TUTORIAL DE INSTALAÇÃO DO ROS E DO REPOSITÓRIO NXT LEGO

Instalando o ROS:

1. Para instalar o ROS, abra o terminal e digite os seguintes comandos:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80
--recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

sudo apt-get update

sudo apt-get install ros-kinetic-desktop-full

sudo rosdep init

rosdep update

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

source ~/.bashrc

sudo apt install python-rosinstall python-rosinstall-generator
python-wstool build-essential
```

1.1. Para mais detalhes, acesse: http://wiki.ros.org/kinetic/Installation/Ubuntu.
Observação: o comando `sudo apt-get install ros-kinetic-desktop-full` *pode demorar um pouco.*

2. Instale catkin_tools. Ainda no terminal, digite:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
`lsb_release -sc` main" >
/etc/apt/sources.list.d/ros-latest.list'

wget http://packages.ros.org/ros.key -O - | sudo apt-key add -

sudo apt-get update

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

3. Para começar a trabalhar com ROS, é preciso criar uma workspace:

```
mkdir -p ~/[nome_da_workspace]/src

cd ~/[nome_da_workspace]

catkin build
```

Substitua `[nome_da_workspace]` pelo nome que deseja dar à sua workspace (usualmente, tem o nome de *catkin_ws*). Workspace criada!

Instalando o Repositório NXT Lego:

4. Antes de baixar o repositório NXT Lego, é necessário instalar alguns pacotes. Dê os seguintes comandos no terminal:

```
sudo apt-get install ros-kinetic-robot-pose-ekf
sudo apt-get install ros-kinetic-joy
sudo apt-get install ros-kinetic-libcreate
```

5. Uma vez que esses 3 comandos foram dados, podemos prosseguir com a instalação do repositório Lego NXT:

```
cd ~/[nome_da_workspace]/src
git clone -b master https://github.com/h3ct0r/ros_lego_nxt
cd ~/[nome_da_workspace]
catkin build
```

6. Para conectar o robô com o computador é preciso mudar certas permissões de USB. No terminal, digite:

```
sudo groupadd lego
sudo usermod -a -G lego $(id -un)

echo SUBSYSTEM==”usb”, ATTR{idVendor}==”0694”, MODE:=”0666”, OWNER:=”[nome_do_usuario]”, GROUP:=”lego”> /tmp/70-legou.rules && sudo mv /tmp/70-legou.rules /lib/udev/rules.d/99-legou.rules
```

Substitua **[nome_do_usuario]** pelo nome do seu usuário.

7. A seguir, é preciso instalar certos arquivos. Para instalar 7.1 e 7.2, entre no diretório deles e digite no terminal, já o 7.3 leia seu README contém as instruções para instalação:

```
sudo python setup.py install
```

7.1. Versão 2.2.2 nxt-python para **python 2.7**. Use o link: https://pypi.org/project/nxt-python/#files

7.2. PyUSB: https://github.com/pyusb/pyusb

7.3. http://www.linuxfromscratch.org/blfs/view/6.3/general/libusb.html

Agora reinicie seu computador ou faça Log Out.

Configurando o robô:

8. Antes de iniciar o robô, é preciso configurar as suas portas. Abra o diretório `~/[nome_da_workspace]/src/ros_lego_nxt/nxt_robots/nxt_robots_sensor_car/config`. Abra o arquivo robot.yaml, é neste local em que as portas são setadas. Comente os blocos que não estão sendo usados e coloque as portas da maneira como o robô se encontra montado, como por exemplo:

```
nxt_robot:
  - type: motor
    name: r_wheel_joint
    port: PORT_A
    desired_frequency: 20.0

  - type: motor
    name: l_wheel_joint
    port: PORT_C
    desired_frequency: 20.0

#  - type: touch
#    name: touch
#    frame_id: r_touch_link
#    port: PORT_1
#    desired_frequency: 1.0

  - type: ultrasonic
    frame_id: ultrasonic_link
    name: ultrasonic_sensor
    port: PORT_2
    spread_angle: 0.05
    min_range: 0.01
    max_range: 2.5
    desired_frequency: 4.0

  - type: sound
    frame_id: sound_link
    name: sound
    port: PORT_3
    adjusted: 1
    desired_frequency: 4.0

  - type: light
    frame_id: light_link
    name: light
    port: PORT_4
    illuminated: 1
    desired_frequency: 4.0
```

9. Com as configurações corretas, já é possível iniciar o robô com o seguinte comando:

```
roslaunch nxt_robots_sensor_car robot.launch
```

roslaunch é uma ferramenta para facilmente iniciar diversos nós ao mesmo tempo.

Realizando ajuste de velocidade:

10. Do jeito que está, o robô se movimenta de maneira muito rápida e abrupta, para melhorar isso, iremos alterar os comandos de velocidade. Abra o arquivo base_controller.py, no diretório */catkin_ws/src/nxt/nxt_controllers/src/nxt_controllers*. A partir da linha 81, faça as seguintes mudanças (destacadas em vermelho):

```
# velocity commands
        vel_trans = (self.vel_trans_desi)*0.3 + self.k_trans*
(self.vel_trans_desi - self.vel_trans)*0.0
        vel_rot = (self.vel_rot_desi)*1.75 +
self.k_rot*(self.vel_rot_desi - self.vel_rot)*0.0

# wheel commands
        l_cmd = JointCommand()
        l_cmd.name = self.l_joint
        l_cmd.effort = 0.45*self.vel_to_eff*
(vel_trans/self.wheel_radius -
vel_rot*self.wheel_basis/self.wheel_radius)
        self.pub.publish(l_cmd)

        r_cmd = JointCommand()
        r_cmd.name = self.r_joint
        r_cmd.effort = 0.45*self.vel_to_eff*
(vel_trans/self.wheel_radius +
vel_rot*self.wheel_basis/self.wheel_radius)
        self.pub.publish(r_cmd)
```

Realizando ajustes de odometria:

11. Para ajustes de odometria, será preciso alterar os parâmetros do robô em 3 arquivos diferentes:

    base_controller.py - */catkin_ws/src/nxt/nxt_controllers/src/nxt_controllers*

    base_odometry.py - */catkin_ws/src/nxt/nxt_controllers/src/nxt_controllers*

    robot.launch - */catkin_ws/src/nxt_robots/nxt_robots_sensor_car/launch*

Nesses arquivos encontre as linhas de código:

```
                    self.wheel_radius   =   rospy.get_param(self.ns
+'wheel_radius', raio_roda)
         self.wheel_basis = rospy.get_param(self.ns +'wheel_basis',
dist_rodas)
```

raio_roda: Raio da roda;
dist_rodas: Metade da distância entre as rodas;

Faça as medições do robô (em metros) e as insira no código.

12. Por fim, digite no terminal:

```
catkin build
```

Controlando o robô com o teclado:

13. Vamos agora controlar o robô via teclado com o seguinte comando:

```
roslaunch nxt_teleop teleop_keyboard.launch
```

Isso fará com que o nó *nxt_teleop* publique mensagens de velocidade no tópico */cmd_vell*.

14. Mas antes vamos precisar mudar alguns parâmetros do pacote *nxt_teleop*, que contém os códigos usados para teleoperação do robô, mais especificamente no arquivo *nxt_key.cpp*. Ele provavelmente estará no diretório: `~/[nome_da_workspace]/src/ros_lego_nxt/nxt_teleop/src`. Modifique as linhas 148 a 166 que inicialmente estão da seguinte maneira:

```
switch(c)
  {
          case KEYCODE_L:
          ROS_DEBUG("LEFT");
          angular_ = 0.55;
          break;
          case KEYCODE_R:
          ROS_DEBUG("RIGHT");
          angular_ = -0.55;
          break;
          case KEYCODE_U:
          ROS_DEBUG("UP");
          linear_ = 0.1;
          break;
          case KEYCODE_D:
          ROS_DEBUG("DOWN");
          linear_ = -0.1;
          break;
  }
```

As linhas destacadas são as que definem o quanto de velocidade será mandada para o tópico */cmd_vel*. Com estes valores, o robô não se movimentará devido às modificações feitas para melhorar o controle de velocidade. Altere esses valore para que o robô tenha uma movimentação aceitável (para as velocidades lineares, |0.9| é bom, já para as angulares, |0.80|).

15. Agora podemos usar o comando:

```
roslaunch nxt_teleop teleop_keyboard.launch
```

Controlando o robô com um Joystick:

16. Observe em qual porta o joystick foi conectado, para isso, digite no terminal:

```
cd /dev/input

ls
```

Veja qual o nome que aparecerá ao conectar o joystick (exemplo `js1`).

17. Vá para a pasta: */catkin_ws/src/nxt_teleop/nxt_teleop/launch* e abra o arquivo telop_joy.launch. Na linha 6, mude ***porta_joystick*** para porta do controle.

```
<param name="dev" type="string" value="/dev/input/ porta_joystick" />
```

```
1 <launch>
2
3  <!-- joy node -->
4   <node respawn="true" pkg="joy"
5         type="joy_node" name="nxt_joy" >
6     <param name="dev" type="string" value="/dev/input/js0" />
7     <param name="deadzone" value="0.12" />
8     <param name="autorepeat_rate" value="10.0" />
9   </node>
10
11
12  <!-- teleop node -->
13   <node pkg="nxt_teleop" type="nxt_teleop_joy_node" name="nxt_teleop_joy_node" output="screen">
14     <param name="axis_linear" value="3" type="int"/>
15     <param name="axis_angular" value="0" type="int"/>
16     <param name="axis_deadman" value="4" type="int"/>
17     <param name="scale_linear" value="1" type="double"/>
18     <param name="scale_angular" value="1" type="double"/>
19   </node>
20
21 </launch>
```

Caso no passo 16. o nome que tiver aparecido for `js0`, não será preciso mudar nada, uma vez que a porta que já vem no código é `js0`.

18. Ainda neste mesmo arquivo, mude a linha 14 para:

```
<param name="axis_linear" value="1" type="int"/>
```

Isso fará com que o controle de velocidade seja feito somente por meio de um analógico. *value* identifica qual botão é usado para controlar o eixo linear.

19. Inicialmente, o joystick tem um botão de enable, caso queira tira-lo, vá no arquivo *nxt_joy.cpp* no diretório */**[nome_da_workspace]**/src/nxt_teleop/nxt_teleop/src*.

```
92 void NxtTeleopJoy::publish()
93 {
94   boost::mutex::scoped_lock lock(publish_mutex_);
95   if (deadman_pressed_)
96     {
97       vel_pub_.publish(last_published_);
98     }
99
100 }
```

Na linha 95, altere a condição do if para:

```
if (1)
```

20. Agora use o comando:

```
roslaunch nxt_teleop teleop_joy.launch
```