# Exemplos ROS2 passo a passo

Pré-requisitos:

- ROS2
- Python 3

## Passo a passo

Dê source na sua instalação do ROS2:

```
source /opt/ros/foxy/setup.bash
```

Crie um diretório, o workspace, que servirá de base para os pacotes, e entre nele:

```
mkdir -p ~/workspace/src
cd ~/workspace/src
```

Ainda nesse diretório, crie um pacote novo, com um nó

```
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

Para certificar-se que tudo deu certo, volte para o diretório base do workspace e construa o pacote e dê source na instalação e execute:

```
cd ~/workspace
colcon build
. install/setup.bash
ros2 run my_package my_node
```

Se tudo deu certo você recebeu uma mensagem "Hi from my_package" em seu terminal.

#### Adicionando nó publisher e nó subscriber

Vá até o diretório aonde fica o seu nó de exemplo:

```
cd ~/workspace/src/my_package/my_package
```

Baixe os exemplos de nós:

```
wget
https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_publi
sher/examples_rclpy_minimal_publisher/publisher_member_function.py
wget
https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_subsc
riber/examples_rclpy_minimal_subscriber/subscriber_member_function.py
```

Volte um diretório e abra o arquivo setup.py:

```
cd ..
vim/code/emacs setup.py
```

e adicione os novos arquivos como entry points:

```
entry_points={
    'console_scripts': [
        'my_node = my_package.my_node:main',
        'talker = my_package.publisher_member_function:main',
        'listener = my_package.subscriber_member_function:main',
    ],
},
```

Volte para o diretório base do workspace, construa o pacote e dê source na instalação:

```
cd ~/workspace
colcon build
. install/setup.bash
```

Abra outro terminal, vá para o diretório base do workspace e dê source na instalação também:

```
cd ~/workspace
. install/setup.bash
```

Em um dos terminais coloque o nó de publicação para rodar

```
ros2 run my_package talker
```

E no outro um nó de inscrição

```
ros2 run my_package listener
```

Se tudo deu certo, um dos terminais envia uma mensagem e outro recebe.

## Laser simples

Agora vamos adicionar um leitor de laser, primeiro vá ao diretório com os nós e baixe o exemplo de leitura de laser:

```
cd ~/workspace/src/my_package/my_package
wget
https://gist.githubusercontent.com/tikaradate/5bfa74a4f319b13fa91c0a2f4a62a3bc/
raw/a97a89fcd0e8bc5ee8c8c77bb7efd05c6a517235/laser.py
```

Adicione ao arquivo de configuração:

```
cd ..
vim/code/emacs setup.py

entry_points={
    'console_scripts': [
        'my_node = my_package.my_node:main',
        'talker = my_package.publisher_member_function:main',
        'listener = my_package.subscriber_member_function:main',
        'laser = my_package.laser:main'
    ],
},
```

Volte para a base do workspace, construa o pacote e dê source na instalação

```
cd ~/workspace
colcon build
. install/setup.bash
```

Execute o novo nó

```
ros2 run my_package laser
```

(Queria explicar como rodar a simulação mas não lembro como instalei exatamente, na simulação ainda é imporante inserir um objeto na frente do robô para que algum dado seja lido)Execute uma simulação ou o robô real e então excute de novo o nó, se tudo deu certo você verá valores representando a distância em metros do objeto na frente dele.

# Movimentação

Igual o exemplo anterior, baixe o arquivo no diretório aonde ficam os nós

```
cd ~/workspace/src/my_package/my_package
wget
https://gist.githubusercontent.com/tikaradate/3d1727d45caff8fa13d30184cb754b27/
raw/f56983e34884aad411ed31550abaff8a12290c7c/movement.py
```

Adicione ao arquivo de configuração:

```
cd ..
vim/code/emacs setup.py
entry_points={
    'console_scripts': [
        'my_node = my_package.my_node:main',
        'talker = my_package.publisher_member_function:main',
        'listener = my_package.subscriber_member_function:main',
        'laser = my_package.laser:main',
        'movement = my_package.movement:main',
    ],
},
```

Volte ao diretório base, construa o pacote e dê source em sua

```
cd ~/workspace
colcon build
. install/setup.bash
```

Certifique-se de ter o robô, simulado ou real, ativo e execute o nó

```
ros2 run my_package movement
```

# Exemplo turtlesim

Crie um novo pacote a partir do diretório src

```
cd ~/workspace/src
ros2 pkg create --build-type ament_python --node-name turtle_node custom_turtle
```

Baixe o arquivo de auxílio de movimentação

```
cd ~/workspace/src/custom_turtle/custom_turtle
wget
https://gist.githubusercontent.com/tikaradate/2e5bd4b3cdfbc87426b3a1005cfe359e/
raw/d31880ea65bf8a1811f756995602f03d468270b9/turtle_movement.py
```

Adicione no arquivo de setup

```
cd ..
vim/code/emacs setup.py
entry_points={
    'console_scripts': [
        'turtle_node = custom_turtle.turtle_node:main',
        'turtle_movement = custom_turtle.turtle_movement:main',
    ],
},
```

Construa o pacote e dê source na instalação

```
cd ~/workspace
colcon build
. install/setup.bash
```

Em um outro terminal execute o turtlesim e nesse o nó

```
TERMINAL A
ros2 run turtlesim turtlesim_node

TERMINAL B
ros2 run custom_turtle turtle_movement
```

# Dicas

Para ver os tópicos disponívei você pode executar

```
ros2 topic list
```

ou executar, para usar uma interface gráfica

```
rqt
```