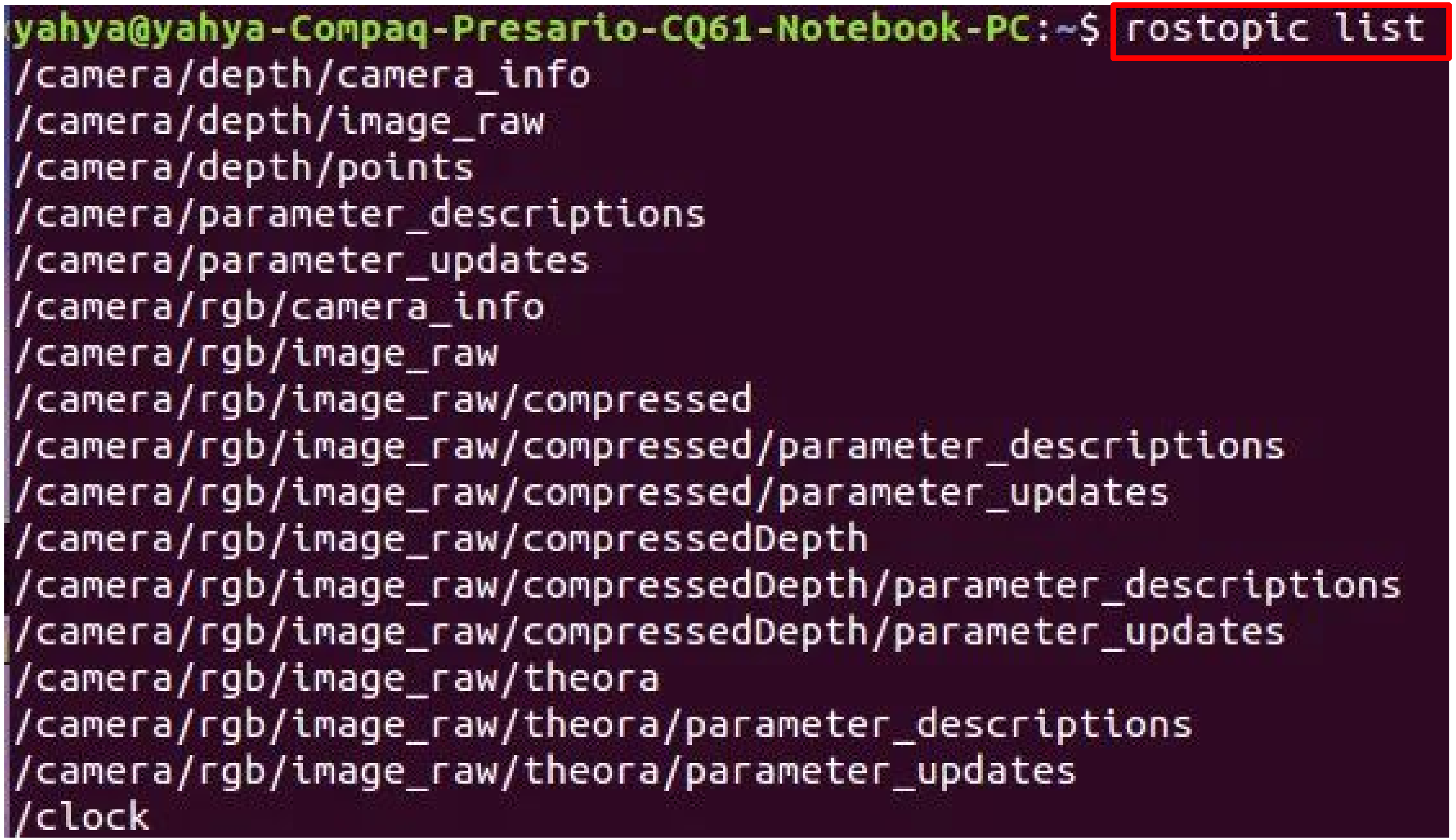


```
turtlebot@turtlebot-X200CA:~$ roscore
... logging to /home/turtlebot/.ros/log/6ef6185c-9127-11e4-83da-0c84dc11754b/ros
launch-turtlebot-X200CA-9168.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.8:45853/
ros_comm version 1.11.9


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.9

NODES

auto-starting new master
process[master]: started with pid [9180]
ROS_MASTER_URI=http://192.168.0.8:11311/

setting /run_id to 6ef6185c-9127-11e4-83da-0c84dc11754b
process[rosout-1]: started with pid [9193]
started core service [/rosout]
```

#2: ros::Topic

- *Topics* are used to send messages from a node to other nodes
- *Publish* = send message to a topic
- *Subscribe* = receive message from a topic





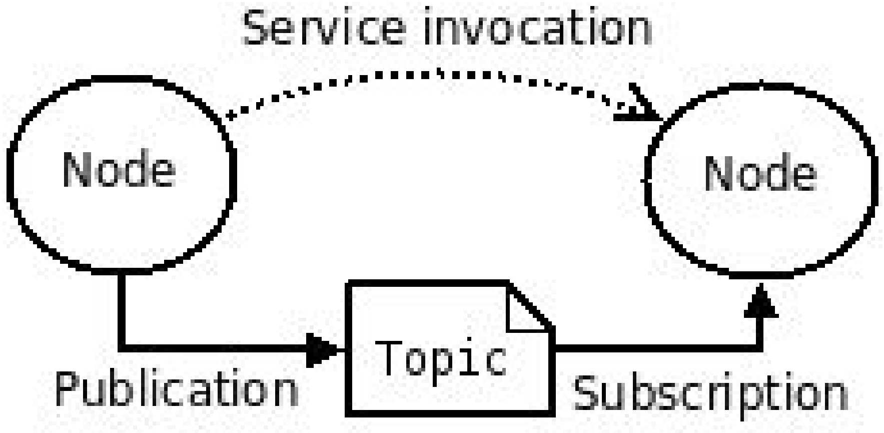
```
yahya@yahya-Compaq-Presario-CQ61-Notebook-PC:~$ rostopic list
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/points
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
/clock
```

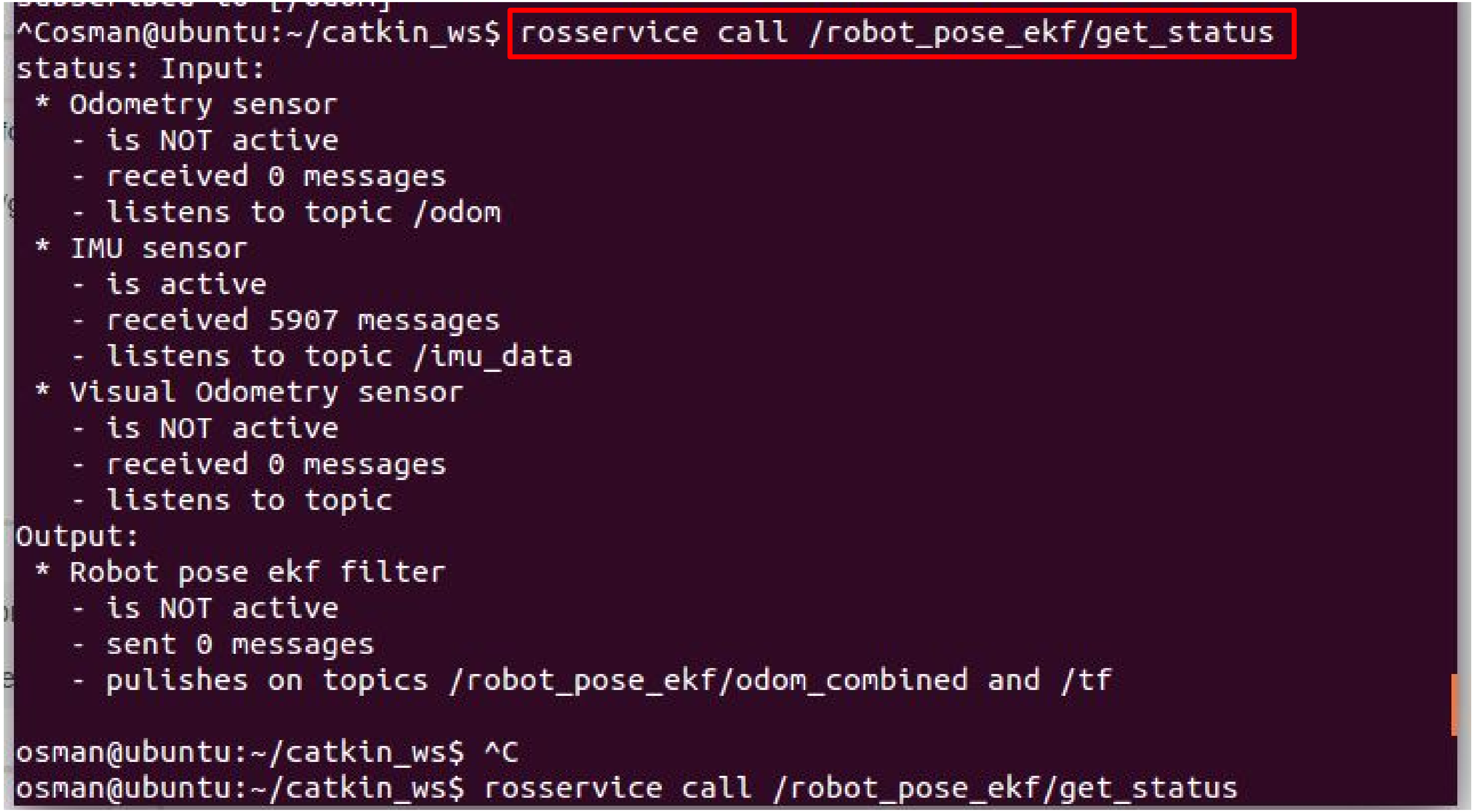
#3: ros::Service

- *Services* are used to send a request to another node and receive a response

- A service is called with a *request* struct and *response* struct is returned

- These structs are different from *topic* messages

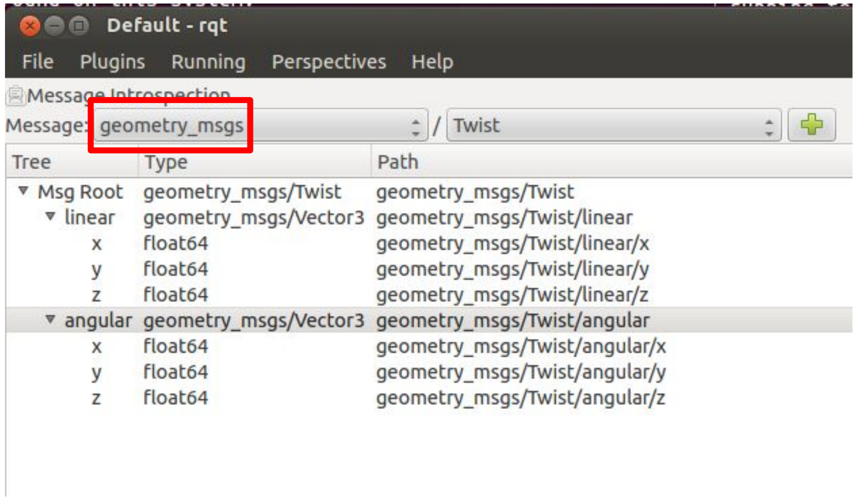




```
^Cosman@ubuntu:~/catkin_ws$ rosservice call /robot_pose_ekf/get_status
status: Input:
 * Odometry sensor
   - is NOT active
   - received 0 messages
   - listens to topic /odom
 * IMU sensor
   - is active
   - received 5907 messages
   - listens to topic /imu_data
 * Visual Odometry sensor
   - is NOT active
   - received 0 messages
   - listens to topic
Output:
 * Robot pose ekf filter
   - is NOT active
   - sent 0 messages
   - pulishes on topics /robot_pose_ekf/odom_combined and /tf

osman@ubuntu:~/catkin_ws$ ^C
osman@ubuntu:~/catkin_ws$ rosservice call /robot_pose_ekf/get_status
```

#4: ros::Message

- *Messages* in ROS are used for inter process interactions like topics or services
- Defined as text files with internal variable declarations
- Single file contains both request and response



Message: geometry_msgs / Twist

Tree	Type	Path
Msg Root	geometry_msgs/Twist	geometry_msgs/Twist
linear	geometry_msgs/Vector3	geometry_msgs/Twist/linear
x	float64	geometry_msgs/Twist/linear/x
y	float64	geometry_msgs/Twist/linear/y
z	float64	geometry_msgs/Twist/linear/z
angular	geometry_msgs/Vector3	geometry_msgs/Twist/angular
x	float64	geometry_msgs/Twist/angular/x
y	float64	geometry_msgs/Twist/angular/y
z	float64	geometry_msgs/Twist/angular/z

ROS workflow

ROS workflow : Demo

Please Visit this link and clone the repository.

- https://github.com/harshsinh/ros-demo

- git clone https://github.com/harshsinh/ros-demo.git
- cp ros-demo/demo ~/catkin_ws/src -rf
- cd ~/catkin_ws/
- catkin_make

ROS workflow : workspace

The typical ROS workspace would look somewhat like this :

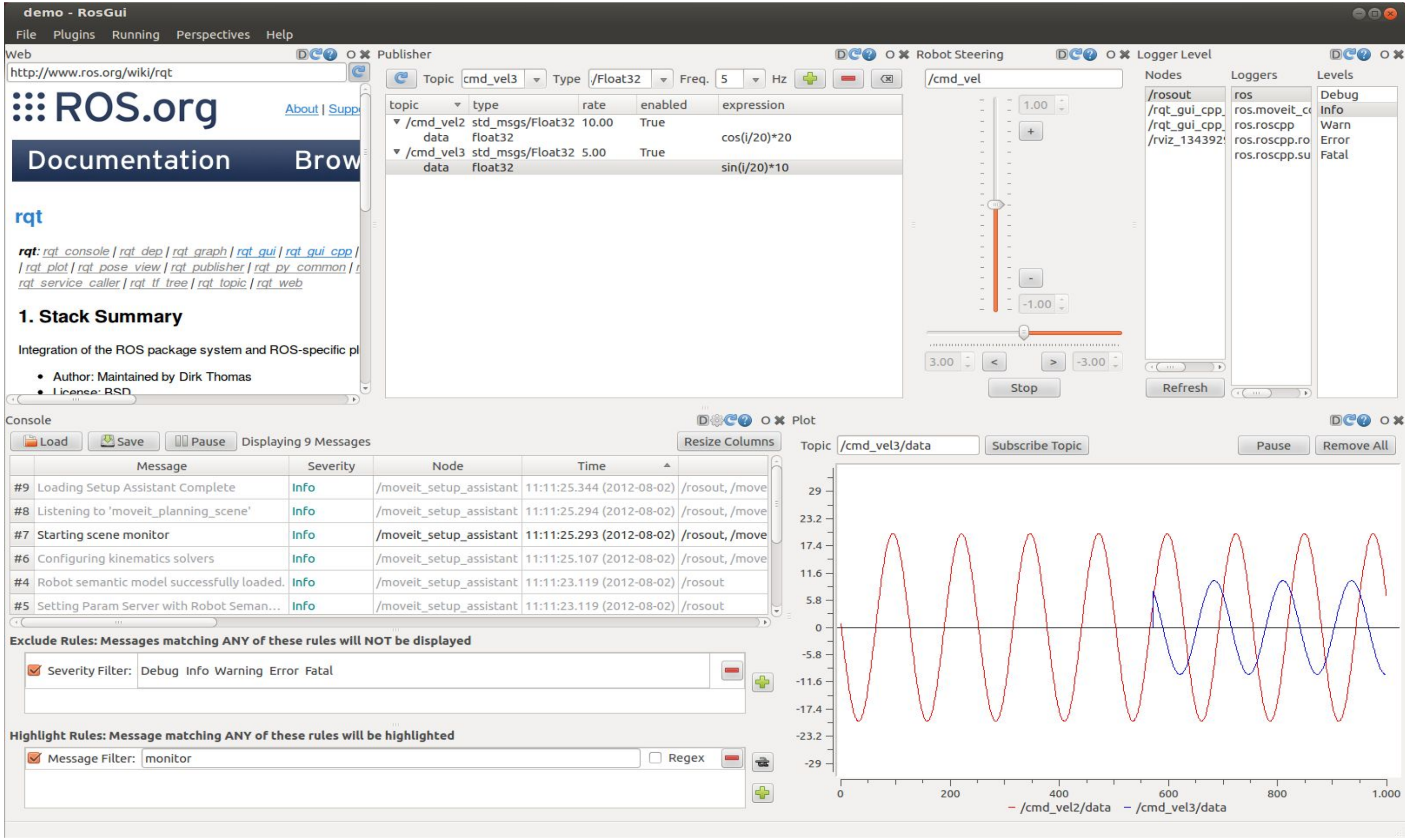
- catkin_ws/
    - build/
    - devel/
    - src/
        - CMakeLists.txt
        - Package_1/
        - Package_2/
            - CMakeLists.txt
            - package.xml
            - include/
            - launch/
            - src/

ROS Build System : catkin

- ROS uses a `catkin` build system.
    - `catkin_make` or `catkin build` would generate **executables**, **libraries** and **interfaces**.
    - Choose one of the above and stick to it.
    - Always `*source*` your workspace after you build.
- A cross platform build system which treats your entire workspace as a single CMake project where each project is a subproject then on.

Add Ons

rqt : A QT based GUI developed for ROS



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 / r 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

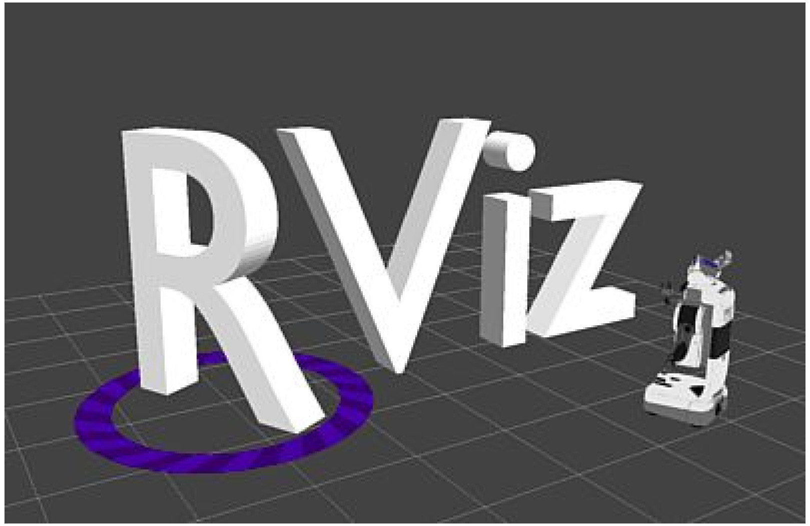
Topic /cmd_vel3/data Subscribe Topic Pause Remove All

rqt : A QT based GUI developed for ROS

- Lots of different plugins.
  - `*rqt_graph*`, `*rqt_image_view*`, `*rqt_console*` to name a few.
  - `*rqt_graph*` can be used to view the graph structure of the system, i.e the nodes, and how are they related etc.
- Multiple plugins can be run simultaneously.
- Anyone can add more custom plugins written in C++ or Python.

RVIZ : ROS

- A Powerful tool for 3D Visualization in ROS
- Modular state and sensor visualization
- Excellent community support.



Gazebo Simulator

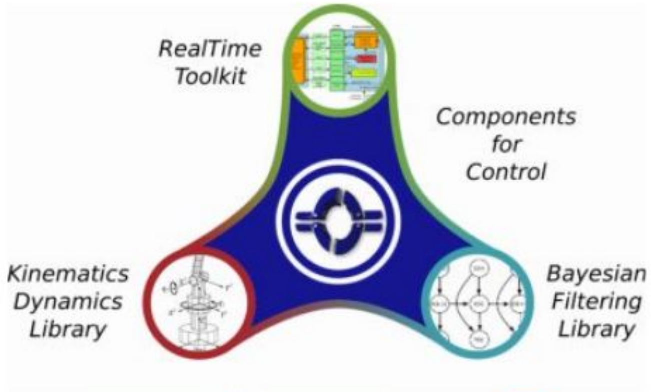
- Simulate 3D rigid body dynamics
- Simulate a variety of different sensors, many of them are available online ready to use.
- Has many environments and robots pre-implemented.
- With ROS interface, it can be used to directly to test the applications inside a simulation.
- Has even more plugins available.

Further References

- ROS Wiki
  - http://wiki.ros.org/
- Tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Available Packages
  - http://www.ros.org/browse/list.php
- ROS Style Guides
  - http://wiki.ros.org/StyleGuide
- ROS Cheat Sheet
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
- ROS Answers
  - https://answers.ros.org/

ROS is not alone

BULLET PHYSICS LIBRARY



pcl

MoveIt!

OpenCV

Player

Stage

ROS industrial

S2DR

Thank You