ROBOTIQUE AEROTERRESTRE

Systèmes Embarqués

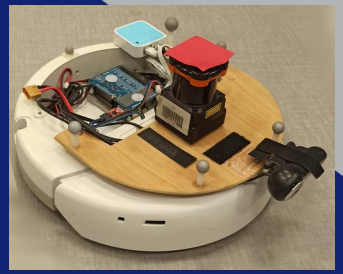
Middlewares robotiques
(ORoCoS - ROS)

ROBOTIQUE AEROTERRESTRE

Systèmes Embarqués





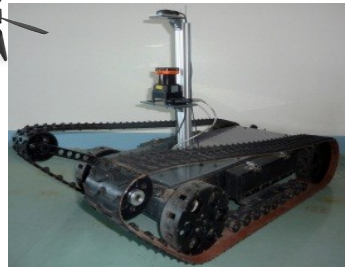












ISAE - Pascal CHAUVIN

Système embarqué (PixHawk + Odroid XU-4 (linux))

Drone

(config Pixhawk)

Capteurs spécifiques

USB
I²C
L.Série

Développement communautaire

Pixhawk

Lois de commande par défaut

Mécanismes de sécurité

Protocole MAVLINK

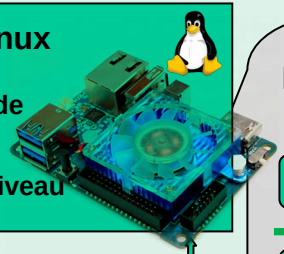
Liaison série

Calculateur Linux

Estimation état

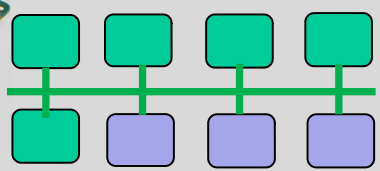
Lois de commande (mode offboard)

Fonctions haut niveau



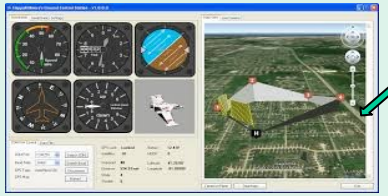
Développement communautaire

Logiciel à base de composants (ROS OROCOS)



Commandes manuelles
+ passage en mode offBoard

Liaison de données longue portée bas débit



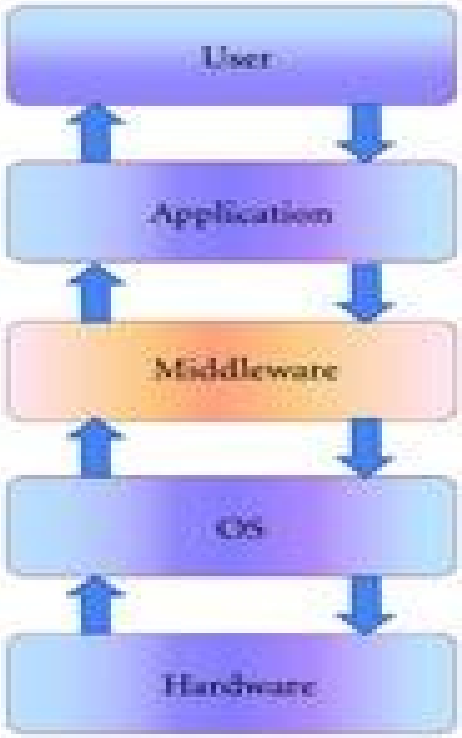
Liaison de données haut débit WIFI

Interfaces spécifiques

Outils

- -Simulateurs (MORSE, Gazebo
- -Matlab
- -Enregistrement de données
- -Visualisation
- -mesure de performances
- - …..

Middleware

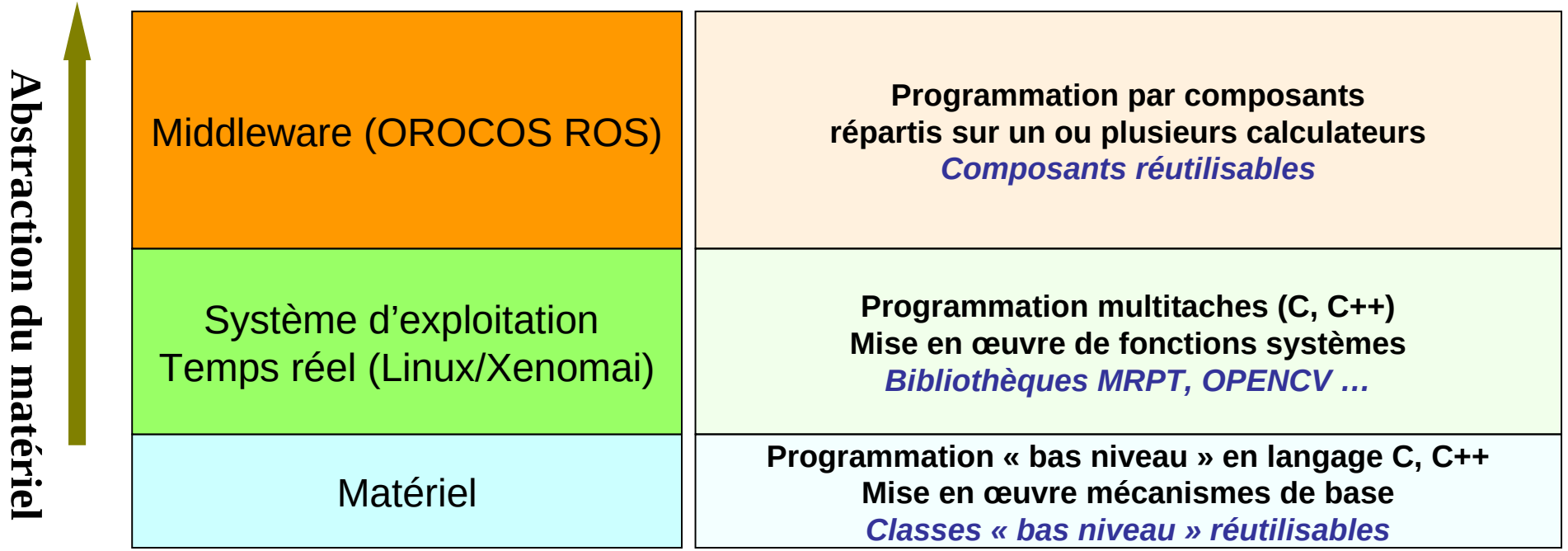


- Abstraction du matériel (et de l'OS)

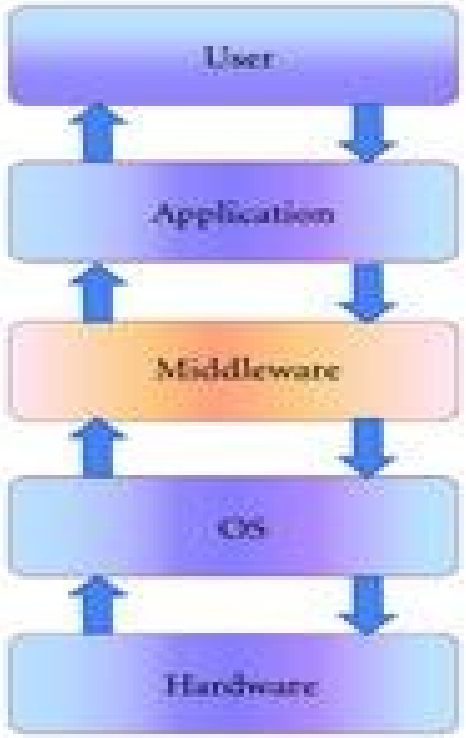
- Développement modulaire orienté composant

- Des outils de simulation associés pour la mise au point ds composants

Développement logiciel embarqué



Some middlewares ...

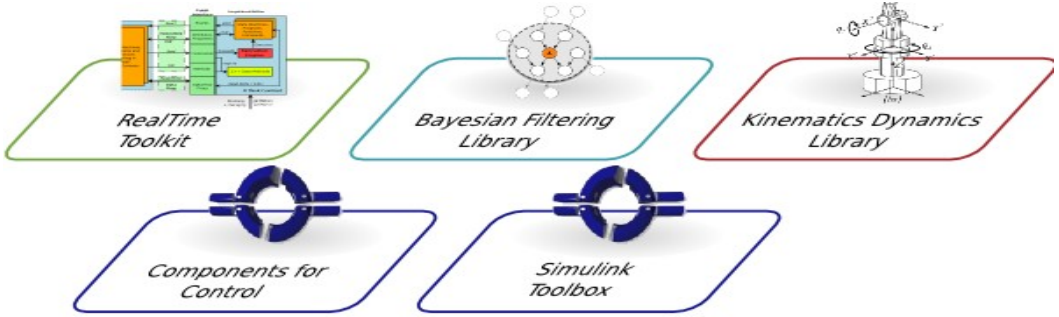


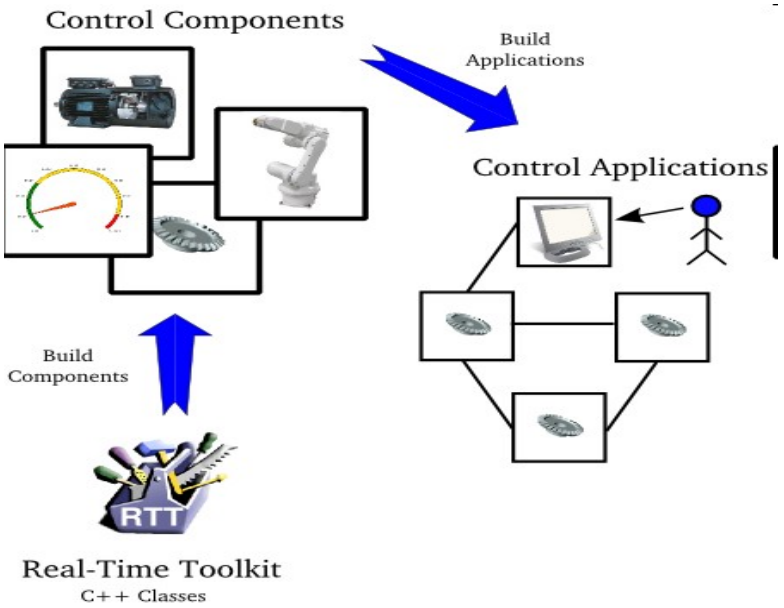
Player
stage
RT-middleware
Urbi
MIRO
Orca
OpenRDK
ROS
ROS 2
YARP
OROCOS
MAUVE
MOOS
Hla (CERTI)
...

OROCOS

Open Robot Control Software (was born in December 2000)

- Une application est construite à base de composants
- Un composant est décrit par des flots de données, des services, des propriétés
- Un composant peut réagir à des événements, traiter des requêtes et exécuter des scripts en temps réel





Institutes

K.U.Leuven (B), FU Berlin (D), Polytechnic University of Catalonia (ES), University of Florida (US), German research centre for Artificial Intelligence (DFKI,Bremen, D), University of New South Wales (Sydney,AU), University of Maryland (USA), Polytechnic University of Milan (I), University of Southern
Denmark, Onera (F), Irisa (F), Cea (F)

Companies

three FMTC member companies (with already one product several years on the market!); Willow Garage; and some other machine or robotics builder companies in Belgium, France, Spain, Canada, USA (NASA) and the Netherlands, with several products on the market

Hard realtime is Orocos' competitive advantage:

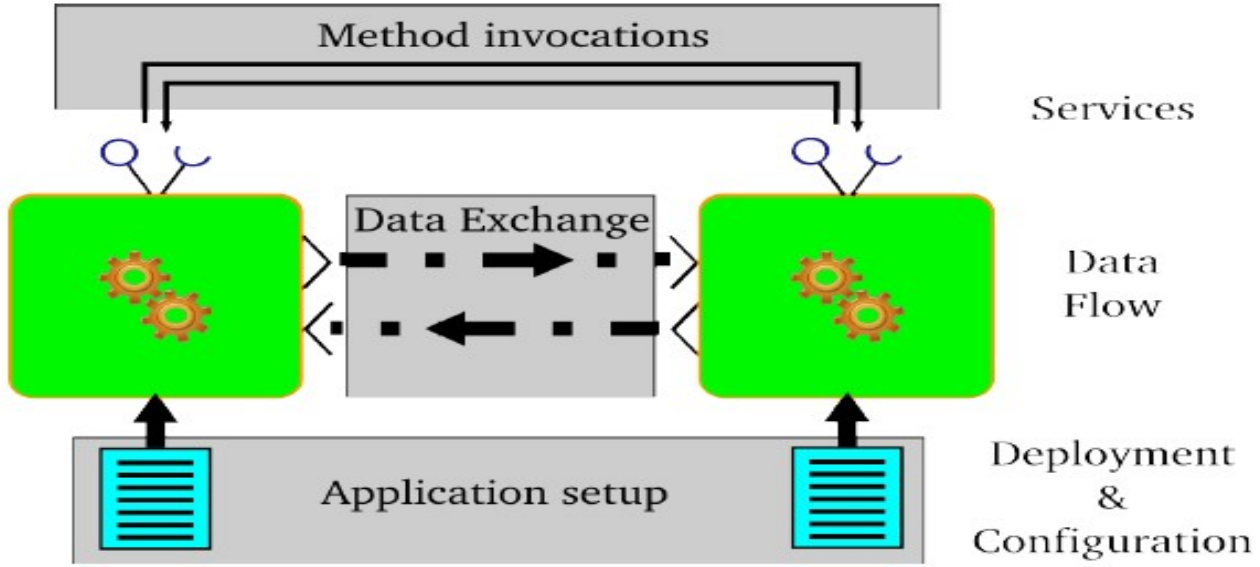
- Lock-free data ports favour highest priority component activity.
- Realtime-aware memory management.
- Does not prevent non-realtime use!
- ... in an extremely flexible and extendibleprogramming environment

Orocos Toolchain is supported on:

- Linux 32/64bit (GNU,clang,Intel)
- Real-Time Linux Extensions
  - Xenomai
  - Real-Time Application Interface (RTAI)
- Mac OS-X (GNU)
- Windows (XP->7) 32/64bit (MSVS2005-2010)
- QNX (GNU) – beta

OROCOS Component's communication

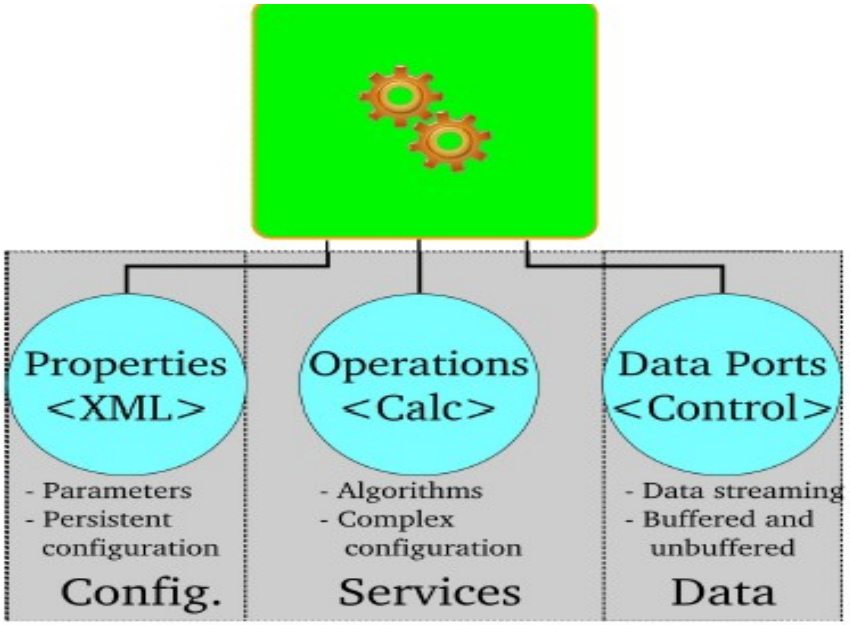
Open Robot Control Software

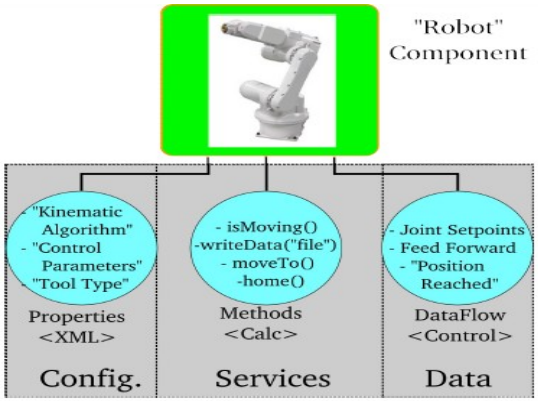


- **Configuration of parameters**
- **Exchange (streaming) data**
- **Cooperate to achieve a task**

OROCOS Component's interface

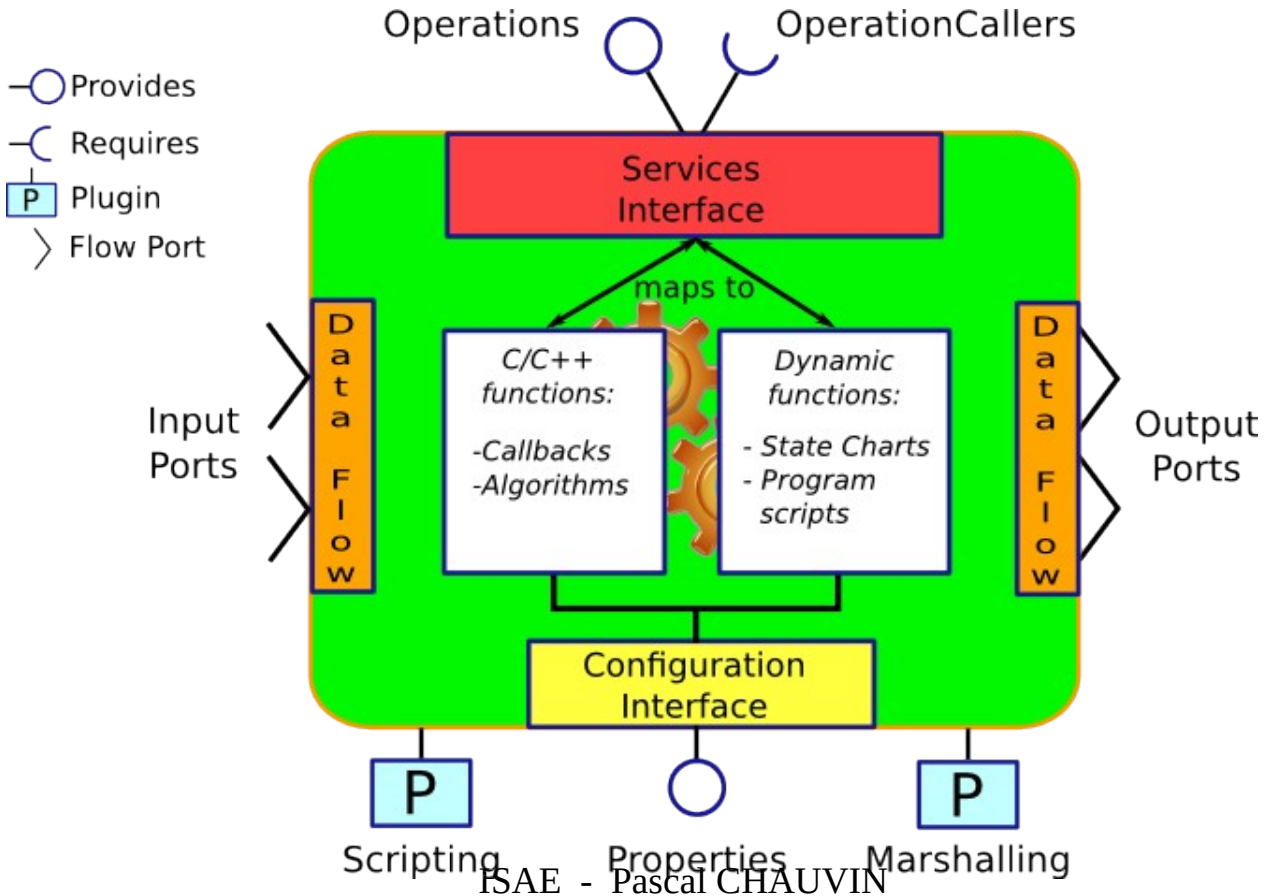
Open Robot Control Software





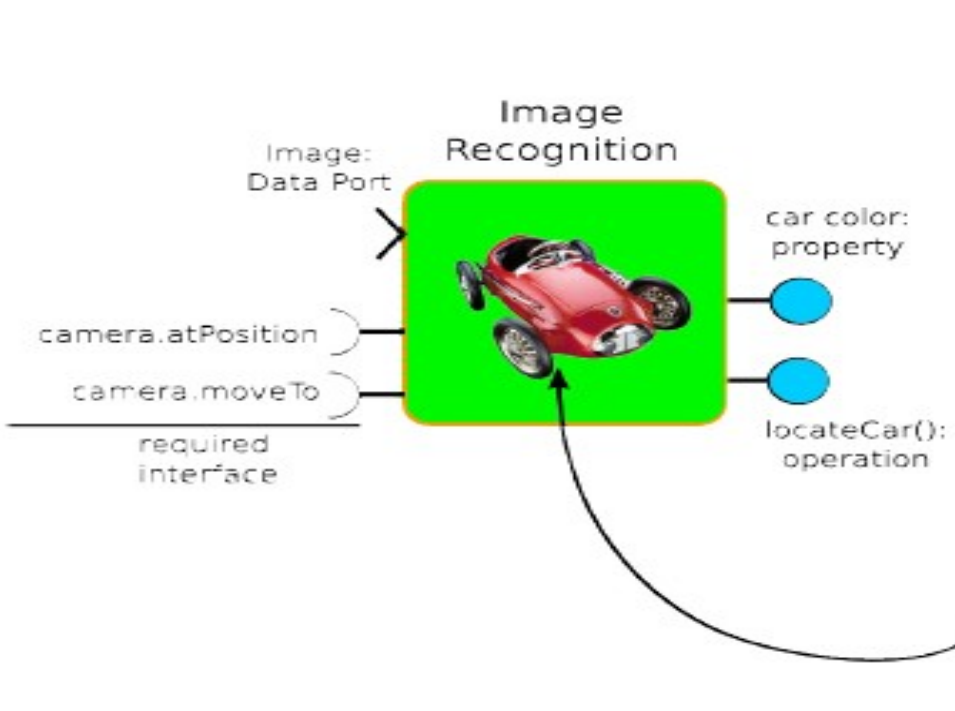
OROCOS Component

Open Robot Control Software



OROCOS Component State Machine Script

Open Robot Control Software



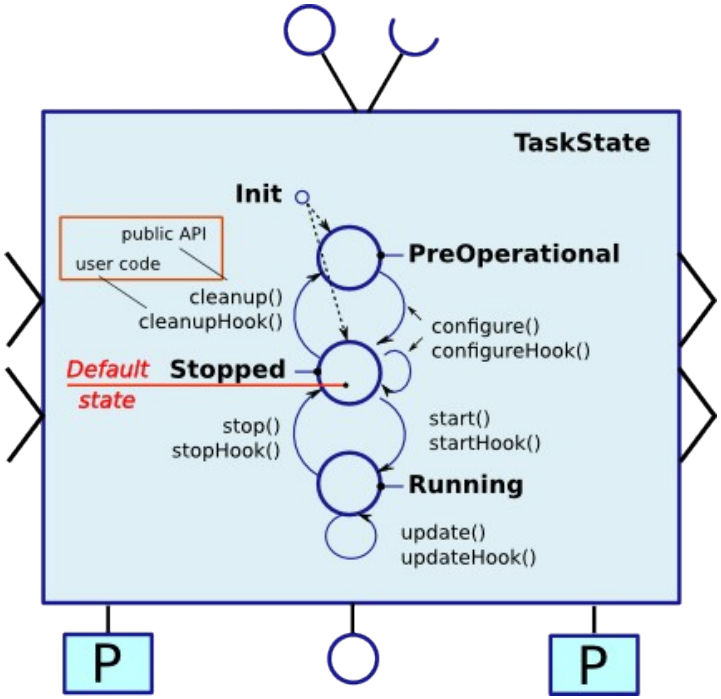
file: statemachine.osd

```
StateMachine ExampleSM
{
   initial state wait_for_image {
       // on imageReady event, make
       // transition to other state:
       transition Image( new_image )
            select image_captured;
   }

   state image_captured {
     // program executed when this state
     // is entered:
     entry {
        set p = this.locateCar();
        camera.moveTo( p );
     }
     // when entry is done, go back:
     if camera.atPosition( p )
          select wait_for_image;
   }

   final state end {}
}
RootMachine ExampleSM sm;
```

Component's life cycle

- Allows non real-time configuration and cleanup
- Starting is only allowed once configured properly

States : [U]ninitialised, [S]topped, [R]unning, [X]error



Task Application Code

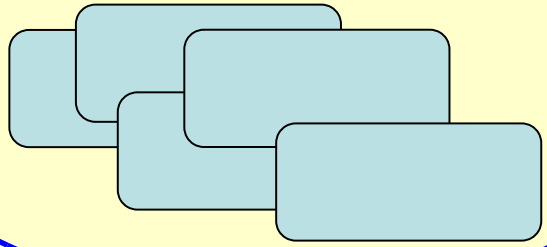
```
class MyTask
   : public TaskContext
 {
 public:
     MyTask(std::string name)  : TaskContext(name)
     {
       // see later on what to put here.
     }
    /* This function is for the configuration code.
    * Return false to abort configuration. */
     bool configureHook() {
       // ...
         return true;
     }
 /* This function is for the application's start up
    code.   Return false to abort start up */
     bool startHook() {
       // ...
         return true;
     }
     bool startHook() {
         // ...
           return true;
       }
     / * This function is called by the Execution Engine */
     void updateHook() {
        // Your component's algorithm/code goes in    here.
       }

/ * This function is called when the task is stopped.*/
    void stopHook() {
        // Your stop code after last updateHook()
       }
    /* This function is called when the task is being
  deconfigured */
     void cleanupHook() {
        // Your configuration cleanup code
       }
 };
```

Logiciel embarqué

Fichier de définition de l'architecture logicielle

Composants, paramètres, répartition sur processeurs,

Composants OROCOS



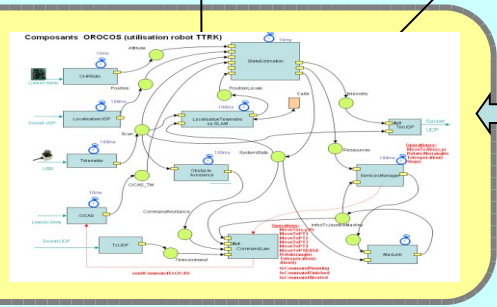
Déploiement en local

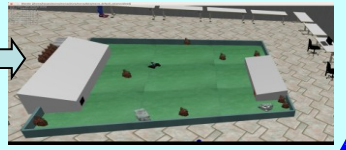
Déploiement sur système embarqué

OU

Deployer Linux

scripts



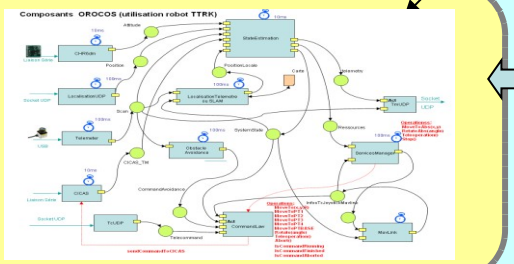


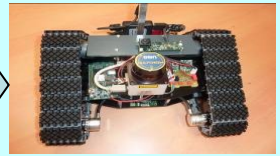
Simulation TR MORSE

PC Linux

Deployer xenomai

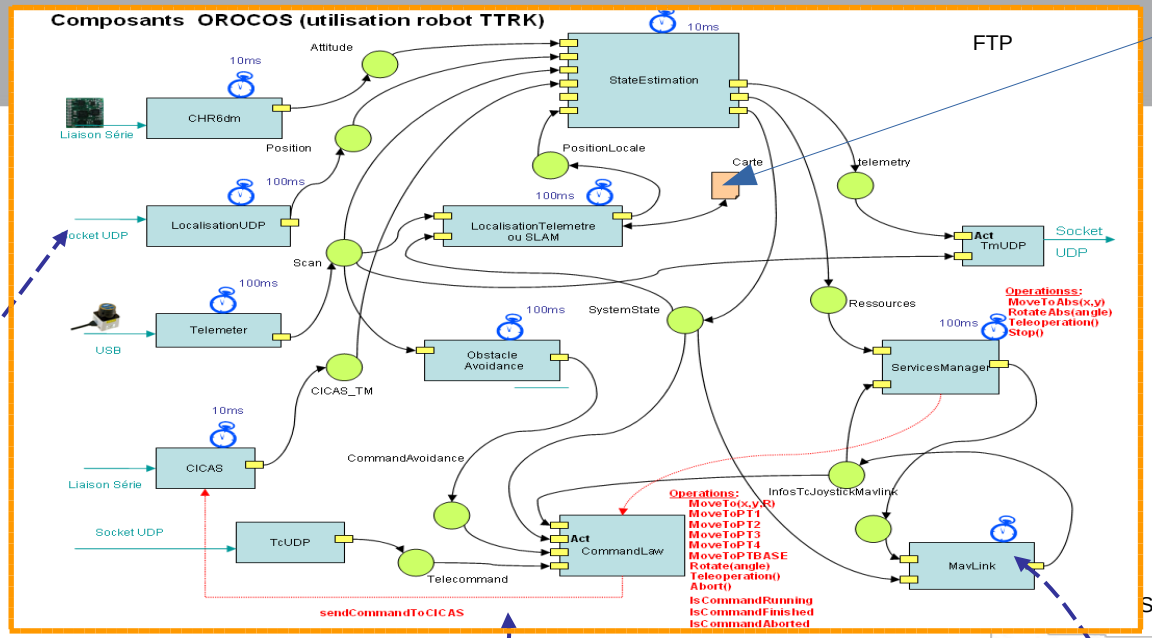
scripts

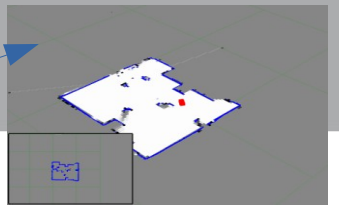




Gumstix

Configuration avec robot TTRK





Visu Carte

WebCAM

Socket UDP

Position locale

Socket UDP TM-TC

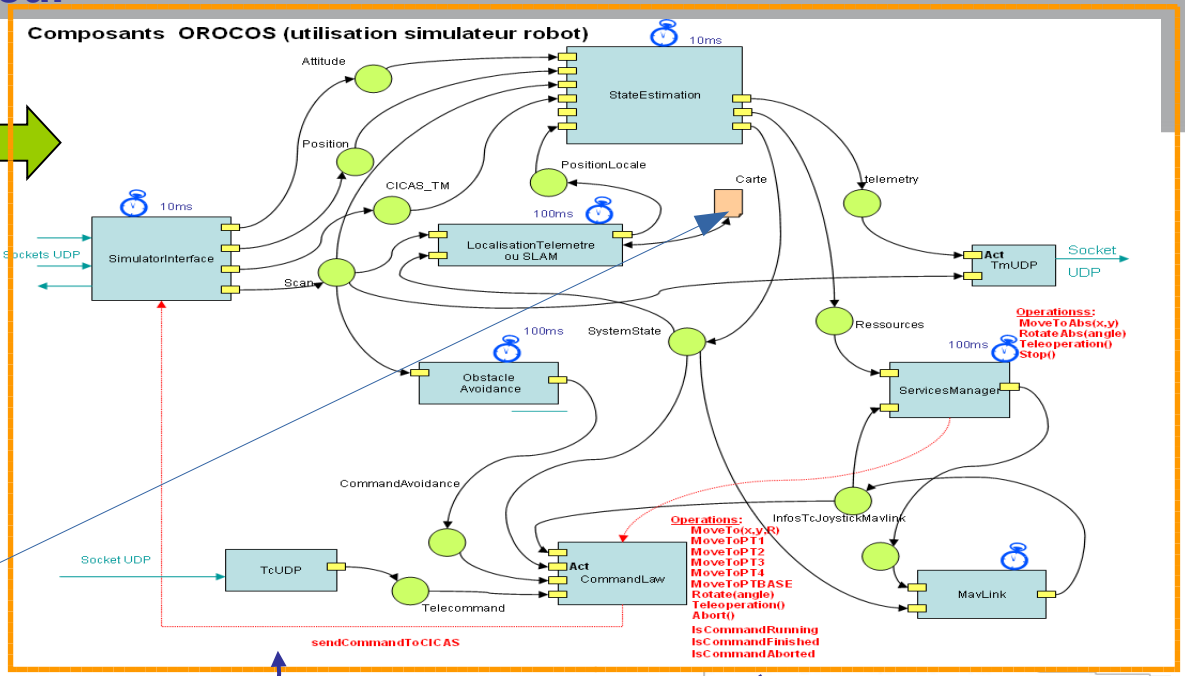
telnet, ftp

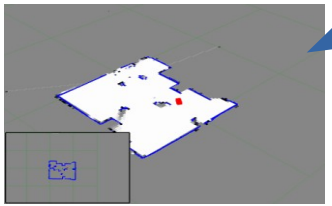
PC Localisation

Poste Développement

Station sol

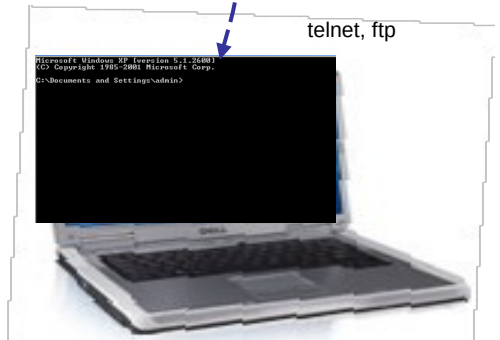
Configuration avec simulateur





Visu Carte

telnet, ftp



Poste Développement

Station sol

ROS

What is ROS ?

Ros distros Release Schedule

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Melodic Morenia (**Recommended**)	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015

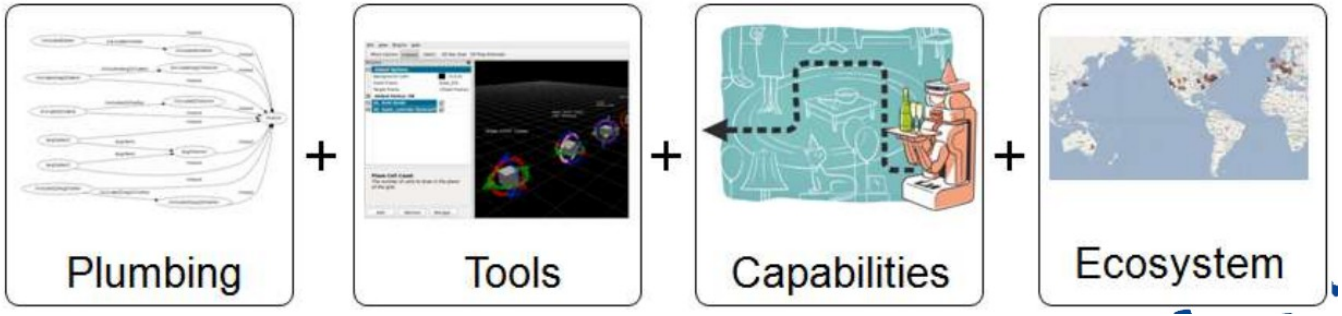
Release rules:

- There is a ROS release every year in May.
- Releases on even numbered years will be a LTS release, supported for five years.
- Releases on odd numbered years are normal ROS releases, supported for two years.
- ROS releases will drop support for EOL Ubuntu distributions, even if the ROS release is still supported.

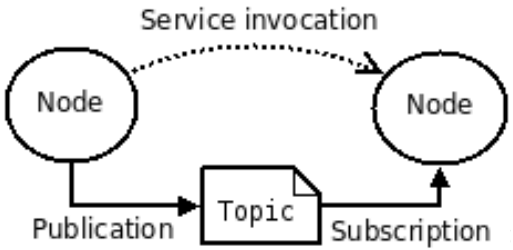
For more details see the official Release Policy

What is ROS

- Open source (BSD)
- Created by Willow Garage
- Maintained by Open Source Robotics Foundation (OSRF)

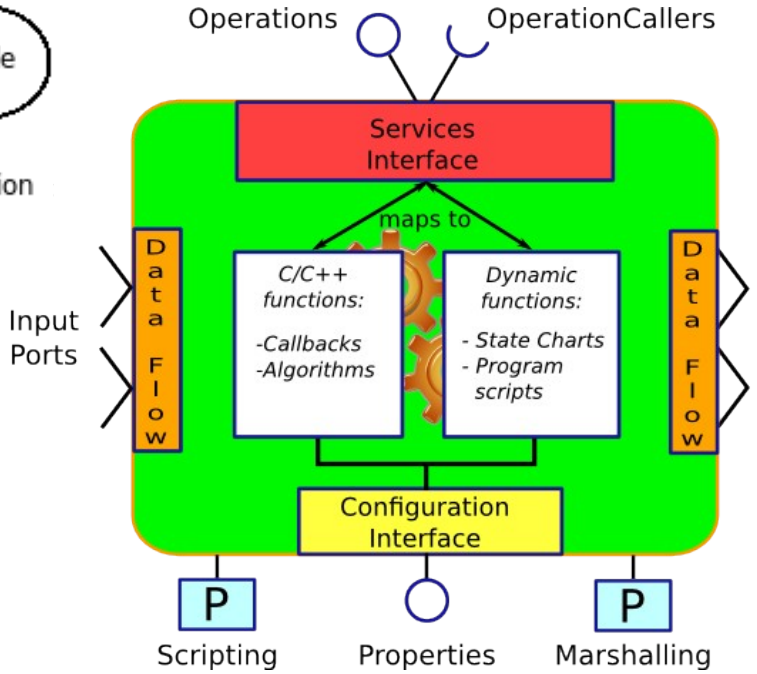


What is ROS



« Services »

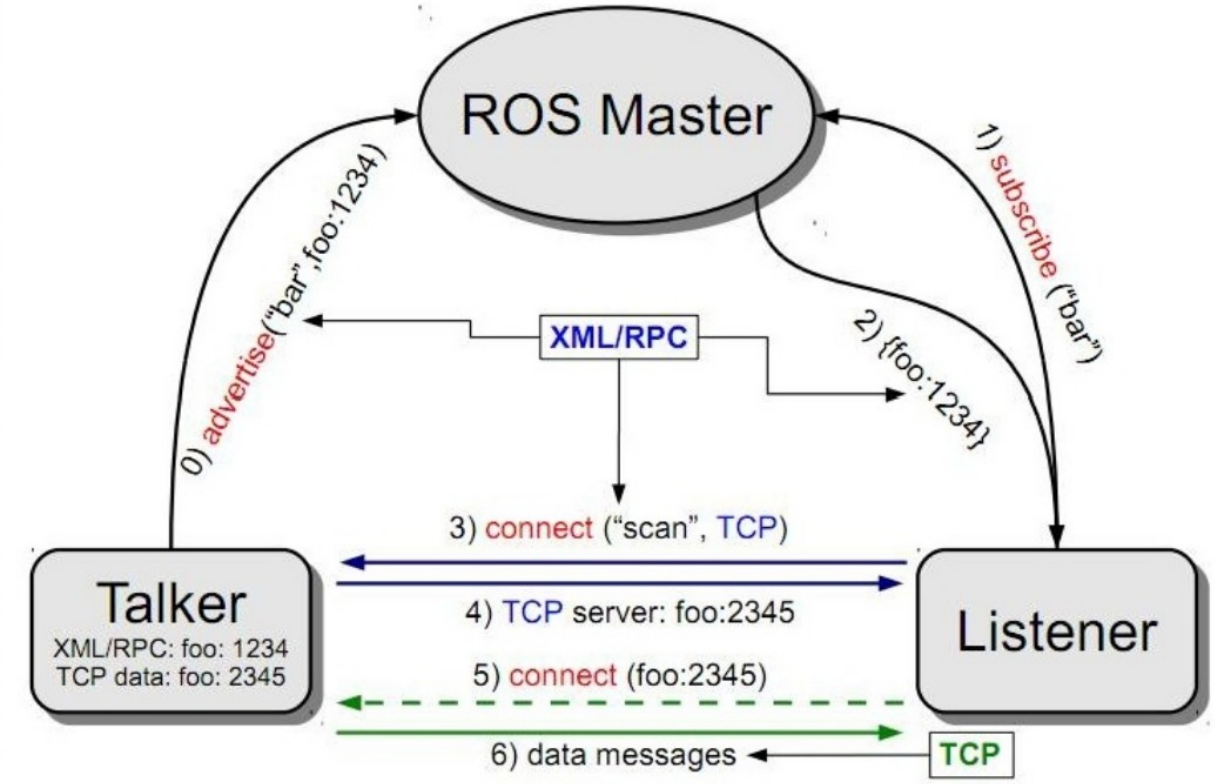
Topics
Type.msg



« Topics »
Type.msg

« params »

What is ROS


ROS Master
0) advertise("bar",foo:1234)
1) subscribe ("bar")
XML/RPC
2) {foo:1234}
3) connect ("scan", TCP)
4) TCP server: foo:2345
5) connect (foo:2345)
6) data messages
TCP
Talker
XML/RPC: foo: 1234
TCP data: foo: 2345
Listener

rosnode

rosnode is a command-line tool for printing information about ROS Nodes.

Commands:

rosnode ping test connectivity to node
rosnode list list active nodes
rosnode info print information about node
rosnode machine list nodes running on a particular machine or list machines
rosnode kill kill a running node
rosnode cleanup purge registration information of unreachable nodes

Type rosnode <command> -h for more detailed usage, e.g. 'rosnode ping -h'

rostopic

rostopic is a command-line tool for printing information about ROS Topics.

Commands:

```
rostopic bw     display bandwidth used by topic
rostopic delay  display delay of topic from timestamp in header
rostopic echo   print messages to screen
rostopic find   find topics by type
rostopic hz     display publishing rate of topic
rostopic info   print information about active topic
rostopic list   list active topics
rostopic pub    publish data to topic
rostopic type   print topic or field type
```

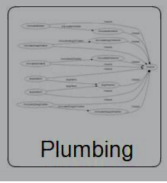
Type rostopic <command> -h for more detailed usage, e.g. 'rostopic echo -h'

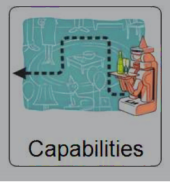
rosparam

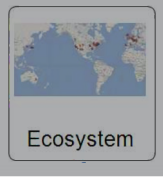
rosparam is a command-line tool for getting, setting, and deleting parameters from the ROS Parameter Server.

Commands:

- rosparam set set parameter
- rosparam get get parameter
- rosparam load load parameters from file
- rosparam dump dump parameters to file
- rosparam delete delete parameter
- rosparam list list parameter names





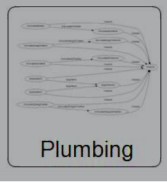


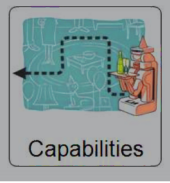
rosservice

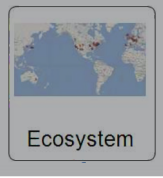
Commands:

rosservice args print service arguments
rosservice call call the service with the provided args
rosservice find find services by service type
rosservice info print information about service
rosservice list list active services
rosservice type print service type
rosservice uri print service ROSRPC uri

Type rosservice <command> -h for more detailed usage, e.g. 'rosservice call -h'







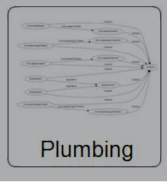
rosbag

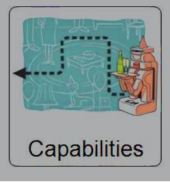
Usage: rosbag <subcommand> [options] [args]

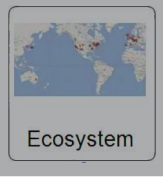
A bag is a file format in ROS for storing ROS message data.
The rosbag command can record, replay and manipulate bags.

Available subcommands:

Check	… Compress … Decompress ...
filter	Filter the contents of the bag.
fix	Repair the messages in a bag file so that it can be played in the current system.
help	
info	Summarize the contents of one or more bag files.
play	Play back the contents of one or more bag files in a time-synchronized fashion.
record	Record a bag file with the contents of specified topics.
reindex	Reindexes one or more bag files.

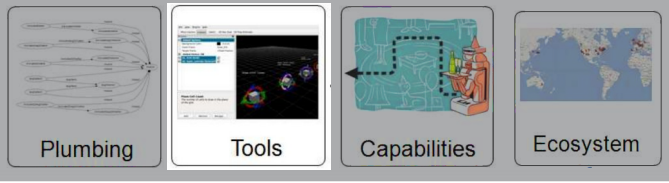






Ros...

rosawesome rosdep roslaunch-complete rospd
rosbag rosdep-source roslaunch-deps rospython
rosboost-cfg rosdistro_build_cache roslaunch-logs rosrun
roscat rosdistro_freeze_source roslocate rosservice
roscd rosdistro_migrate_to_rep_141 rosls rossrv
rosclean rosdistro_migrate_to_rep_143 rosmake rosstack
rosco rosdistro_reformat rosmaster rostest
rosconsole rosed rosmsg rostopic
roscore rosgraph rosmsg-proto rosunit
roscp rosinstall rosnode rosversion
roscreate-pkg rosinstall_generator rospack rosws
rosd roslaunch rosparam roswtf



Rqt...

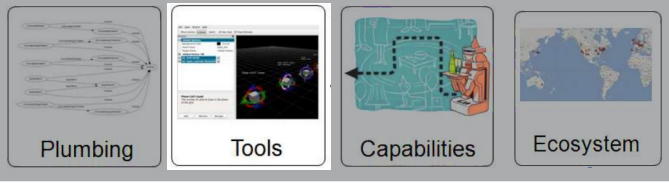
rqt is a Qt-based framework for GUI development for ROS. It consists of three parts/metapackages

rqt

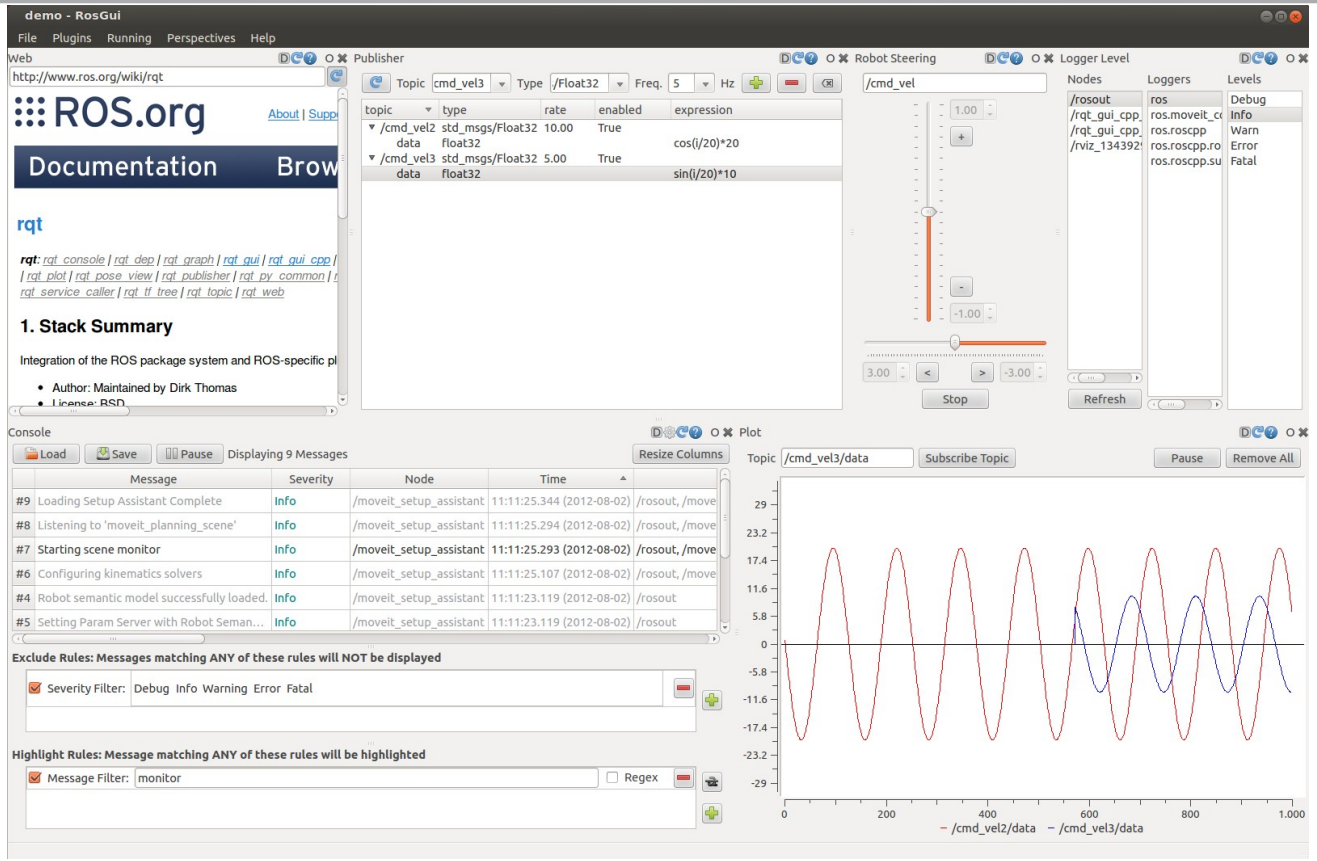
rqt_common_plugins - ROS backend tools suite that can be used on/off of robot runtime.

rqt_robot_plugins - Tools for interacting with robots during their runtime.

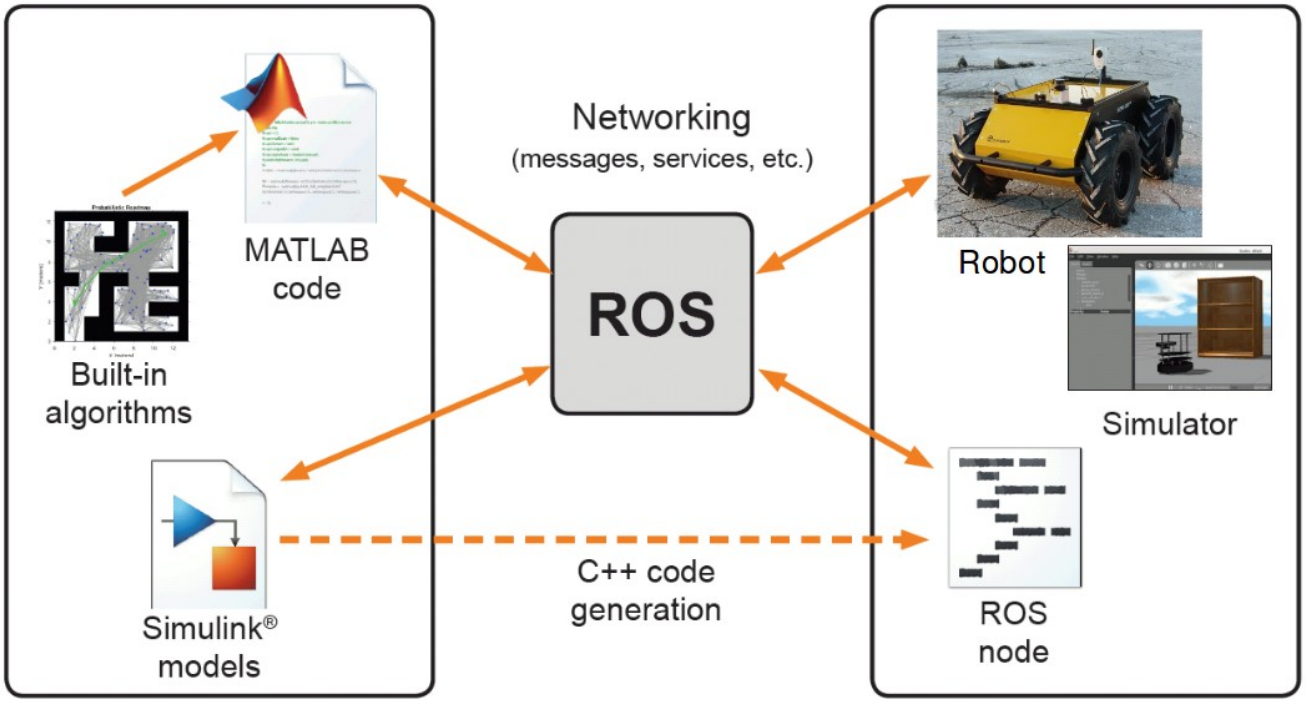
rqt metapackage provides a widget rqt_gui that enables multiple `rqt` widgets to be docked in a single window.


Plumbing
Tools
Capabilities
Ecosystem

Rqt...


demo - RosGui
File Plugins Running Perspectives Help
Web
http://www.ros.org/wiki/rqt
ROS.org
Documentation
rqt
1. Stack Summary
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
Stop
Logger Level
Nodes Loggers Levels
/rosout ros Debug
Info
Warn
Error
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
/cmd_vel2/data
/cmd_vel3/data

Matlab Robotic Toolbox

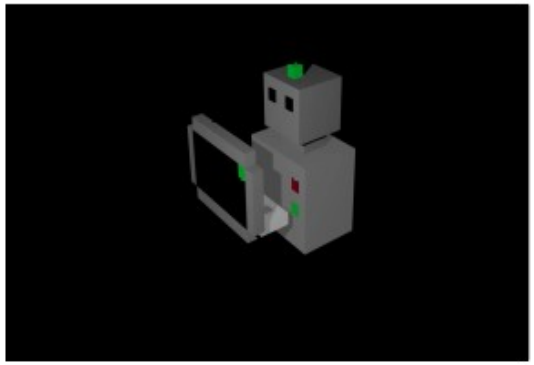


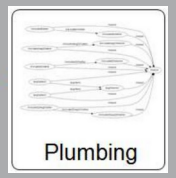
ROS interface Simulators

AIRSIM

GAZEBO

MORSE

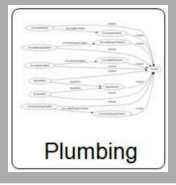




(c++) Talker node (python)

```cpp
#include "ros/ros.h"
#include "std_msgs/String.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "talker");
  ros::NodeHandle nh;

  ros::Publisher chatter_pub = nh.advertise<std_msgs::String>("chatter", 1000);

  ros::Rate loop_rate(10);
  while (ros::ok())
  {
    std_msgs::String msg;
    msg.data = "hello world";

    chatter_pub.publish(msg);

    ros::spinOnce();

    loop_rate.sleep();
  }

  return 0;
}
```

```python
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

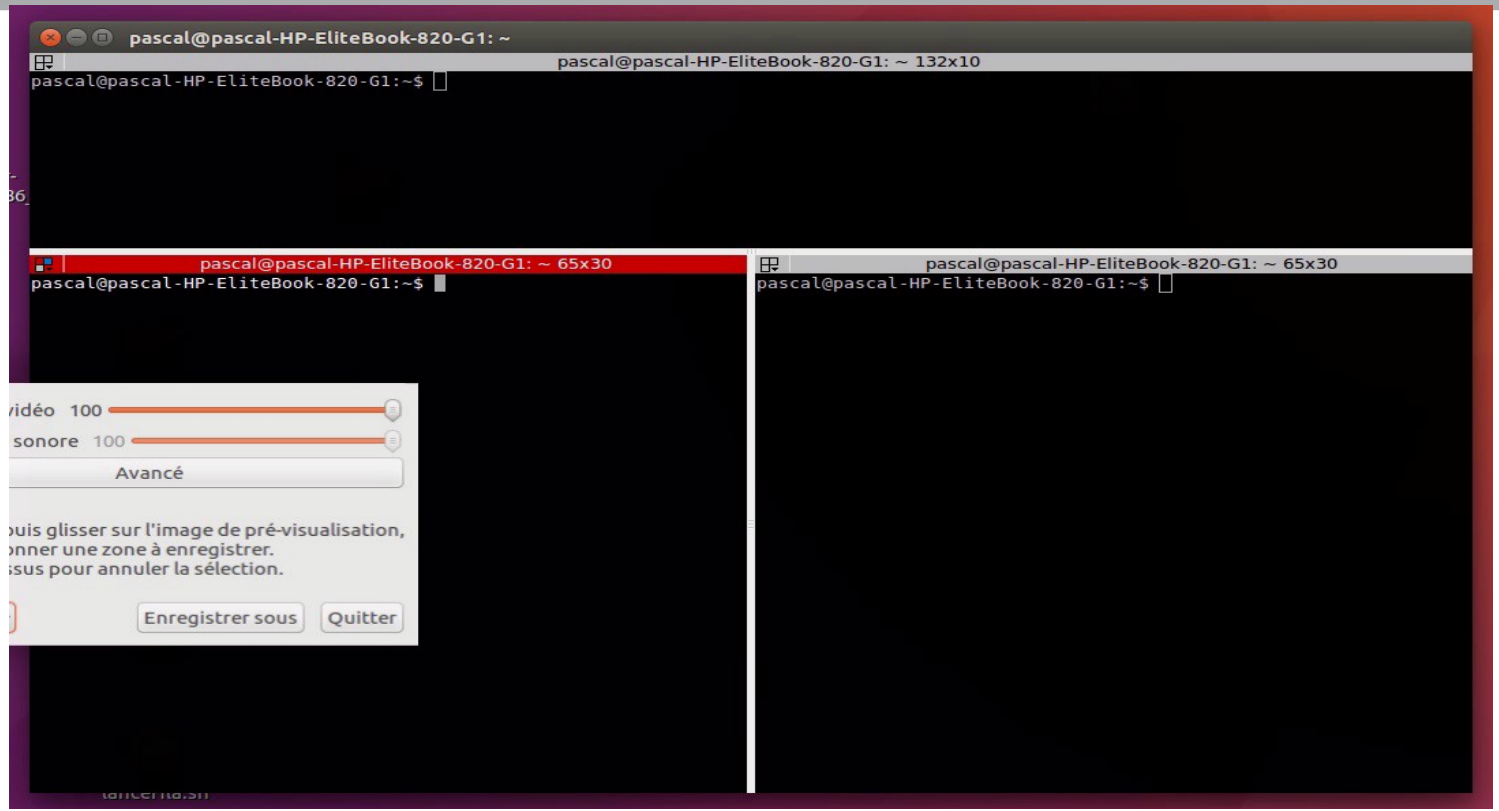


(c++) Listener node (python)

```cpp
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "listener");
  ros::NodeHandle nh;

  ros::Subscriber sub = nh.subscribe("chatter", 1000, chatterCallback);

  ros::spin();

  return 0;
}
```

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

Talker Listener



Let’s go to demo !!!