ROS 설치 확인

$ roscore
... logging to
/home/kimsooyoung/.ros/log/5b440a74-4cda-11eb-a263-9cb6d08bf543/roslaunch-kimsooyoung-XPS-13-9370-10527.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:38251/
ros_comm version 1.14.10

SUMMARY
========

PARAMETERS
* /rosdistro: melodic
* /rosversion: 1.14.10

NODES

auto-starting new master
process[master]: started with pid [10537]
ROS_MASTER_URI=http://localhost:11311/
setting /run_id to 5b440a74-4cda-11eb-a263-9cb6d08bf543
process[rosout-1]: started with pid [10559]
started core service [/rosout]

** 확인 완료 **

turtlesim으로 확인
$ rosrun turtlesim turtlesim_node

** 새 터미널 열기 ctrl + 't'

$ rosrun turtlesim turtle_teleop_key

* 방향 키를 사용하여 거북이를 이동해봅니다.

** Bye turtle ~!! **

****
** 가제보의 설치

1. 패키지 다운로드 하기

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

2. 셋업 키

```
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

3. 가제보 인스톨 및 제거 방법

```
sudo apt-get remove .*gazebo.* && sudo apt-get update && sudo apt-get install gazebo11
```

** 이전에 gazebo가 설치되어 있다면 gazebo11가 설치되지 않습니다.
ROS를 설치하면서 같이 설치된 gazebo 구버전을 지워 주셔야 합니다.

4. 가제보 실행하기 〉〉 gazebo 입력

```
$ gazebo
```

catkin build 환경 준비

$ gedit ~/.bashrc

파일 제일 아래에 다음과 같은 내용 입력

```
-------------
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
alias cma='catkin_make -DCATKIN_WHITELIST_PACKAGES=""'
alias cop='catkin_make --only-pkg-with-deps'
alias sds='source devel/setup.bash'

source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://localhost:11311
export ROS_HOSTNAME=localhost
#export ROS_MASTER_URI=http://192.168.1.100:11311
#export ROS_HOSTNAME=192.168.1.100
```

** save ** 닫기

$ source ~/.bashrc

.bashrc 파일 수정 후에는 다시 로그인을 하거나 터미널 종료 후 다시 실행을 해야 적용됨.

source ~/.bashrc 또는 . ~/.bashrc
를 실행하면 터미널을 다시 열지 않아도 적용이 됨.

* 예제 클론

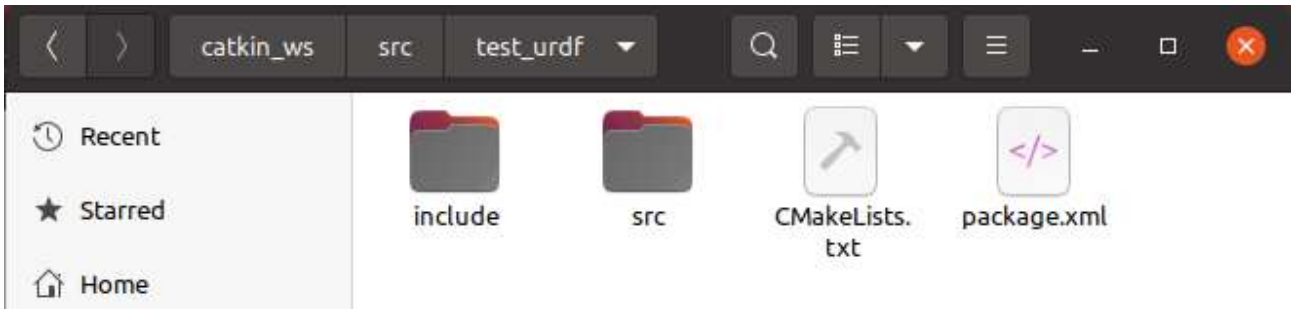
$ git clone https://github.com/Road-Balance/gcamp_ros_basic.git

_ make
$ cma

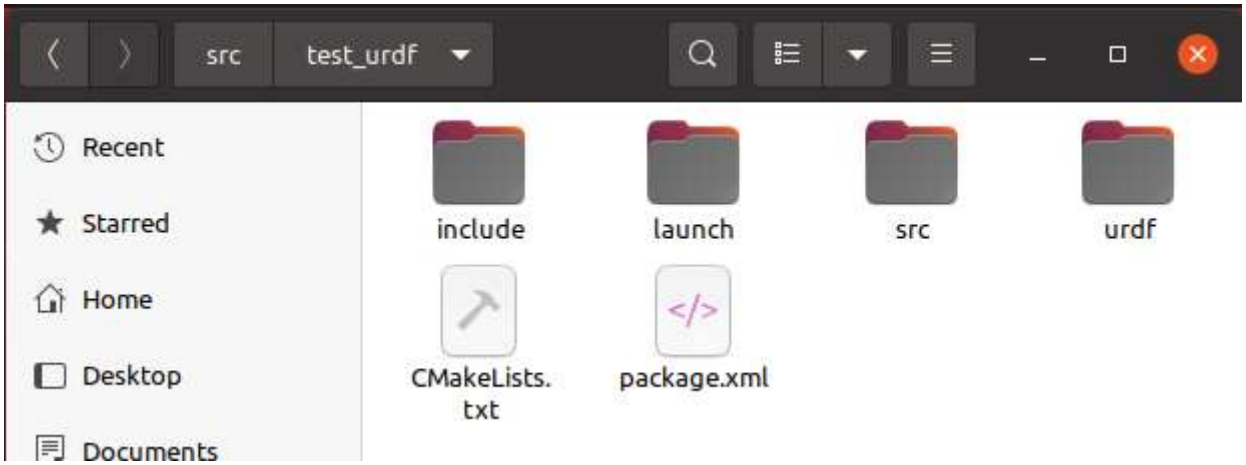
_ source
$ sds

$ roslaunch gcamp_gazebo gazebo_world.launch

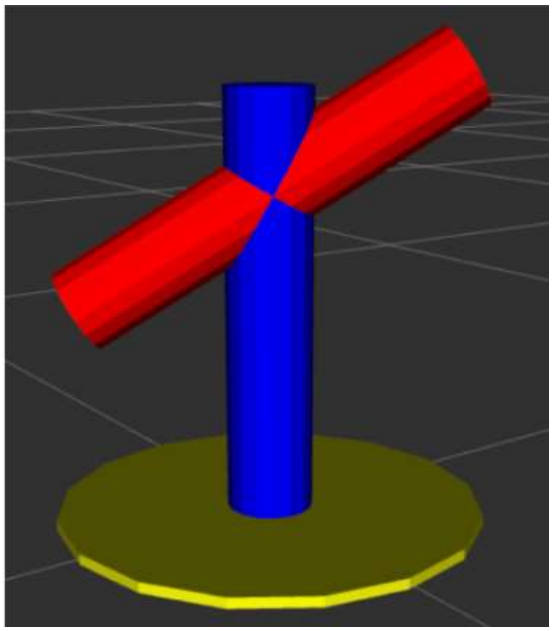
1. catkin_ws/src에 패키지를 만듭니다.
$ catkin_create_pkg test_urdf roscpp tf geometry_msgs urdf rviz xacro



$ mkdir launch
$ mkdir urdf

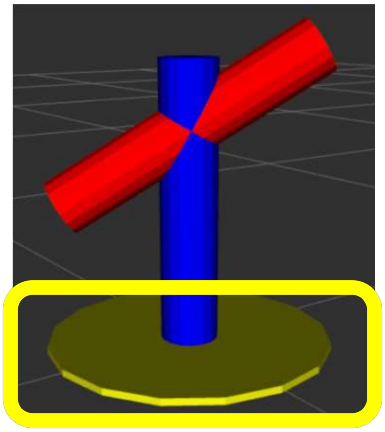


* 예제에 사용될 로봇 외형



예제 로봇은 몸체(받침)인 base와, 기둥인 pan, 팔인 tilt를 가집니다.

* 각 부위의 link들을 작성합니다.
link와 link가 연결 되는 부분의 joint도 기술합니다.
이 로봇은 base_link + pan_joint + pan_link + tilt_joint + tilt 로 구성됩니다.



1. 먼저 base_link를 기술합니다.

각 link는 visual, collision, inertial이라는 항목을

visual 은 로봇의 외형을
collision 은 충돌요소에 대한 시뮬레이션을 위한 것.
Inertial 은 로봇의 무게와 관성모멘트 값을 표현.

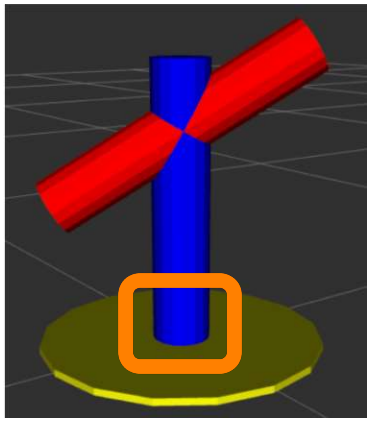
```
<?xml version="1.0"?>
<robot name="test_urdf_pan_tilt">

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.01" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <material name="yellow">
        <color rgba="1 1 0 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.03" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </collision>

    <inertial>
      <mass value="1"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
  </link>


</robot>
```

geometry: 해당 link 의 모습과 크기 결정. 대표적으로 Cylinder, mesh, box, sphere 4종류가 있음. 이 중 mesh는 cad 디지털 출력 데이터를 불러오는 방법.

material: link의 색. rgba 각각의 값으로 색 조절. 값은 0~1.0 사이.

collision : gazebo 시뮬레이션 프로그램에서 충돌에 관한 부분. 기본적으로 visual과 똑같은 값을 가져가면 되지만 예외적인 경우들이 있음.

origin: 해당 물체의 무게중심에서 얼마나 이동할 것인지 결정.



2. base와 pan(기둥)을 연결하는 joint 부분입니다.

joint는 움직이는 관절 부분입니다.
Dynamic 엘리먼트에서 damping은 반대 방향으로 적용하는 관성을 표현하고, friction은 접촉면의 마찰계수 입니다.

```
〈joint name="pan_joint" type="revolute"〉
  〈parent link="base_link"/〉
  〈child link="pan_link"/〉
  〈origin xyz="0 0 0.1"/〉
  〈axis xyz="0 0 1"/〉
  〈limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/〉
  〈dynamics damping="50" friction="1"/〉
〈/joint〉
```

* joint elements

The joint element has following elements:

- ⟨origin⟩ (optional: defaults to identity if not specified)
  - •This is the transform from the parent link to the child link. The joint is located at the origin of the child link, as shown in the figure above.
  - xyz (optional: defaults to zero vector)
    - •Represents the x, y, z offset.
  - rpy (optional: defaults 'to zero vector 'if not specified)
    - •Represents the rotation around fixed axis: first roll around x, then pitch around y and finally yaw around z. All angles are specified in radians.
- ⟨parent⟩ (required)
  - •Parent link name with mandatory attribute:
  - link
    - •The name of the link that is the parent of this link in the robot tree structure.
- ⟨child⟩ (required)
  - •Child link name with mandatory attribute:
  - link
    - •The name of the link that is the child link.
- ⟨axis⟩ (optional: defaults to (1,0,0))
  - •The joint axis specified in the joint frame. This is the axis of rotation for revolute joints, the axis of translation for prismatic joints, and the surface normal for planar joints. The axis is specified in the joint frame of reference. Fixed and floating joints do not use the axis field.
  - xyz (required)
    - •Represents the x, y, z components of a vector. The vector should be normalized.
- ⟨calibration⟩ (optional)
  - •The reference positions of the joint, used to calibrate the absolute position of the joint.
  - rising (optional)
    - •When the joint moves in a positive direction, this reference position will trigger a rising edge.
  - falling (optional)
    - •When the joint moves in a positive direction, this reference position will trigger a falling edge.
- ⟨dynamics⟩ (optional)
  - •An element specifying physical properties of the joint. These values are used to specify modeling properties of the joint, particularly useful for simulation.
  - damping (optional, defaults to 0)
    - •The physical damping value of the joint ($\frac{N \cdot s}{m}$ for prismatic joints, $\frac{N \cdot m \cdot s}{rad}$ for revolute joints).
  - friction (optional, defaults to 0)
    - •The physical static friction value of the joint (N for prismatic joints, $N \cdot m$ for revolute joints).
- ⟨limit⟩ (required only for revolute and prismatic joint)
  - •An element can contain the following attributes:
  - lower (optional, defaults to 0)
    - •An attribute specifying the lower joint limit (radians for revolute joints, meters for prismatic joints). Omit if joint is continuous.
  - upper (optional, defaults to 0)

- An attribute specifying the upper joint limit (radians for revolute joints, meters for prismatic joints). Omit if joint is continuous.

effort (required)

- An attribute for enforcing the maximum joint effort (|applied effort| < |effort|). See safety limits.

velocity (required)

- An attribute for enforcing the maximum joint velocity. See safety limits.

⟨safety_controller⟩ (optional)

- An element can contain the following attributes:

soft_lower_limit (optional, defaults to 0)

- An attribute specifying the lower joint boundary where the safety controller starts limiting the position of the joint. This limit needs to be larger than the lower joint limit (see above). See See safety limits for more details.

soft_upper_limit (optional, defaults to 0)

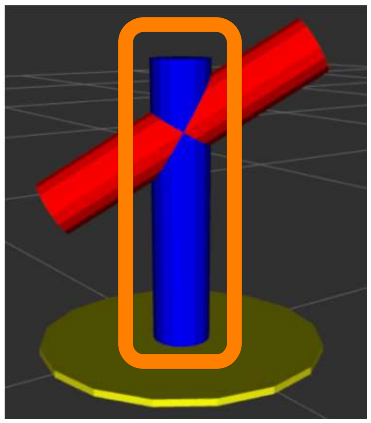
- An attribute specifying the upper joint boundary where the safety controller starts limiting the position of the joint. This limit needs to be smaller than the upper joint limit (see above). See See safety limits for more details.

k_position (optional, defaults to 0)

- An attribute specifying the relation between position and velocity limits. See See safety limits for more details.

k_velocity (required)

- An attribute specifying the relation between effort and velocity limits. See See safety limits for more details.

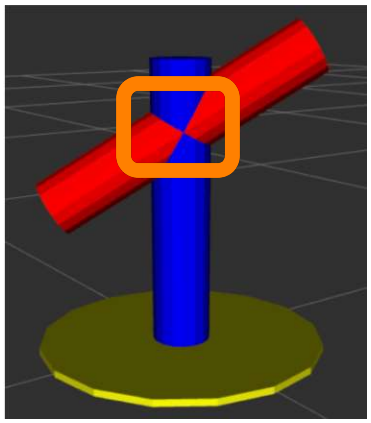


3. pan link 부분입니다.

각 link는 visual, collision, inertial이라는 항목을

visual 은 로봇의 외형을
collision 은 충돌요소에 대한 시뮬레이션을 위한 것.
Inertial 은 로봇의 무게와 관성모멘트 값을 표현.

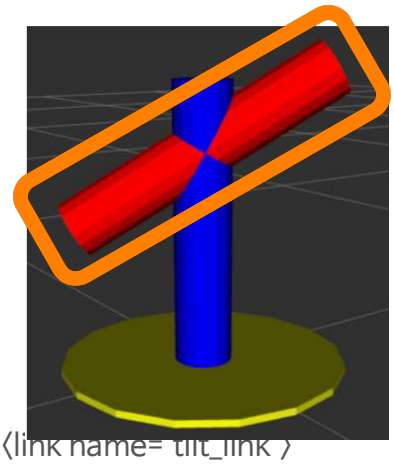
```
<link name="pan_link">
  <visual>
    <geometry>
      <cylinder length="0.4" radius="0.04"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
    <material name="red">
      <color rgba="0 0 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <cylinder length="0.4" radius="0.06"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
  </collision>

  <inertial>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>
```



4. **pan** 과 **tilt** 를 연결하는 **joint** 부분입니다.

joint는 움직이는 관절 부분입니다.
Dynamic 엘리먼트에서 **damping**은 반대 방향으로 적용하는 관성을 표현하고, **friction**은 접촉면의 마찰계수 입니다.
단위는 각형 조인트의 경우 Newton second per meter
회전 조인트의 경우 Newton meter second per radian 입니다.

```
〈joint name="tilt_joint" type="revolute"〉
  〈parent link="pan_link"/〉
  〈child link="tilt_link"/〉
  〈origin xyz="0 0 0.2"/〉
  〈axis xyz="0 1 0" /〉
  〈limit effort="300" velocity="0.1" lower="-4.71239" upper="-1.570796"/〉
  〈dynamics damping="50" friction="1"/〉
〈/joint〉
```



5. **tilt link** 부분입니다.

각 **link**는 **visual**, **collision**, **inertial**이라는 항목을

visual 은 로봇의 외형을
collision 은 충돌요소에 대한 시뮬레이션을 위한 것.
Inertial 은 로봇의 무게와 관성모멘트 값을 표현.

```
〈link name= tilt_link 〉
  〈visual〉
    〈geometry〉
      〈cylinder length="0.4" radius="0.04"/〉
    〈/geometry〉
    〈origin rpy="0 1.570796 0" xyz="0 0 0"/〉
    〈material name="green"〉
      〈color rgba="1 0 0 1"/〉
    〈/material〉
  〈/visual〉

  〈collision〉
    〈geometry〉
      〈cylinder length="0.4" radius="0.06"/〉
    〈/geometry〉
    〈origin rpy="0 1.570796 0" xyz="0 0 0"/〉
  〈/collision〉

  〈inertial〉
    〈mass value="1"/〉
    〈inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/〉
  〈/inertial〉
〈/link〉

〈/robot〉
```

* urdf 전체

```
<?xml version="1.0"?>
<robot name="test_urdf_pan_tilt">

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.01" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <material name="yellow">
        <color rgba="1 1 0 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.03" radius="0.2"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </collision>

    <inertial>
      <mass value="1"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
  </link>

  <joint name="pan_joint" type="revolute">
    <parent link="base_link"/>
    <child link="pan_link"/>
    <origin xyz="0 0 0.1"/>
    <axis xyz="0 0 1" />
    <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/>
    <dynamics damping="50" friction="1"/>
  </joint>

  <link name="pan_link">
    <visual>
      <geometry>
        <cylinder length="0.4" radius="0.04"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.09"/>
      <material name="red">
        <color rgba="0 0 1 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.4" radius="0.06"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0.09"/>
    </collision>
```

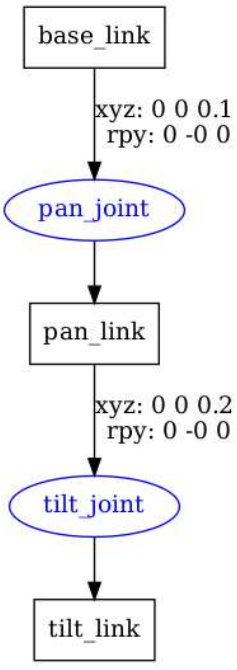
```
    <inertial>
      <mass value="1"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
  </link>

  <joint name="tilt_joint" type="revolute">
    <parent link="pan_link"/>
    <child link="tilt_link"/>
    <origin xyz="0 0 0.2"/>
    <axis xyz="0 1 0" />
    <limit effort="300" velocity="0.1" lower="-4.71239" upper="-1.570796"/>
    <dynamics damping="50" friction="1"/>
  </joint>

  <link name="tilt_link">
    <visual>
      <geometry>
        <cylinder length="0.4" radius="0.04"/>
      </geometry>
      <origin rpy="0 1.570796 0" xyz="0 0 0"/>
      <material name="green">
        <color rgba="1 0 0 1"/>
      </material>
    </visual>

    <collision>
      <geometry>
        <cylinder length="0.4" radius="0.06"/>
      </geometry>
      <origin rpy="0 1.570796 0" xyz="0 0 0"/>
    </collision>

    <inertial>
      <mass value="1"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
  </link>

</robot>
```

* 체크 urdf

$ check_urdf pan_tilt.urdf

```
robot name is: test_urdf_pan_tilt
---------- Successfully Parsed XML ---------------
root Link: base_link has 1 child(ren)
  child(1): pan_link
    child(1): tilt_link
```

$ urdf_to_graphiz pan_tilt.urdf



* 이제 launch 폴더에 view_pan_tilt_urdf.launch 라는 launch 파일을 만들어 줍니다.

$ gedit view_pan_tilt_urdf.launch

내용:

```
<launch>
  <arg name="model" />

  <param name="robot_description" textfile="$(find test_urdf)/urdf/pan_tilt.urdf" />

  <!-- Setting gui parameter to true for display joint slider -->
  <param name="use_gui" value="true"/>
  <!-- Starting Joint state publisher node which will publish the joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <!-- Starting robot state publish which will publish tf -->
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" />
  <!-- Launch visualization in rviz -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find test_urdf)/urdf.rviz"
required="True" />
</launch>
```

***

$ roslaunch test_urdf view_pan_tilt_urdf.launch

Global Option의 Fixed Frame을 base_link로,
Add > RobotModel 과 TF

noetic에서 gui가 안나오면, 다른 Terminal 창을 열어서
$ rosrun joint_state_publisher_gui joint_state_publisher_gui