PSI 3442 & Skyrats apresentam:

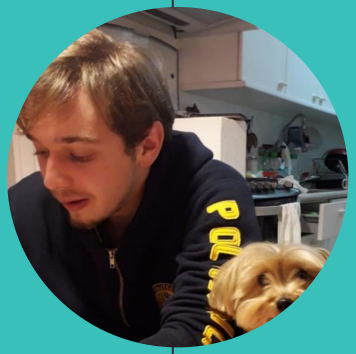
ROS

Robot Operating System

28/08/2020

José Colombini

Tiago Takeda



Emanuel Iwanow

O que veremos hoje?

- Arquitetura do ROS
- ROS master, nodes e tópicos
- Comandos do Console
- Estrutura de Pacotes
- Ambientes de trabalho catkin
- Launch files - Arquivos de inicialização

1 O que é ROS?

“Robot Operating System (ROS, sistema operacional de robôs) é uma coleção de frameworks de software para desenvolvimento de robôs, que fornece a funcionalidade de um sistema operacional em um cluster de computadores heterogêneo. ROS fornece serviços padrões de sistema operacional, tais como abstração de hardware, controle de dispositivos de baixo nível, a implementação de funcionalidades comumente usadas, passagem de mensagens entre processos e gerenciamento de pacotes.”

ROS

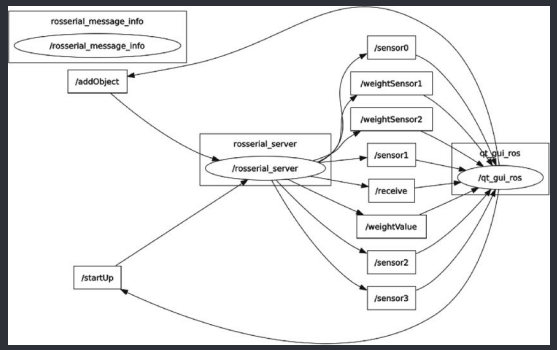
Robot Operating System

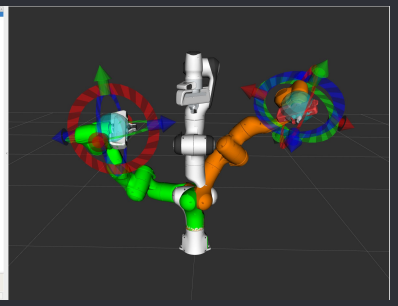
Canalização + Ferramentas + Ecossistema

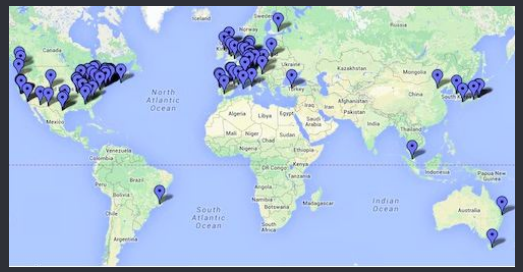
- Manejo de processos
- Intercomunicação de processos
- Drivers de dispositivos

- Simulações
- Visualizações
- Interface gráfica
- Registro de dados

- Organização de pacotes
- Distribuição de softwares
- Documentações
- Tutoriais







Filosofia do ROS

- Peer to Peer (Ponto a Ponto)

  Programas se comunicam entre si (ROS messages, services, etc)

- Possível distribuição

  Programas podem rodar em múltiplos computadores e se comunicarem por meio da rede

- Multilinguismo

  Módulos de ROS podem ser escritos em qualquer linguagem de programação contanto que a biblioteca permita. (C++, Python, MATLAB, Java, etc)

- Pouco pesado

  Bibliotecas sozinhas podem ser encapsulados por uma pequena camada de ROS

- Open-Source

- Como usaremos ROS na disciplina?

- MAVROS
- Gazebo
- Drivers da PX4

GAZEBO

Como usar o ROS?

Como funciona?

Como programar?

Setup inicial do ROS

- Para possuir as variáveis do ROS em seu terminal é necessário rodar o seguinte script em todo novo terminal:

```
> source /opt/ros/<ros_distro>/setup.bash
```

- Para não ter que rodar esse script toda vez que um terminal é inicializado, rode:

```
> echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

- Isso adiciona uma linha de código no script que é compilado todo novo terminal

ROS Master

- Gerencia a comunicação entre nodes (nó)
- Cada node se registra com o master em sua inicialização
- Como iniciar o ROS Master:

```
> roscore
```

ROS Master

ROS Nodes

- Programas executáveis de propósito único
- São individualmente compilados, executados e gerenciados
- Organizados em pacotes

Inicialize um node com:

```
> rosrun package_name node_name
```

Veja os nodes ativos com:

```
> rosnode list
```

Veja as informações do node com:

```
> rosnode info node_name
```

ROS Topics

- Nodes se comunicam por tópicos
- Um node pode publicar (publish) ou se inscrever (subscribe) em um tópico
- Tópico é um nome corrente de mensagens

Veja os topicos ativos com:

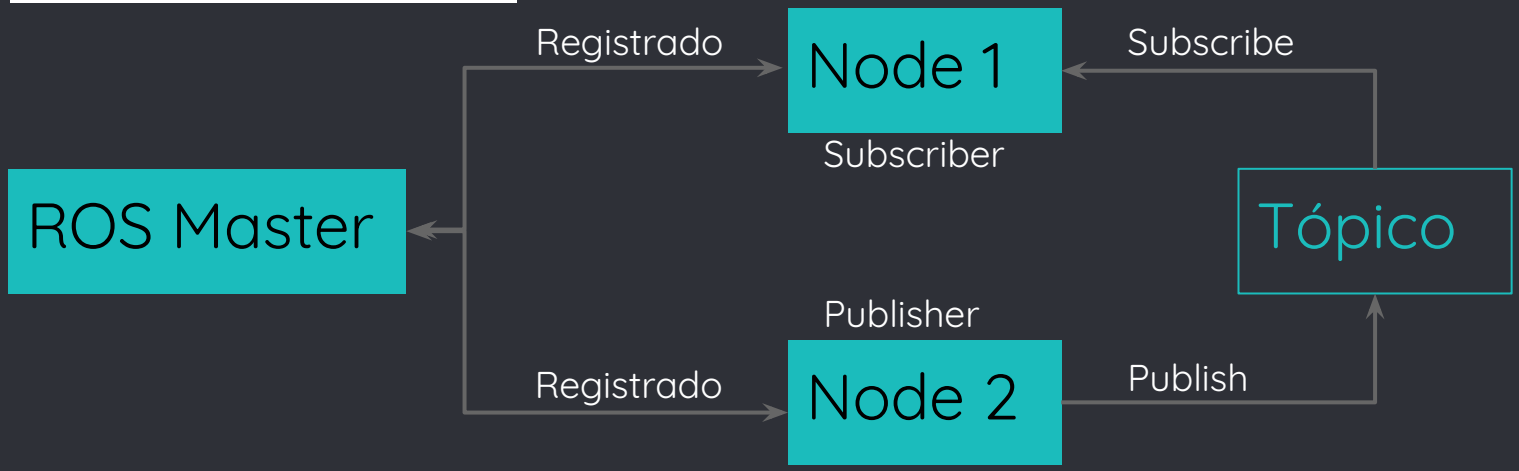
```
> rostopic list
```

Imprimir os dados de um tópico com:

```
> rostopic echo /topic
```

Veja as informações do tópico com:

```
> rostopic info /topic
```



ROS Messages

- Estrutura de dado que define o tipo de tópico
- Comprometido com uma estrutura hierarquizada de inteiros, valores flutuantes, booleanos, etc.
- Definidos no arquivo *.msg

Veja o tipo de tópico com:

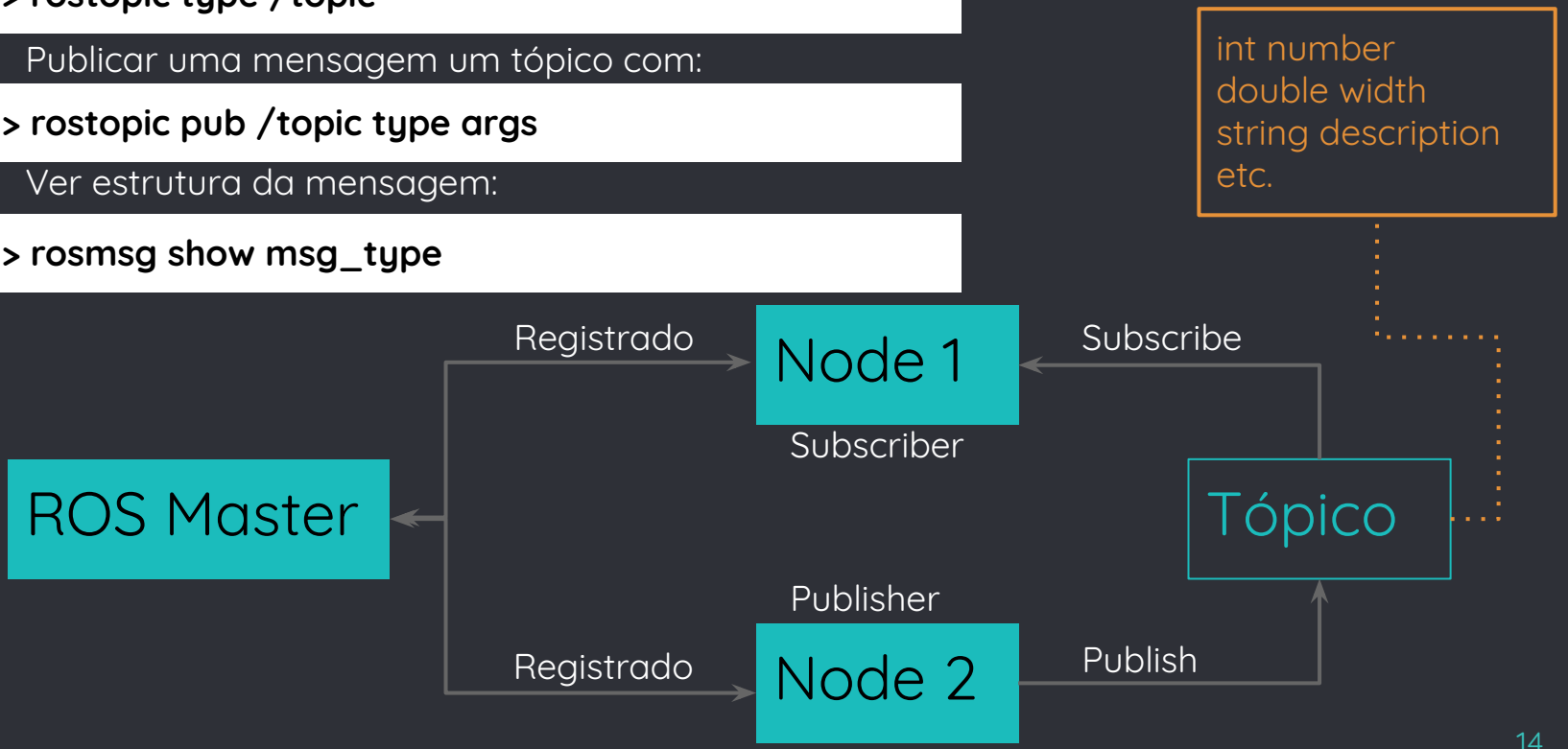
```
> rostopic type /topic
```

Publicar uma mensagem um tópico com:

```
> rostopic pub /topic type args
```

Ver estrutura da mensagem:

```
> rosmsg show msg_type
```



Exemplos

Inicie o ROS Master:

```
> roscore
```

Em outro terminal inicie o node TurtleSim:

```
> rosrun turtlesim turtlesim_node
```

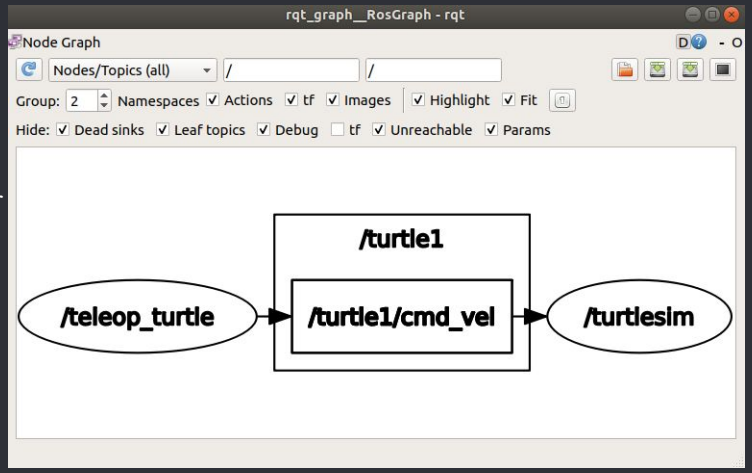
Em outro terminal inicie o node turtle_teleop_key:

```
> rosrun turtlesim turtle_teleop_key
```

Agora você pode movimentar a tartaruga pelas setas do teclado.

Uma ferramenta importante para análise dos processos é o rqt_graph, que demonstra os nodes e o tópicos graficamente. Para iniciá-lo basta rodar em outro terminal:

```
> rosrun rqt_graph rqt_graph
```



Ambientes/Workspaces do ROS

- catkin build Systems
- Catkin é o sistema de configuração do ROS que gera executáveis, bibliotecas e interfaces
- Sugerimos a utilização do Catkin Command Line Tools

Use “catkin build” em vez de “catkin_make”

```
> sudo apt-get install python-catkin-tools
```

Crie um diretório para seu workspace:

```
> mkdir -p ~/catkin_ws/src
```

Vá para o diretório do seu ambiente catkin com:

```
> cd ~/catkin_ws
```

Para iniciar o workspace:

```
> catkin init
```

Para construir (“build”) um pacote:

```
> catkin build package_name
```

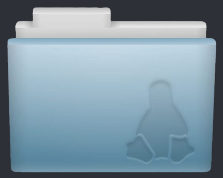
Toda vez que você constrói um pacote é necessário é necessário atualizar o seu terminal para o ROS achar seus pacotes.

```
> source ~/catkin_ws/devel/setup.bash
```

Ambientes/Workspaces do ROS

- Um workspace catkin possui as seguintes pastas:

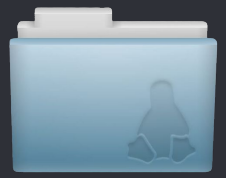
Trabalhe aqui



src

A pasta source contém a fonte dos códigos. Nela você pode clonar, criar e editar os pacotes.

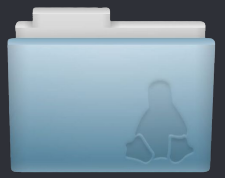
Não mexer



build

A pasta build é onde o CMake é chamado para construir os pacotes da pasta source. Informações de cache e outros arquivos intermediários são guardados aqui.

Não mexer



devel

A pasta development (devel) é onde os pacotes são colocados antes de serem instalados.

ROS Packages

- ROS é organizado em pacotes
- Forma de modularizar pedaços de código com funções diferentes
- Podem conter nodes, messenges, configs, libraries, drivers, etc.
- Possuem:
  - Um arquivo `CMakeLists.txt` .
  - Um arquivo `package.xml`
  - Uma pasta `src`
  - Uma pasta `scripts`
  - Uma pasta `include/package_name`
  - Pastas `msg` e `srv`
  - Uma pasta `launch` para guardar os arquivos launchfiles

Criando um Package

- Para criar um pacote vazio, basta:

  Ir até o diretório source do seu workspace:

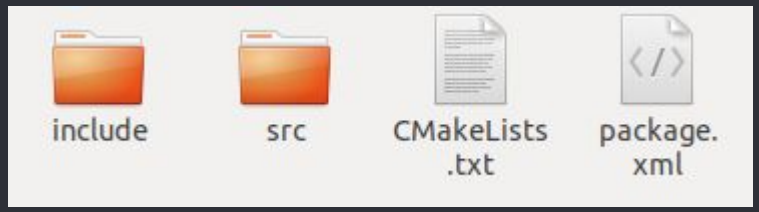
```
> cd ~/catkin_ws/src
```

  Utilizar o comando:

```
> catkin_create_pkg package_name dep1 dep2 dep3
```

  Esse comando já irá criar um pacote que depende das dependências citadas no comando de inicialização (dep1, dep2, dep3)

```
File  Edit  View  Search  Terminal  Help
iwa9@iwa-DELL:~$ cd teste3_ws/src/
iwa9@iwa-DELL:~/teste3_ws/src$ catkin_create_pkg teste roscpp rospy std_msgs
Created file teste/package.xml
Created file teste/CMakeLists.txt
Created folder teste/include/teste
Created folder teste/src
Successfully created files in /home/iwa9/teste3_ws/src/teste. Please adjust the
values in package.xml.
```



Roslaunch

- O roslaunch é uma ferramenta que serve para rodar diversos nodes ROS ao mesmo tempo, setar parâmetros no Parameter Server do ROS e rodar outros arquivos launch

```
<?xml version="1.0"?>
<launch>

    <node pkg="your_package" name="josé" type="your_node"/>

    <arg name="x" default="0"/> <!--Seta o parametro x como 0-->

    <include file="$(find outropackage)/launch/file2.launch"> <!--lansa o outro arquivo-->
        <arg name="y" value="1"/> <!-- y = 1 -->
    </include>
</launch>
```

Como usar:

```
> roslaunch package_name file.launch
```

Obs: O roslaunch jah inicializa o Ros master, caso ele ainda não tenha sido inicializado.

Conclusões e próximos passos

- Prática leva a perfeição!

- Simular o drone
  - Pacote do Firmware da px4

Obrigado!

Questões?

Podem entrar em contato com os monitores pelo grupo da disciplina, pelo inbox ou pelo moodle.