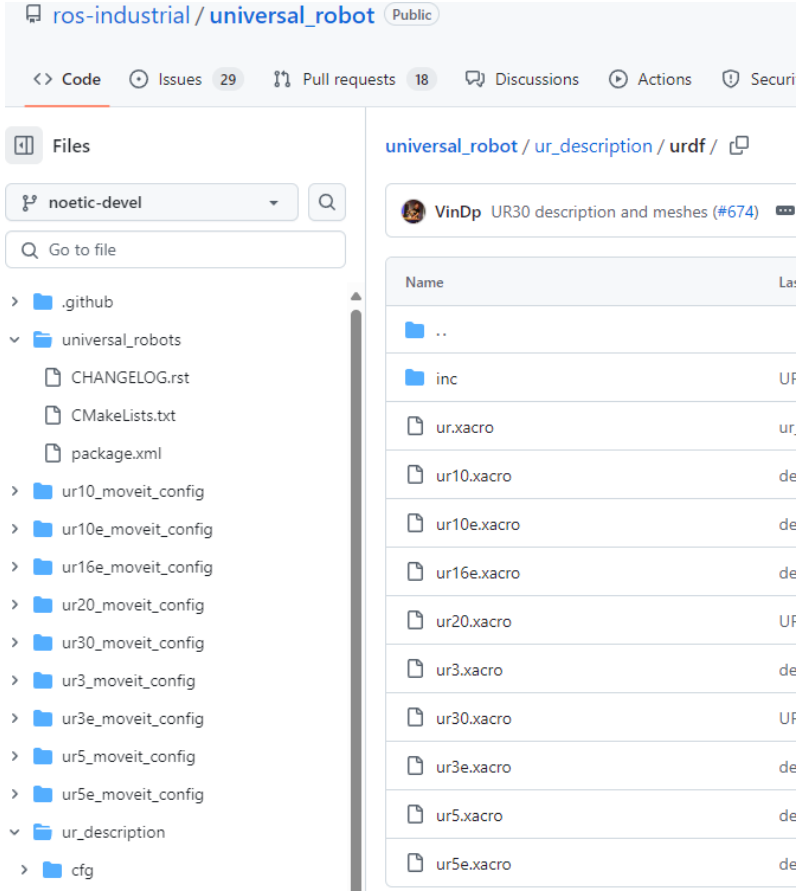
URDF는 무엇인가?

- URDF(Unified Robot Description Format)는 XML 파일을 사용하여 로봇, 센서 및 작업 환경을 모델링
  - XML 태그를 사용하여 URDF는 다음을 나타낼 수 있습니다.
    * 로봇의 기구학, 동역학, 시각 모델, 충돌 모델
    ** 확장자는 .urdf이고 xacro file에서 변환하기도 함.



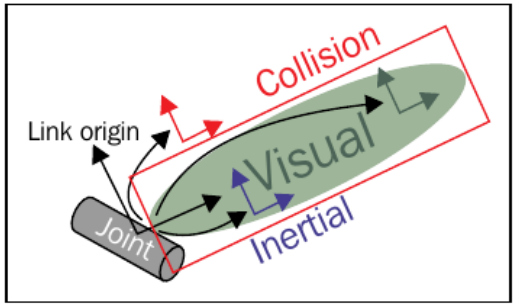
URDF는 무엇인가?

URDF의 구성

– link 태그

* 로봇의 단일 링크(질량 및 관성 매트릭스), 시각적 모델(형상 및 재료) 및 충돌 모델(형상)의 관성 속성

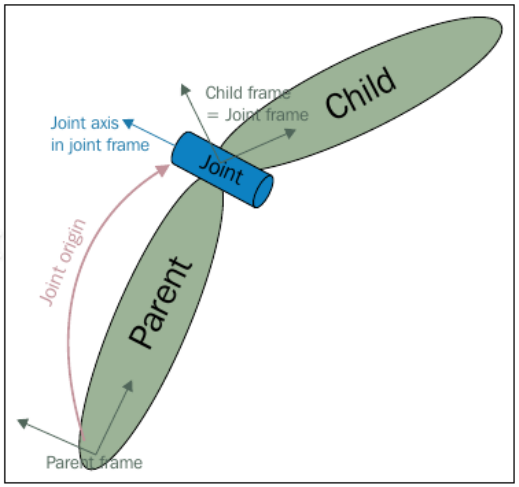
```
<link name="<name of the link>">
  <inertial>.....</inertial>
  <visual> .....</visual>
  <collision>.....</collision>
</link>
```



– joint 태그

* 로봇 조인트의 유형(revolute, continuous, prismatic, fixed, floating, planar), 연결하는 링크(부모 링크, 자식 링크) 관계 설명

```
<joint name="<name of the joint>" type="<type of the joint>">
   <origin xyz="..." rpy="..."/>
   <parent link="link1"/>
   <child link="link2"/>
   <axis>"..."</axis>
   <dynamics damping="...."/>
   <limit effort="..."/>
</joint>
```

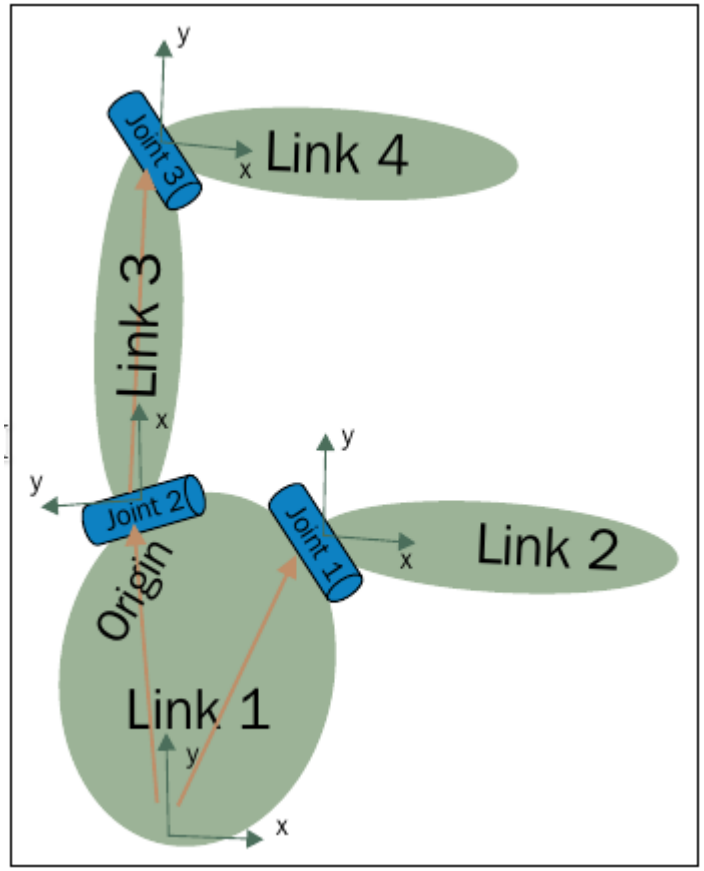


URDF는 무엇인가?

URDF의 구성

- Robot 태그
  - * 이름, 링크 및 관절. URDF를 사용하여 트리 구조

```
<robot name="<name of the robot>"/>
  <link> ..... </link>
  <link> ..... </link>
  <joint> ..... </joint>
  <joint> ..... </joint>
</robot>
```



URDF는 무엇인가?

RCI LAB KHU

(Option) Xacro

Xacro (XML Macros)

Xacro is an XML macro language that extends URDF by providing:

- **Modularity**: It creates macros that can be reused (even in other files), which results in a modular and more readable code.
- **Programmability**: It supports simple programming statements, which makes the description more efficient.

Xacro files modify the *robot* tag to include a namespace:

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="pan_tilt">
```

Properties

Constants can be declared and used as named values inside the Xacro file, thus allowing to change these values more easily.

```
<xacro:property name="pan_link_length" value="0.01" />
<xacro:property name="pan_link_radius" value="0.2" />
....
<cylinder length="${pan_link_length}" radius="${pan_link_radius}"/>
```

Math expressions

Basic operations such as + , -, * , / , unary minus, and parenthesis can be used inside ${}.

```
<cylinder length="${pan_link_length}" radius="${pan_link_radius+0.02}"/>
```

Includes

Other Xacro files can be included inside a Xacro file using the *xacro:include* tag, e.g.:

```
<xacro:include filename="$(find packageName)/urdf/anotherXacroFile.xacro"/>
```

Macros

Macros can be defined to allow modularity. They improve the code readability and reduce the number of lines compared to urdf, e.g.:

```
<xacro:macro name="inertial_matrix" params="mass">
 <inertial>
  <mass value="${mass}" />
  <inertia ixx="0.5" ixy="0.0" ixz="0.0" iyy="0.5" iyz="0.0" izz="0.5" />
 </inertial>
</xacro:macro>
...
<xacro:inertial_matrix mass="1"/>
```

URDF는 무엇인가?

- 실습
  - Pan_tilt 로봇의 URDF 파일 만들기 (urdf_tutorial/urdf folder)

    * Xacro 에서 URDF로 파일 만들기

      (Terminal) ros2 run xacro xacro original_file.xaro 〉 new_file.urdf

    * Xacro 가 없다라는 오류가 뜬다면, xacro 설치

      (Terminal) sudo apt install ros-humble-xacro

URDF는 무엇인가?

- Visualization: Rviz2
  - Launch File을 만들어서,

    * Rviz 노드
    * Joint_state_publisher 노드
    * Robot_state_publisher 노드를 실행시키도록.

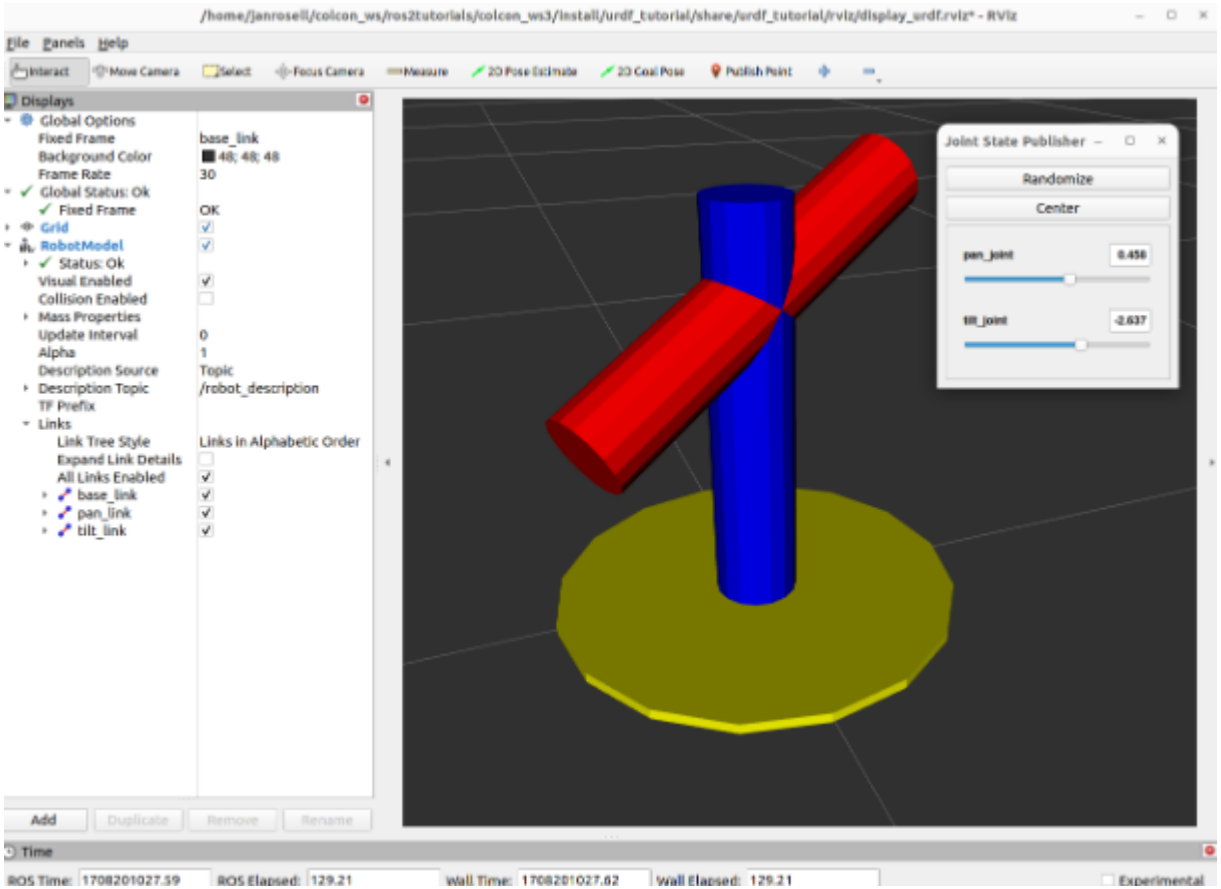
```
def generate_launch_description():

    # General arguments
    description_package = LaunchConfiguration("description_package", default='urdf_tutorial')
    description_file = LaunchConfiguration("description_file", default='pan_tilt.xacro')

    launch_rviz = LaunchConfiguration("launch_rviz", default='true')
    rviz_config_file = PathJoinSubstitution(
        [FindPackageShare("urdf_tutorial"), "rviz", "display_urdf.rviz"]
    )

    robot_description_content = Command(
        [
            PathJoinSubstitution([FindExecutable(name="xacro")]),
            " ",
            PathJoinSubstitution(
                [FindPackageShare(description_package), "urdf", description_file]
            )
        ]
    )
    robot_description = {"robot_description": robot_description_content}

    robot_state_publisher_node = Node(
        package="robot_state_publisher",
        executable="robot_state_publisher",
        output="both",
        parameters=[{"use_sim_time": False}, robot_description],
    )

    joint_state_publisher_gui = Node(
        package="joint_state_publisher_gui",
        executable="joint_state_publisher_gui",
        name="joint_state_publisher_gui",
        output="screen",
    )
```

URDF는 무엇인가?

실습

- Pan_tilt 로봇의 Launch를 실행해보기.

  * 만약 joint_state_publisher_gui 오류가 뜬다면
  (Terminal) sudo apt install ros-humble-joint-state-publisher-gui



URDF는 무엇인가?

- 실습
  - UR5로봇의 URDF 파일을 만들고 (ur_urdf/urdf folder)
    * UR5를 실행해볼 것.

URDF는 무엇인가?

joint_state_publisher

joint_state_publisher 패키지에는 지정된 URDF에 대한 공동 상태 값을 설정하고 게시하기 위한 *joint_state_publisher* 노드가 포함되어 있습니다.

- URDF는 명령줄 또는 */robot_description* 항목을 통해 전달될 수 있습니다.
- 노드는 URDF 모델에서 고정되지 않은 조인트를 찾아 각 조인트의 조인트 상태 값을 *sensor_msgs/JointState* 메시지 형식으로 */joint_states* 항목에 게시합니다.

매개 변수

- *rate*(int): */joint_states* 주제에 대한 업데이트를 게시하는 속도입니다. 기본값은 10입니다.
- *publish_default_positions* (bool): 각 이동 가능한 조인트의 기본 위치를 */joint_states* 토픽에 게시할지 여부입니다. 기본값은 True입니다.
- *source_list* (문자열 배열): 구독할 *sensor_msgs/msg/JointStates* 유형의 주제 목록입니다.

게시된 주제

- */joint_states* (sensor_msgs/msg/JointState): 시스템에 있는 모든 이동 가능한 조인트의 상태입니다.

구독한 주제

- (선택 사항) */robot_description* (std_msgs/msg/String): 명령줄에 URDF가 제공되지 않으면 이 노드는 게시할 URDF에 대한 /robot_description 토픽을 수신합니다. 한 번 이상 수신되면 이 노드는 */joint_states*에 조인트 값을 게시하기 시작합니다.
- (선택 사항) */any_topic* (sensor_msgs/msg/JointState): sources_list 매개변수가 비어 있지 않은 경우(위의 매개변수 참조) 이 매개변수의 명명된 모든 토픽은 공동 상태 업데이트를 위해 구독됩니다.

슬라이더 기반 제어 창과 함께 GUI를 사용하여 각 조인트를 제어하려면 패키지 조인트 상태 publisher_gui 대신 사용해야 합니다.

URDF는 무엇인가?

robot_state_publisher

로봇 상태 게시자 패키지에는 로봇 상태 게시자, 노드 및 로봇 상태를 tf2에 게시하기 위한 클래스가 포함되어 있습니다. 시작 시 Robot State Publisher에는 로봇의 URDF(Kinematic Tree Model)가 제공됩니다. 그런 다음 joint_states 주제(sensor_msgs/msg/JointState 유형)를 구독하여 개별 조인트 상태를 가져옵니다. 이러한 조인트 상태는 키네마틱 트리 모델을 업데이트하는 데 사용되며, 결과 3D 포즈는 tf2에 게시됩니다.

매개 변수

- *robot_description*(문자열): 시작 시간에 설정된 URDF 형식의 로봇 설명robot_state_publisher 이 매개변수에 대한 향후 업데이트가 가능하며 robot_description 항목에 반영됨).
- *frame_prefix*(문자열): 게시된 tf2 프레임에 추가할 임의의 접두사입니다. 기본값은 빈 문자열입니다.

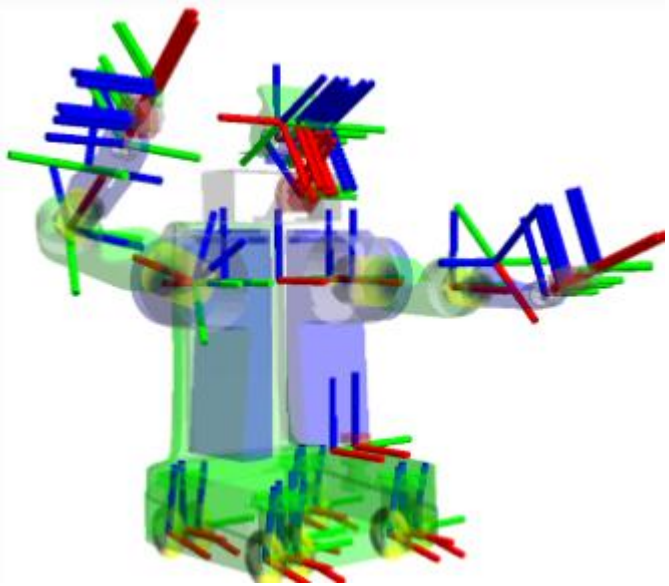
구독한 주제

- *joint_states* (sensor_msgs/msg/JointState): 로봇 포즈에 대한 관절 상태 업데이트입니다.

게시된 주제

- *robot_description* (std_msgs/msg/String): 로봇 URDF에 대한 설명(문자열)입니다.
- *tf* (tf2_msgs/msg/TFMessage): 로봇의 이동 가능한 관절에 해당하는 트랜스폼입니다.
- *tf_static* (tf2_msgs/msg/TFMessage): 로봇의 정적 조인트에 해당하는 트랜스폼입니다.

사용자가 시간 경과에 따른 여러 좌표 프레임을 추적할 수 있도록 하는 Transform 라이브러리



- 5초 전에 세계 프레임에 대한 헤드 프레임은 어디에 있었습니까?
- 내 베이스에 대한 그리퍼에 있는 물체의 위치는 무엇입니까?
- 맵 프레임에서 기본 프레임의 현재 포즈는 무엇입니까?

실습

- UR5를 Rviz에 띄우고, world frame에서 tool0 프레임까지의 위치를 구해보시오

  * ros2 run tf2_ros tf2_echo frameA frameB

Action과 함께하는 Rviz

- 실습
  - ros2 launch pantilt_action_tutorials_py pantilt_action_server.launch.py
  - ros2 launch pantilt_action_tutorials_py pantilt_action_client_arguments.launch.py joint_name:="tilt_joint" desired_angle:=70 stepsize:=5

Navigation

Navigation 노드 실행

Terminal 1

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

Terminal 2

```
$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=path/to/my_map.yaml
```

Home에 위치한다면 ${HOME}/my_map.yaml