

URDF

ROS မှာ Robot Model တစ်ခုကို Represent လုပ်ဖို့အတွက် XML Format နဲ့ URDF (Unified Robot Description Format) ဆိုတဲ့ Package တစ်ခုရှိတယ် ၊ သူ့ကို အသုံးပြုဖို့အတွက် urdf extension နဲ့ဆုံးတဲ့ File လေးတစ်ခုတည်ဆောက်ပေးရမယ် ၊ အဲဒီ File လေးထဲမှာ Robot Model တစ်ခုအတွက်လိုအပ်တဲ့ အချက်အလက်မှန်သမျှပါပါတယ် ။

URDF ထဲမှာသုံးမယ့် အရေးကြီးတဲ့ Tag ၆ ခုရှိပါတယ် ။

1. robot tag (<robot>.....</robot>)
2. link tag (<link>.....</link>)
3. joint tag (<joint>.....</joint>)
4. sensor tag (<sensor>.....</sensor>)
5. transmission (<transmission>.....</transmission>)
6. gazebo (<gazebo>.....</gazebo>)

အခုပြောပြမှာကတော့ simulation လုပ်တဲ့အခါ urdf file ထဲမှာ robot model တစ်ခုလုံးအတွက် လိုအပ်မယ့် tag လေးခုကို tag တစ်ခုချင်းစီ အသေးစိတ်ရှင်းပြပေးသွားမှာ ဖြစ်ပါတယ် ၊ သို့ပေမယ့် ဒီ pdf ထဲမှာ အရေးကြီးတဲ့ tag နှစ်ခုတော့ မပါဝင်ပါဘူး ၊ transmission tag နဲ့ gazebo tag ပါ ။ အဲဒီ tag နှစ်ခုက မိမိတို့ robot model ကို simulation လုပ်တဲ့အခါ control လုပ်မယ့် tag တွေ ဖြစ်တာကြောင့် မထည့်ထားပါဘူး (ကျနော်လည်း သေချာမလေ့လာရသေးလို့) ။

ဒီ pdf ရဲ့ရည်ရွယ်ချက်က robot model ကို control လုပ်တာမပါဘဲ robot model သီးသန့်ပါ ။

tag တွေအကြောင်း စမပြောခင် ကြိုသိထားရမှာက Tag တစ်ခုမှာ attribute နဲ့ element ဆိုပြီး အချက်နှစ်ချက် ရှိတယ်ဆိုတာ သိထားပေးပါ ။

Attribute ကတော့ ဒီ tag ရဲ့ characteristic ဖြစ်ပါတယ် ၊ Elements ဆိုတာကတော့ ဒီ Tag ထဲမှာရှိနိုင်တဲ့ တခြားသော Tag တွေကိုဆိုလိုခြင်းဖြစ်ပါတယ် ။

1 - robot tag

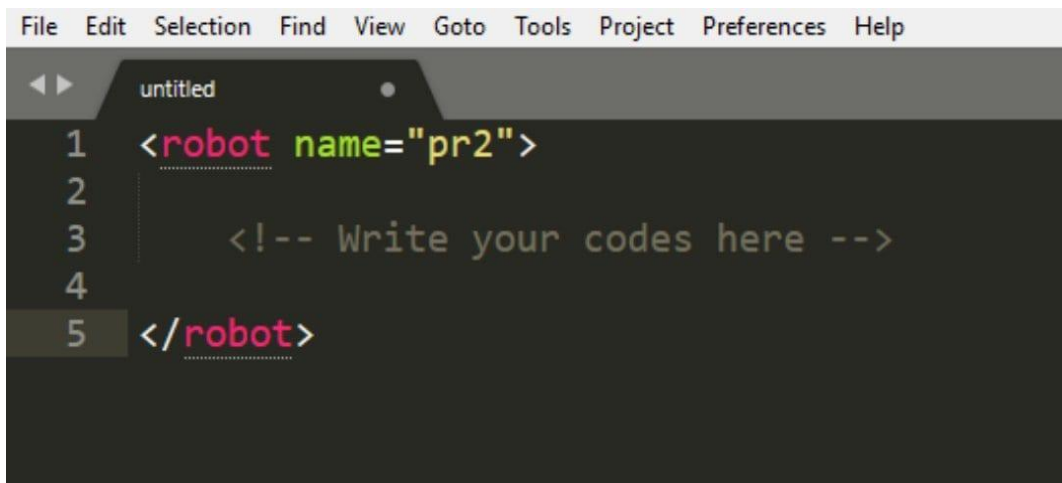
Syntax - `<robot>.....</robot>`

ဒီ Tag နှစ်ခုအတွင်းမှာ robot အတွက် လိုအပ်တဲ့ properties တွေအကုန်ဖော်ပြရပါတယ် ။ သူကတော့ root tag ကြီးပါ ၊ တခြားသော tag တွေလည်း သူ့ထဲမှာ ရေးရပါတယ် ။

Attributes of robot tag

1. name

name ဆိုတာကတော့ မိမိ robot ရဲ့နာမည်ပါ - ဥပမာအနေနဲ့ အောက်ကပုံလေးမှာ ကြည့်ပါ ။

A screenshot of a code editor window titled 'untitled'. The editor shows the following code:

```
1 <robot name="pr2">
2
3   <!-- Write your codes here -->
4
5 </robot>
```

The code is color-coded: '<robot' is red, 'name="pr2"' is green, and '</robot>' is red. The comment on line 3 is in grey.

အထက်ပါပုံမှာဆိုရင် robot name က pr2 ဖြစ်ပါတယ်

Elements of robot tag

link tag (`<link>.....</link>`)

joint tag (`<joint>.....</joint>`)

transmission tag (`<transmission>.....</transmission>`)

gazebo tag (`<gazebo>.....</gazebo>`)

robot tag ရဲ့ elements တွေကိုတော့ အောက်က tag တစ်ခုချင်းစီမှာ အသေးစိတ်ကြည့်ပါ

2 - link tag

Syntax - `<link>.....</link>`

ဒီ tag နှစ်ခုအတွင်းမှာတော့ robot link တစ်ခုရဲ့ kinematic နဲ့ dynamic properties တွေကို ဖော်ပြရာမှာ သုံးပါတယ် ၊ အောက်မှာဆက်ကြည့်ရင် ပိုရှင်းသွားပါလိမ့်မယ် ။

Attributes of link tag

1 – name (required)

name ဆိုတာကတော့ မိမိ robot ရဲ့ link နာမည်ပါ - ဥပမာအနေနဲ့ အောက်ကပုံလေးမှာ ကြည့်ပါ ။

A screenshot of a Sublime Text editor window titled 'untitled - Sublime Text (UNREGISTERED)'. The editor shows a URDF link tag example. The code is as follows:

```
1 <link name="first_link">
2
3   <!-- Write your codes here -->
4
5 </link>
```

အထက်ပါပုံမှာဆိုရင် link name က first_link ဖြစ်ပါတယ်

Elements of link tag

- (i) - `<visual>.....</visual>` tag
- (ii) - `<collision>.....</collision>` tag
- (iii) - `<inertial>.....</inertial>` tag

Tag တစ်ခုချင်းစီကို ဆက်ရှင်းပါမယ်

(i) **<visual>**.....**</visual>**

ဒီ Tag မှာတော့ robot ရဲ့ real link ကို ဖော်ပြပါတယ် ၊ ကျနော်တို့ Gazebo ထဲမှာ simulation လုပ်တဲ့အခါ Rviz ထဲမှာ visualization လုပ်တဲ့အခါမှာ ဒီ tag ထဲမှာရေးထားတဲ့အတိုင်းမြင်ရမှာပါ။ သူ့မှာလည်း sub-tag လေးတွေရှိပါတယ်

1- **<origin>** - ဒီ sub-tag မှာတော့ ကိုယ်ဖန်တီးတဲ့ robot link ကို initial pose (position+orientation) ဘယ်နေရာမှာထားမယ်ဆိုတာတွေကို ဒီ Tag အတွင်းမှာရေးရပါတယ် ၊ တနည်းအားဖြင့် link's reference frame ရဲ့ pose ပေါ့ ။ position အတွက် xyz ဆိုတဲ့ parameter နဲ့ orientation အတွက် rpy ဆိုတဲ့ parameter လေးတွေရှိပါတယ် ။ ဒီ origin sub-tag လေးကတော့ optional ပါ ၊ သဘောကတော့ ထည့်ထည့်၊မထည့်ထည့် ရတယ်ပေါ့ဗျာ ၊ မထည့်ဖူးဆိုရင် default က identity matrix ပါပဲ ။

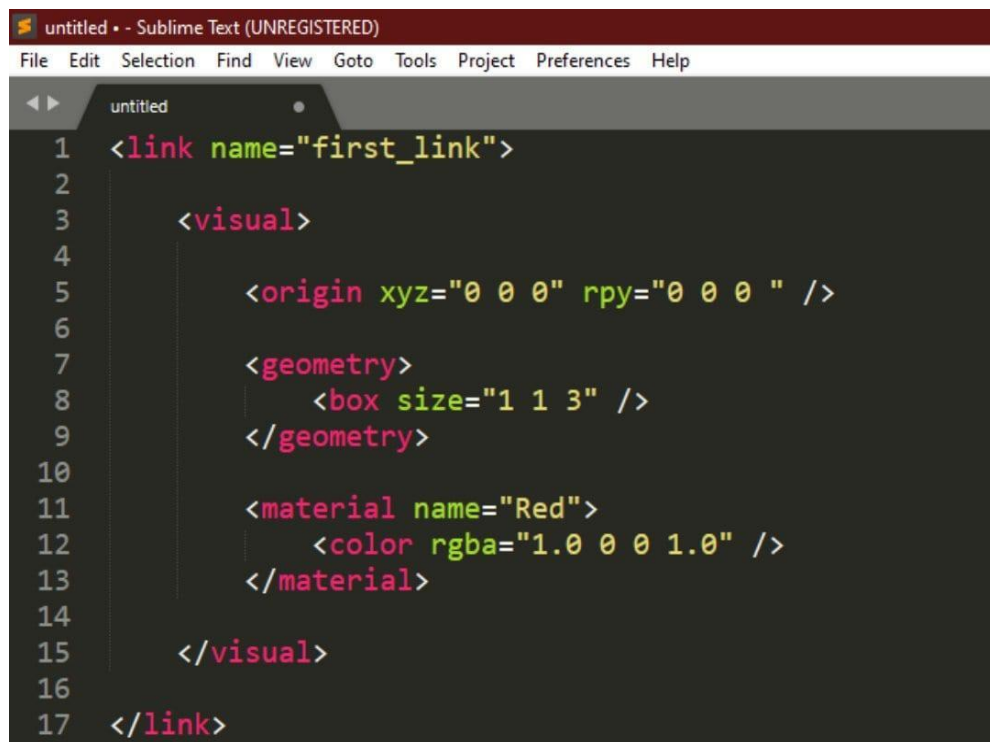
2 - **<geometry>**.....**</geometry>** - ဒီ sub-tag မှာတော့ ကိုယ်ဖန်တီးချင်တဲ့ link ရဲ့ shape(box,cylinder,sphere,etc...) ကို ရေးပေးရပါတယ် ၊ mesh file လည်း ထည့်ပေးလို့ရပါတယ် ၊ ထည့်ပေးဖို့အတွက် tag လေး သေးသေးကလေး ၄ ခုရှိပါတယ် (**<box>**,**<cylinder>**,**<sphere>**,**<mesh>**) ။ ဒီ Tag က required ပါ ၊ geometry sub-tag ထဲမှာ မဖြစ်မနေ ထည့်ပေးရပါတယ် ။

3 - **<material>**.....**</material>** - ဒီ sub-tag မှာတော့ ကိုယ်က ဒီ robot link ကို ဘယ်လို material နဲ့ ဖန်တီးချင်လဲဆိုတာ ရေးပေးရပါတယ် ၊ material ရဲ့ နာမည်ကိုဖော်ပြဖို့ name parameter ၊ material ရဲ့ အရောင်ကိုဖော်ပြဖို့အတွက် color ဆိုတဲ့ tag လေး သေးသေး ကလေးထဲမှာ

rgba ဆိုတဲ့ parameter နဲ့ material ရဲ့ texture ကို ဖော်ပြဖို့အတွက် texture ဆိုတဲ့ tag လေး သေးသေးလေးထဲမှာ texture file ကို ထည့်ပေးလို့ရပါတယ် ၊ ဒီ material sub-tag လေးကတော့ **optional** ပါ ။

ပြီးတော့ visual tag မှာ **name** ဆိုတဲ့ parameter လေးလည်းရှိပါတယ် ၊ link ရဲ့ geometry name ပါ ၊ သုံးတာတော့ သိပ်မတွေ့ဖူးဘူးဗျ ။ သူလည်း optional ပါ ။

အောက်က ပုံလေးမှာကြည့်ပါ ။



```
1 <link name="first_link">
2
3   <visual>
4
5     <origin xyz="0 0 0" rpy="0 0 0" />
6
7     <geometry>
8       <box size="1 1 3" />
9     </geometry>
10
11     <material name="Red">
12       <color rgba="1.0 0 0 1.0" />
13     </material>
14
15   </visual>
16
17 </link>
```

အထက်ပါပုံမှာ အလျား 1m ၊ အနံ 1m အမြင့် 3m ရှိတဲ့ စတုရန်းပုံစံ link လေးတစ်ခုကို initial position(x=0,y=0,z=0) နဲ့ initial orientation(row=0,pitch=0,angle=0) ရှိတဲ့ reference frame တစ်ခုမှာ နေရာချထားပါတယ် ။ material name ကတော့ Red လို့ပေးထားပြီး color ကတော့ အနီရောင် ပါ ။

(ii) - **<collision>**.....**</collision>**

ဒီ Tag ကတော့ collision checking အတွက် အသုံးပြုပါတယ် ၊ သဘောကတော့ visual tag ကနေ ဖန်တီးထားတဲ့ real link ကို တခြား object တစ်ခုခုနဲ့ မတိုက်မိအောင် သို့ link အချင်းချင်း self-collision မဖြစ်အောင် detect လုပ်ပေးတာပါ ။ မြင်အောင်ပြောရရင် အကာအုပ်ပေးတဲ့သဘောပါပဲ - ကြက်ဥပြုတ်နဲ့ ဥပမာပေးရမယ်ဆို ပိုသိသာမယ်ထင်တယ် ၊ visual tag က ဖန်တီးထားတဲ့ real link က ကြက်ဥပြုတ်ရဲ့အနှစ်ဖြစ်ပြီး collision tag က ဖန်တီးထားတဲ့ link က ကြက်ဥပြုတ်ရဲ့အကာပေါ့ ။ သူ့မှာလည်း sub-tag လေးတွေရှိပါတယ်။

1 - **<geometry>**.....**</geometry>** ဒီ sub-tag မှာတော့ အပေါ်က visual tag မှာရှိတဲ့ geometry sub-tag လေးနဲ့ တူတူပါပဲ - value တွေလည်း တူတူပေးလို့လည်းရပါတယ် ။

2 - **<origin>** ဒီ sub-tag မှာလည်း အပေါ်က visual tag မှာရှိတဲ့ origin sub-tag လေးနဲ့ တူတူပါပဲ - value တွေလည်း တူတူပေးလို့လည်းရပါတယ် ။

ပြီးတော့ collision tag မှာလည်း **name** ဆိုတဲ့ parameter လေးရှိပါတယ် ၊ link ရဲ့ geometry name ပါ ၊ သုံးတာတော့ သိပ်မတွေ့ဖူးဘူးဗျ ။ သူလည်း optional ပါ ။

အောက်က ပုံလေးမှာကြည့်ပါ ။

```
untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

1 <link name="first_link">
2
3   <collision>
4
5     <origin xyz="0 0 0" rpy="0 0 0" />
6
7     <geometry>
8       <box size="1 1 3" />
9     </geometry>
10
11   </collision>
12
13 </link>
```

အထက်ပါပုံမှာ အလျား 1m ၊ အနံ 1m အမြင့် 3m ရှိတဲ့ စတုရန်းပုံစံ collision object လေးတစ်ခုကို initial position($x=0, y=0, z=0$) နဲ့ initial orientation($roll=0, pitch=0, angle=0$) ရှိတဲ့ reference frame တစ်ခုမှာ နေရာချထားပါတယ်။ ။

(iii) - `<inertial>.....</inertial>`

ဒီ Tag ကတော့ robot link ရဲ့ dynamic properties တွေထဲက တစ်ခုဖြစ်တဲ့ inertial properties အတွက် အသုံးပြုပါတယ်။ dynamics ကိုသေချာ မလေ့လာဖူးသေးလို့ မပြောပြနိုင်ပေမယ့် code ထဲမှာ ရေးထားသလောက်ကိုတော့ပြောပြပါမယ်။ သူ့ sub-tag လေးတွေအကြောင်း ပြောကြရအောင်

1 - `<origin>` ဒီ sub-tag မှာတော့ အပေါ်က visual tag နဲ့ collision tag မှာရှိတဲ့ origin sub-tag လေးနဲ့ တူတူပါပဲ - value တွေလည်း တူတူပေးလို့လည်းရပါတယ်။ ။

2 - `<mass>` ဒီ sub-tag မှာတော့ မိမိ robot link ရဲ့ mass ကို value တစ်ခု ရေးပေးရပါတယ်။ ဒီမှာတော့ mass က inertia အတွက် ဘယ်လိုအရေးပါကြောင်း အသေးစိတ်မရှင်းတော့ပါဘူး။ သေချာလေးသိချင်တယ်ဆို ဒီ link ထဲ သွားကြည့်လိုက်ပါ

(<https://www.youtube.com/watch?v=1XSyyjcEHo0>)

3 - **<inertia>** ဒီ sub-tag မှာတော့ link ရဲ့ shape ကို inertia frame ထဲမှာ 3x3 rotational inertia matrix အနေနဲ့ ဖော်ပြတာပါ ၊ အောက်က matrix ကို ကြည့်နိုင်ပါတယ် ၊ 3x3 matrix ဖြစ်လို့ Elements ကိုးခုပေမယ့် ကိုးခုလုံးမသုံးဘဲ Diagonal elements ရဲ့အပေါ်ခြမ်းကိုပဲ သုံးပါတယ် ဘာကြောင့်လဲဆိုတော့ rotational inertia matrix က symmetric ဖြစ်လို့ပါ ။

```

ixx  ixy  ixz
ixy  iyy  iyz
ixz  iyz  izz

```

ဒီ matrix အတိုင်းဆို **ixx,ixy,ixz,iyy,iyz,izz** - ဒီ ခြောက်ခုပေါ့ ၊ ဒီ matrix ခြောက်ခုကို value သတ်မှတ်တာနဲ့ ပက်သက်ပြီး shape က ပြောင်းလဲ သွားတာပါ ၊ ဒီ link မှာ matrix အလိုက် shape အမျိုးမျိုးသွားကြည့်လို့ရပါတယ်

(https://en.wikipedia.org/wiki/List_of_moments_of_inertia#List_of_3D_inertia_tensors)

အောက်က ပုံလေးမှာကြည့်ပါ ။

```

1  <link name="first_link">
2
3      <inertial>
4
5          <origin xyz="0 0 0.5" rpy="0 0 0"/>
6
7          <mass value="1"/>
8
9          <inertia ixx="1.0" ixy="0.0" ixz="0.0"
10             iyy="0.5" iyz="0.0"
11             izz="1.0"/>
12
13      </inertial>
14
15 </link>

```


အပေါ်မှာ ရှင်းပြခဲ့တဲ့ link tag တစ်ခုလုံးကို ရှင်းရှင်းလင်းလင်း ဒီ အောက်က example မှာ ကြည့်နိုင်ပါတယ် ။

```
untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

<> untitled
1 <link name="first_link">
2
3   <visual>
4     <origin xyz="0 0 0" rpy="0 0 0" />
5     <geometry>
6       <box size="1 1 3" />
7     </geometry>
8     <material name="Red">
9       <color rgba="1.0 0 0 1.0"/>
10    </material>
11  </visual>
12
13  <collision>
14    <origin xyz="0 0 0" rpy="0 0 0"/>
15    <geometry>
16      <box size="1 1 3" />
17    </geometry>
18  </collision>
19
20  <inertial>
21    <origin xyz="0 0 0.5" rpy="0 0 0"/>
22    <mass value="1"/>
23    <inertia ixx="1.0" ixy="0.0" ixz="0.0"
24              iyy="0.5" iyz="0.0"
25              izz="1.0" />
26  </inertial>
27
28 </link>
```

3 – Joint Tag

Syntax - `<joint>.....</joint>`

ပထမဦးစား ကြိုသိရမှာက Joint ဆိုတာက link နှစ်ခုကို ဆက်ပေးတဲ့အရာပါ ၊ ပိုရှင်းအောင်ပြောရရင် parent link နဲ့ child link ကိုဆက်ပေးတဲ့အရာပေါ့ ။ ဒီလိုပဲ စေ့ချင်း အလွယ်မှတ်နိုင်ပါတယ် ။

ဒီ tag နှစ်ခုအတွင်းမှာတော့ robot joint တစ်ခုရဲ့ kinematic နဲ့ dynamic properties တွေကို ဖော်ပြရာမှာ သုံးပါတယ် ၊

Joint တစ်ခုရဲ့ safety limits ကိုလည်း သတ်မှတ်ပေးလို့ရပါတယ် ။ အောက်မှာဆက်ကြည့်ရင် ပိုရှင်းသွားပါလိမ့်မယ် ။


Attributes of joint tag

1 – name (required)

name ဆိုတာကတော့ မိမိ robot ရဲ့ Joint နာမည်ပါ - ဥပမာအနေနဲ့ အောက်ကပုံလေးမှာ ကြည့်ပါ ။

2 – type (required)

type ဆိုတာကတော့ မိမိ robot ရဲ့ Joint အမျိုးအစားပါ ။ Joint အမျိုးအစားတွေကတော့ revolute , continuous, prismatic, fixed , floating , planar တို့ဖြစ်ပါတယ် ။ ဥပမာအနေနဲ့ အောက်ကပုံလေးမှာ ကြည့်ပါ ။



```
untitled • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
untitled
1 <joint name="first_joint" type="revolute">
2
3   <!-- Writes your codes here -->
4
5 </joint>
```

အထက်ပါပုံမှာဆိုရင် joint name က first_joint ဖြစ်ပါတယ် ။ Joint type က revolute ပါ ။

Elements of Joint tag

- (i) - **<origin>** tag
- (ii) - **<parent>** tag
- (iii) - **<child>** tag
- (iv) - **<axis>** tag
- (v) - **<limit>** tag
- (vi) - **<dynamics>** tag
- (vii) - **<mimic>** tag
- (viii) - **<safety_controller>** tag
- (ix) - **<calibration>** tag

Tag တစ်ခုချင်းစီကို ဆက်ရှင်းပါမယ်

- (i) - **<origin>** Tag (optional)

ဒီ Tag မှာတော့ ကိုယ်ဖန်တီးတဲ့ robot joint ကို initial pose (position+orientation) ဘယ်နေရာမှာထားမယ်ဆိုတာတွေကို ဒီ Tag အတွင်းမှာရေးရပါတယ် ၊ တနည်းအားဖြင့် joint's reference frame ရဲ့ pose ပေါ့ ။ position အတွက် xyz ဆိုတဲ့ parameter နဲ့ orientation အတွက် rpy ဆိုတဲ့ parameter လေးတွေရှိပါတယ် ။

- (ii) - **<parent>** Tag (required)

ဒီ Tag မှာတော့ Joint မတိုင်ခင် လာမယ့် parent link ရဲ့နာမည်ကို ရေးရပါတယ် ၊ name ဆိုတဲ့ parameter တစ်ခုရှိပါတယ် ။

(iii) - **<child>** Tag (required)

ဒီ Tag မှာတော့ Joint ပြီးရင်လာမယ့် child link ရဲ့နာမည်ကို ရေးရပါတယ် ၊ **name** ဆိုတဲ့ parameter တစ်ခုရှိပါတယ် ။

(iv) - **<axis>** Tag (optional)

ဒီ Tag မှာတော့ မိမိဖန်တီးထားတဲ့ Joint ရဲ့ axis ကို သတ်မှတ်တာပါ ။ **xyz** ဆိုတဲ့ parameter လေးတစ်ခုရှိပါတယ် ၊ မိမိ actuate လုပ်စေချင်တဲ့ axis နေရာမှာ 1 လို့ရေးပေးပြီး ကျန် axis နေရာမှာ 0 လို့ ရေးပေးရမှာပါ ။ Default အနေနဲ့ကတော့ (1,0,0) ပါ ၊ x axis မှာ rotation or translation ဖြစ်မယ်ပေါ့ ။

ရှင်းအောင်ထပ်ပြောရရင်

- revolute joint type အတွက်ဆို axis of rotation ပေါ့
- prismatic joint type အတွက်ဆို axis of translation ဖြစ်ပြီး
- planar joint type အတွက်ဆိုရင်တော့ surface normal ပါ ၊
- တစ်ခု သတိထားရမှာက fixed နဲ့ floating joint types တွေမှာတော့ ဒီ axis tag ကို မသုံးပါဘူး ။

အကယ်၍ ကိုယ်က revolute , prismatic နဲ့ planar joint types တွေ အတွက် axis tag ကို သုံးမယ်ဆိုရင်တော့ ဒီ xyz parameter လေးက required ဆိုတာ သဘောပေါက်မယ်ထင်ပါတယ် ။

(v) - **<limit>** Tag (optional)

ဒီ tag ကတော့ joint ရဲ့ limit values , force နဲ့ velocity ကို ဖော်ပြချင်တဲ့အခါ ရေးရပါတယ် ၊ အဲ့လို ဖော်ပြဖို့အတွက် parameter လေးခု ရှိပါတယ် ၊ အောက်မှ ကြည့်ရင် ပိုရှင်းပါလိမ့်မယ် ။ **lower** ၊ **upper** ၊ **effort** နဲ့ **velocity** တို့ဖြစ်ပါတယ် ။ သူ့ကို revolute နဲ့ prismatic joint types တွေအတွက်သာ သုံးပါတယ် ၊ တခြား joint types တွေ အတွက် မသုံးပါ ။

- **lower** parameter (**optional** , **default = 0**) - ဒီ parameter ကတော့ joint ရဲ့ အနိမ့်ဆုံးရှိနိုင်တဲ့ joint limit value ပါ (revolute type အတွက်ဆို radians unit နဲ့ယူရတာဖြစ်ပြီး prismatic type အတွက်ဆို meter unit နဲ့ယူရပါတယ်) ။
- **upper** parameter(**optional** , **default = 0**) - ဒီ parameter ကတော့ joint ရဲ့အမြင့်ဆုံးရှိနိုင်တဲ့ joint limit value ပါ (revolute type အတွက်ဆို radians unit နဲ့ယူရတာဖြစ်ပြီး prismatic type အတွက်ဆို meter unit နဲ့ယူရပါတယ်) ။
- **effort** parameter(**required**) - ဒီ parameter ကတော့ joint ရဲ့ force or torque values ပါ ၊ ဆိုလိုချင်တာက prismatic type အတွက်ဆို force value ဖြစ်ပြီး revolute type အတွက်ဆို torque value ပါ ။ အကယ်၍ ကိုယ်က limit tag ကို သုံးမယ်ဆိုရင်တော့ ဒီ effort parameter လေးက **required** ဆိုတာ သဘောပေါက်မယ်ထင်ပါတယ် ။
- **velocity** parameter (**required**) - ဒီ parameter ကတော့ joint ရဲ့အမြင့်ဆုံး ရှိနိုင်တဲ့ velocity ပေါ့ ။ ဆိုလိုချင်တာက prismatic type အတွက်ဆို maximum linear velocity ဖြစ်ပြီး revolute type အတွက်ဆို maximum angular velocity ပေါ့ ။ ။ အကယ်၍ ကိုယ်က limit tag ကို သုံးမယ်ဆိုရင်တော့ ဒီ velocity parameter လေးက **required** ဆိုတာ သဘောပေါက်မယ်ထင်ပါတယ် ။

(vi) - **<dynamics>** Tag (**optional**)

ဒီ Tag ကတော့ joint တစ်ခုရဲ့ dynamics နဲ့ဆိုင်တဲ့ physical properties တွေကို သုံးချင်တဲ့အခါ ရေးရပါတယ် ၊ သူ့မှာ parameter နှစ်ခုရှိပါတယ် ၊ damping parameter နဲ့ friction parameter ပါ ။ အောက်မှ အသေးစိတ် ဆက်ရှင်းပါမယ်

- **damping** parameter (**optional** , **default = 0**) - ဒီ parameter ကတော့ Joint ရဲ့ physical damping value ကို သုံးချင်တဲ့အခါ ရေးရပါတယ် ၊ damping value ဆိုတာ တနည်းအားဖြင့် velocity ပေါ်မူတည်ပြီး ဖြစ်ပေါ်နေတဲ့ resistive force တစ်ခုပါပဲ ၊ velocity က resistive force နဲ့ opposite direction ပါ ။ ပြောချင်တာက joint တစ်ခု မြန်မြန်ရွေ့လျားရင် damping value နည်းနေခြင်းဖြစ်ပြီး ၊ joint တစ်ခု နှေးနှေးလေး ရွေ့လျားနေရင် damping value

များနေတယ်လို့ ဆိုလိုတာပါ (heavy damping လို့ခေါ်ပါတယ်) ။ အလွယ်မှတ်ချင်ရင်တော့ joint တစ်ခု မြန်မြန် ရွေ့လျားလား မရွေ့လျားလားဆိုတာ damping value အပေါ် မူတည်တယ်ပေါ့ဗျာ ။ ကျနော်လည်း ဒီလိုနားလည်ထားလို့ ဒီလောက်ပဲ ရှင်းအောင် ပြောပြနိုင်ပါတယ်ဗျာ - ဒီ link နှစ်ခုကြောင့် နားလည်ခဲ့တာပါ (နားလည်တာ မှားရင်လည်း မှားမှာပေါ့)

(<https://robotics.stackexchange.com/questions/10029/damping-vs-friction>)
(<https://www.youtube.com/watch?v=y0YFw9ZzSyM>)

- **friction** parameter (**optional** , **default = 0**) - ဒီ parameter ကတော့ Joint ရဲ့ physical static friction value ကို သုံးချင်တဲ့အခါ ရေးရပါတယ်။ Static friction ဆိုတာကတော့ motion တစ်ခုစတင်ဖြစ်ပေါ်မှုကို resist လုပ်ထားတဲ့ force တစ်ခုပါပဲ ၊ ကျနော်တို့ ကိုးတန်း ဆယ်တန်းမှာ နားလည်ပီးသားတွေပါ ။ ဆက်ရှင်းမပြတော့ဖူးနော် ၊ အကယ်၍ မေ့နေတယ်ဆို ဒီ link မှာ သွားနွေးလို့ရပါတယ်

(<https://www.youtube.com/watch?v=3EbUa5ZDybg&fbclid=IwAR2wHMogVK-gLbrunX0FGblCicQKdPTek2eEJ4LsRhhxftT0KESN0QOKQI>)

(vii) - **<mimic>** Tag (**optional**)

ဒီ tag ကတော့ မိမိ အခုအသစ်ဖန်တီးမယ့် joint ကို ယခင်ရှိပြီးသား existing joint ရဲ့ value အတိုင်းဖြစ်အောင် သတ်မှတ်ချင်ရင် သုံးရပါတယ် ။ joint ရဲ့ value ကို အောက်ပါ formula နဲ့ တွက်နိုင်ပါတယ် ။

$$value = multiplier * other_joint_value + offset$$

joint value ကိုအထက်ပါ formula အတိုင်းတွက်ချက်ဖို့အတွက် parameter သုံးခု ရှိပြီးသားပါ ။ အောက်မှာ ဆက်ကြည့်ပါ ။

- **joint** parameter (**required**) - ဒီ parameter ကတော့ ကိုယ် mimic လုပ်မယ့် existing joint ရဲ့ နာမည်ကို ရေးရပါတယ်။ ။
- **multiplier** parameter (**optional**) - ဒီ parameter ကတော့ အထက်ပါ formula အတိုင်း ကိုယ်လိုချင်တဲ့ joint value ဖြစ်အောင် multiplicative factor သတ်မှတ်တာပါ။ ။
- **offset** parameter(**optional**) - ဒီ parameter ကတော့ အထက်ပါ formula အတိုင်း joint offset ကို သတ်မှတ်တာပါ။ မိမိ အသစ်ဖန်တီးမယ့် joint နဲ့ မူလရှိပြီးသား joint ကြားက offset ပါ။ ။

(viii) - **<safety controller>** Tag (**optional**)

ဒီ Tag မှာတော့ safety controller ကနေ joint ရဲ့ position နဲ့ velocity ကို safety ဖြစ်ချင်ရင် သတ်မှတ်ပေးရပါတယ်။ သဘောကတော့ joint က သူ့သတ်မှတ်တဲ့ limit value ထဲမှာရှိရင် error မရှိနိုင်ဖူးပေါ့။ safe ဖြစ်အောင် boundary သတ်မှတ်တဲ့သဘောပါ။ parameter လေးခုရှိပါတယ်။ အောက်မှာဆက်ကြည့်ပါ။ ။

- **soft_lower_limit** parameter (**optional**) – ဒီ parameter ကတော့ joint position ကို safe ဖြစ်စေမယ့် lower joint boundary သတ်မှတ်တာပါ။ သူက limit tag ကနေ သတ်မှတ်ထားတဲ့ lower parameter ရဲ့ value ထက် ပိုကြီးပေးရပါမယ်။ မရှုပ်သွားပါနဲ့။ limit tag ကနေ သတ်မှတ်တဲ့ lower parameter က joint အနိမ့်ဆုံးရှိနိုင်တဲ့ limit ပါ။ သူ့ထက် နည်းတာနဲ့ error ဖြစ်ပါပြီ။ အခု tag ရဲ့ parameter က safe ဖြစ်စေမယ့် boundary ပါ။ ။
- **soft_upper_limit** parameter (**optional**) – ဒီ parameter ကတော့ joint position ကို safe ဖြစ်စေမယ့် upper joint boundary သတ်မှတ်တာပါ။ သူက limit tag ကနေ သတ်မှတ်ထားတဲ့ upper parameter ရဲ့ value ထက် ပိုငယ် ပေးရပါမယ်။ မရှုပ်သွားပါနဲ့။ limit tag ကနေ သတ်မှတ်တဲ့ upper parameter က joint အမြင့်ဆုံးရှိနိုင်တဲ့ limit ပါ။ သူ့ထက် ကြီးတာနဲ့ error ဖြစ်ပါပြီ။ အခု tag ရဲ့ parameter က safe ဖြစ်စေမယ့် boundary ပါ။ ။

- **k_position** parameter (**optional**) – ဒီ parameter ကတော့ position ကို safety limit ကနေ ကျော်မသွားအောင် boundary ဝင်အောင် velocity limit ကို scale လုပ်ပေးတဲ့အခါ ရေးရပါတယ်။
- **k_velocity** parameter (**optional**) – ဒီ parameter ကတော့ velocity ကို safety limit ကနေ ကျော်မသွားအောင် boundary ဝင်အောင် effort limit ကို scale လုပ်ပေးတဲ့အခါ ရေးရပါတယ်။

ဒီ Parameter လေးခုကို ပိုရှင်းသွားအောင် အောက်က ပုံနှစ်ပုံမှာ ကြည့်နိုင်ပါတယ် ။

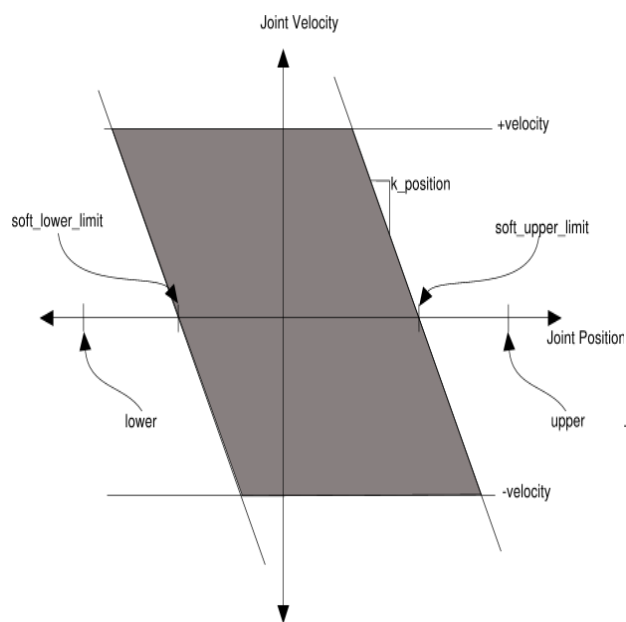


Fig : k_velocity

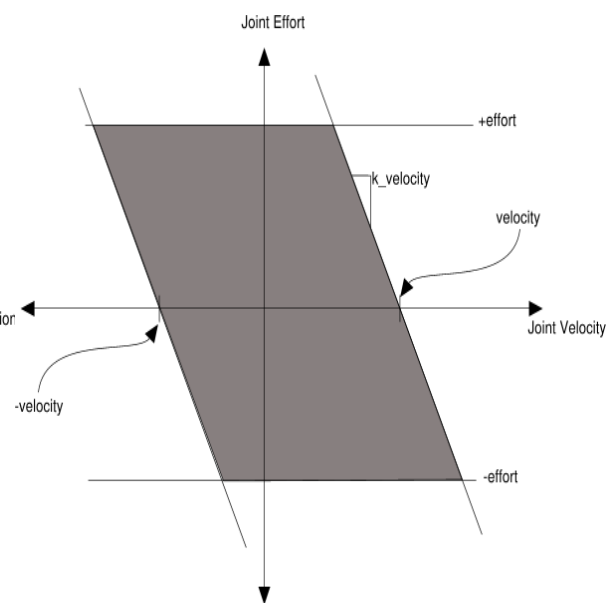


Fig : k_position

(viii) - **<calibration>** Tag (optional)

ဒီ Tag ကို ကျနော် သေချာကြီး နားမလည်ဘူးဗျ။ ခု

ကျနော်နားလည်ထားသလောက်ပြောရရင် joint accuracy ကို improve ဖြစ်အောင်

ရေးချင်တဲ့အခါ သုံးတယ်လို့ပဲ သိပါတယ် ၊ သူ့မှာ **rising** နဲ့ **falling** ဆိုပြီး parameter နှစ်ခုရှိပါတယ်

။ ကျနော်လည်း နားမလည်တော့ ရမ်းမရွှီးချင်ပါဘူး ၊ ros wiki မှာတော့

ဒီလိုလေးရေးထားပါတယ် ။

<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.

အပေါ်မှာ ရှင်းပြခဲ့တဲ့ joint tag တစ်ခုလုံးကို ရှင်းရှင်းလင်းလင်း ဒီ အောက်က example မှာ ကြည့်နိုင်ပါတယ် ။

```
untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

<< untitled
1 <joint name="first_joint" type="revolute">
2
3   <origin xyz="0 0 1" rpy="0 0 3.1416"/>
4   <parent link="first_link"/>
5   <child link="second_link"/>
6   <axis xyz="0 0 1" />
7
8   <calibration rising="0.0"/>
9   <dynamics damping="0.0" friction="0.0"/>
10  <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
11  <safety_controller k_velocity="10" k_position="15" soft_lower_limit="-2.0" soft_upper_limit="0.5" />
12
13 </joint>
```

4 – sensor tag

Syntax - `<sensor>.....</sensor>`

ဒီ Tag နှစ်ခုအတွင်းမှာတော့ မိမိသုံးမယ့် sensor နဲ့ပတ်သက်တဲ့ properties တွေ information တွေကို ရေးရပါတယ်။ အောက်မှာဆက်ကြည့်ရင် ပိုရှင်းသွားပါလိမ့်မယ်။ မိမိသုံးမယ့် sensor အပေါ်မူတည်ပြီး element တွေ မတူညီပါဘူး။ ဒီ link မှာသွားကြည့်နိုင်ပါတယ် (<http://wiki.ros.org/urdf/XML/sensor/proposals>) ၊ ကျနော်ကတော့ camera sensor ကို ဥပမာပေးပြီး ပြောမှာဖြစ်တာကြောင့် camera tag ကိုပဲ သုံးထားပါတယ်။

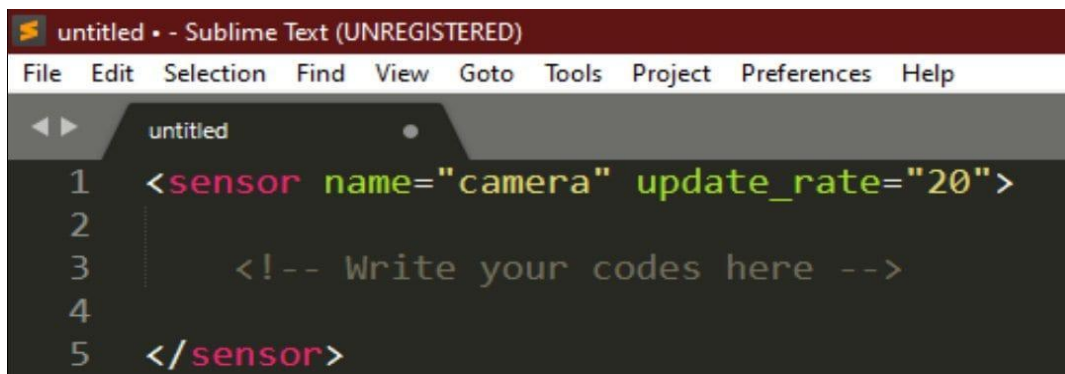
Attribute of sensor tag

1 – name (required)

name ဆိုတာကတော့ မိမိ robot မှာ တပ်ဆင်မယ့် sensor နာမည်ပါ။

2 – update_rate (optional)

update_rate ဆိုတာကတော့ မိမိ sensor က ဘယ်လောက် frequency မှာ sensor data တွေကို generate လုပ်မလဲဆိုတာ သတ်မှတ်ပေးရတာပါ။ frequency ဖြစ်လို့ unit က hz ပါ။ ဒါကြောင့် ဘယ်နှစက္ကန့်တစ်ခါ generate လုပ်မလဲဆိုတာကိုသိချင်ရင် ဒီ formula ($t = 1/f$) ထဲထည့်ပြီး တွက်နိုင်ပါတယ်။



```
untitled • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
untitled
1 <sensor name="camera" update_rate="20">
2
3 <!-- Write your codes here -->
4
5 </sensor>
```

အထက်ပါ ပုံအရဆို မိမိသုံးမယ့် sensor က camera ဖြစ်တာကြောင့် နာမည်ကို camera လို့ပေးထားပြီး 20hz တစ်ခါ data တွေကို generate လုပ်ပေးနေမှာပါ။

Elements of sensor tag

- (i) - **<parent>** tag
- (ii) - **<origin>** tag
- (iii) - **<camera>**.....**</camera>** tag

Tag တစ်ခုချင်းစီကို ဆက်ရှင်းပါမယ် ။

- (i) - **<parent>** Tag (required)

ဒီ tag မှာတော့ မိမိ သုံးမယ့် sensor ကို ဘယ် link မှာ တပ်ဆင်မယ်ဆိုတာကို link name လေးရေးပေးရမှာပါ ။ link parameter လေးတစ်ခုရှိပါတယ် ။ parameter လေးက required ပါ ။

- (ii) - **<origin>** Tag (optional)

ဒီ Tag မှာတော့ မိမိ သုံးမယ့် sensor ကို initial pose (position+orientation) ဘယ်နေရာမှာ ထားမယ်ဆိုတာတွေကို ဒီ Tag အတွင်းမှာရေးရပါတယ် ၊ တနည်းအားဖြင့် sensor` s reference frame ရဲ့ pose ပေါ့ ။ position အတွက် xyz ဆိုတဲ့ parameter နဲ့ orientation အတွက် rpy ဆိုတဲ့ parameter လေးတွေရှိပါတယ် ။ frame ယူတဲ့နေရာမှာ သတိထားရမှာက sensor ဖြစ်တဲ့အတွက်ကြောင့် z axis ကို forward ၊ x axis ကို right ဖြစ်ပြီး y axis ကတော့ down ဆိုပြီး ယူပေးရမှာပါ ။

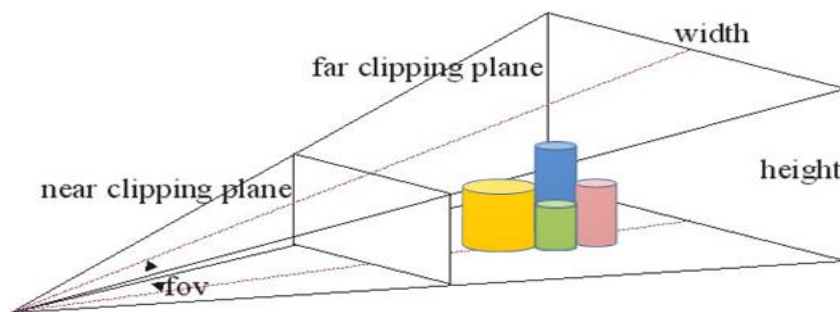
- (iii) - **<camera>**.....**</camera>** Tag (optional)

အကယ်၍ မိမိက camera sensor ကိုသုံးမယ်ဆိုရင် ဒီ Tag လေးအတွင်းမှာ image နဲ့ ဆိုင်တဲ့ information တွေကို image tag လေးအတွင်းမှာရေးပေးရမှာပါ ။ parameter အနေနဲ့ ၆ ခုရှိပါတယ် ။ အောက်မှ အသေးစိတ်ဆက်ကြည့်ပါ ။

မိမိဖန်တီးထားတဲ့ camera sensor ကို တကယ့် real sensor လိုမျိုး အသုံးပြုနိုင်ဖို့ plugin file တစ်ခုထည့်ဖို့လိုတယ်ဆိုတာ သတိချပ်စေချင်ပါတယ်။ ။

- **width** parameter (**required**) – ဒီ parameter ကတော့ image တစ်ခုရဲ့ pixels width ကိုရေးပေးရမှာပါ။ ။
- **height** parameter (**required**) – ဒီ parameter ကတော့ image တစ်ခုရဲ့ pixels height ကိုရေးပေးရမှာပါ။ ။
- **format** parameter (**required**) – ဒီ parameter ကတော့ camera ရဲ့ image format ကိုရေးပေးရမှာပါ။ ။ image encodings တွေပေါ့။ ။ rgb လား bgr လား စသဖြင့်။ ။
- **hfov** parameter (**required**) – ဒီ parameter ကတော့ camera ရဲ့ horizontal field of view ပါ။ ။ မြင်အောင်ပြောရရင် camera ကနေ မြင်နိုင်မယ့် data ဖတ်နိုင်မယ့် horizontal range ပေါ့။ ။ unit က radians ပါ။ ။
- **near** parameter (**required**) – ဒီ parameter ကတော့ camera ရဲ့ near clip distance ပါ။ ။ စာနဲ့ ပြောပြဖို့တော့ ခက်တယ်ဗျ - အောက်က ပုံမှာကြည့်ရင် သိသွားပါလိမ့်မယ်။ ။ near clipping plane ပါ။ ။
- **far** parameter (**required**) – ဒီ parameter ကတော့ camera ရဲ့ far clip distance ပါ။ ။ စာနဲ့ ပြောပြဖို့တော့ ခက်တယ်ဗျ - အောက်က ပုံမှာကြည့်ရင် သိသွားပါလိမ့်မယ်။ ။ far clipping plane ပါ။ ။

အပေါ်က parameter အကုန်လုံးကို လက်တွေ့မြင်နိုင်ဖို့ ဒီအောက်ကပုံလေးက ကူညီပေးပါလိမ့်မယ်။ ။



Source - google

အပေါ်မှာ ရှင်းပြခဲ့တဲ့ sensor tag တစ်ခုလုံးကို ရှင်းရှင်းလင်းလင်း ဒီ အောက်က example မှာ ကြည့်နိုင်ပါတယ် ။

```
untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

untitled
1 <sensor name="camera" update_rate="20">
2
3   <parent link="attached_link"/>
4
5   <origin xyz="0 0 0" rpy="0 0 0"/>
6
7   <camera>
8     <image width="640" height="480" hfov="1.5708" format="RGB8" near="0.01" far="50.0"/>
9   </camera>
10
11 </sensor>
```

ros_control with Gazebo

Contents

- 1. Introduction**
- 2. Types of ROS Controllers**
- 3. Types of Hardware Interfaces**
- 4. ros_control Package**
- 5. Most common used ROS Controller Packages**
- 6. ros_control with Gazebo**
 - a. What is Transmission ?**
 - b. Transmission Tag in URDF**
 - c. Add gazebo_control_plugin to a URDF**
 - d. Example using RRBot**
 - i. Create YAML file for ROS Controllers**
 - ii. Create Launch file to load ROS Controllers and Manager**

Gazebo ထဲမှာ Robot Model ကို control လုပ်ဖို့အတွက် ROS controller တွေ လိုအပ်ပါတယ် ၊ ရှင်းအောင်ပြောရရင် arm မှာပဲဖြစ်ဖြစ် mobile robot မှာပဲဖြစ်ဖြစ် model ရဲ့ joint တွေကို control လုပ်ဖို့အတွက် controller ရှိဖို့လိုတယ်လို့ ပြောချင်တာပါ ။ joint တွေကို control လုပ်ဖို့ဆိုရင် transmission tag ထဲမှာ သတ်မှတ်ထားတဲ့ hardware interface နဲ့ အဆင်ပြေမယ့် controller တွေကို လိုအပ်ပါတယ်။

1 - Introduction

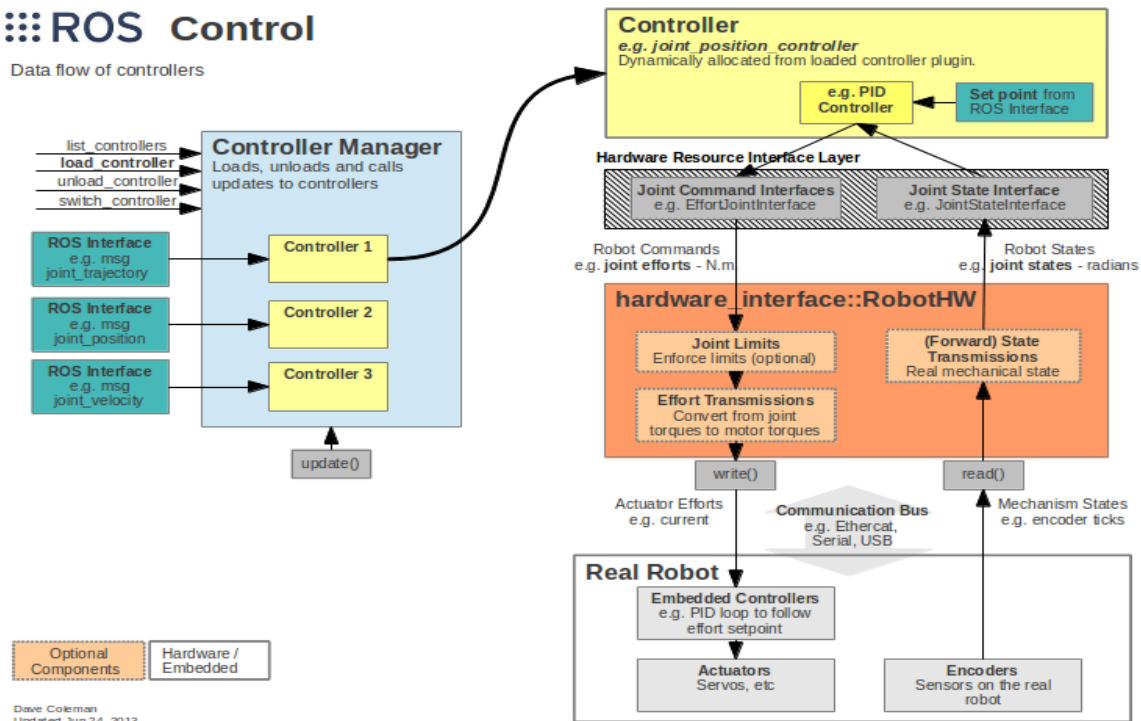
Controller ဆိုတာကတော့ မိမိ robot ရဲ့ joint ကို position ၊ velocity ၊ effort(force or torque) ၊ joint states ၊ joint trajectory စသည်တို့ကို control လုပ်ပေးတဲ့အရာပါ ။ position ကို control လုပ်ဖို့ဆို joint_position_controller ဆိုတဲ့ package လေးရှိတယ်ပေါ့ ၊ အပေါ်မှာပြောခဲ့တဲ့ အရာတွေလည်း ထိုနည်းဌာနပါပဲ ။ မြင်အောင် ထပ်ပြီး ဥပမာပေးရရင် PID Controller ပေါ့ ။

ROS Controller တွေက hardware ကို တိုက်ရိုက် control လုပ်လို့မရပါဖူး ၊ ကြားခံ mediator အနေနဲ့ controller ကနေ real hardware သို့ fake hardware(in simulation) ကို ဆက်သွယ်ဖို့ interface တစ်ခုလိုအပ်ပါတယ် ၊ controller ကနေ ပေးလာတဲ့ command ကို receive လုပ်ပြီး hardware ဆီကို send လုပ်ပေးနိုင်တဲ့ interface ပေါ့ (Note – Vice Versa) ။ ခုနက အပေါ်မှာပြောခဲ့တဲ့ transmission tag ထဲက hardware interface ဆိုလိုတာပါ။

မိတ်ဆက်အနေနဲ့ မြင်အောင်ပြောပြတာပါ ။ အောက်ကပုံကို ကြည့်လိုက်ရင် နည်းနည်းမြင်သာသွားပါလိမ့်မယ် ။

ROS Control

Data flow of controllers



Source – gazebo.org

Controller အမျိုးအစားတွေနဲ့ Hardware Interface အမျိုးအစားတွေအကြောင်း ဆက်ရှင်းပါမယ် ။

2 - Types of ROS Controllers

1. **effort controllers** - မိမိ robot ရဲ့ joints တွေဆီ desired force or torque ကို command ပေးနိုင်တဲ့ controllers တွေပါ ။ command ပေးတဲ့ပုံစံအပေါ်မူတည်ပြီးတော့လည်း သုံးရတဲ့ package တွေက ကွဲပြားပါတယ် ။ အောက်မှာဆက်ပြောမယ့် controller package တွေနဲ့ မရောသွားစေဖို့ မှတ်ထားရမှာက ဒီ controller အမျိုးအစားမှာပါတဲ့ ဘယ်လို controller ပဲ ဖြစ်ဖြစ် သူ့ရဲ့ နောက်ဆုံး desired output က force or torque ဖြစ်တယ်ဆိုတာပါ ။ ငါးခုရှိပါတယ် ။ အောက်မှာကြည့်ပါ ။

- joint_effort_controller
- joint_group_effort_controller
- joint_group_position_controller
- joint_position_controller
- joint_velocity_controller

2. **position_controllers** - မိမိ robot ရဲ့တစ်ခု သို့ တစ်ခုထက်ပိုတဲ့ joint တွေရဲ့ position တွေကို သတ်မှတ်ပေးလို့ရတဲ့ controller တွေပါ။ နှစ်ခုရှိပါတယ် ။ အောက်မှာကြည့်ပါ ။

- joint_position_controller
- joint_group_position_controller

3. **velocity_controllers** - မိမိ robot ရဲ့တစ်ခု သို့ တစ်ခုထက်ပိုတဲ့ joint တွေရဲ့ velocity တွေကို သတ်မှတ်ပေးလို့ရတဲ့ controller တွေပါ ။ နှစ်ခုရှိပါတယ် ။ အောက်မှာကြည့်ပါ ။

- joint_velocity_controller
- joint_group_velocity_controller

4. **joint_state_controller** - မိမိ robot ရဲ့ states ကို publisher လုပ်ပေးတဲ့ controller ပါ။ တစ်ခုပဲရှိပါတယ် ။ သူကိုယ်တိုင်ပါပဲ ။

- joint_state_controller

5. **joint_trajectory_controller** - မိမိ robot ရဲ့ joint သွားမယ့် trajectory (path+time) ကို သတ်မှတ်ပေးတဲ့ controller ပါ ။ ငါးခုရှိပါတယ် ။ အောက်မှာကြည့်ပါ ။

- position_controller
- velocity_controller
- effort_controller
- position_velocity_controller
- position_velocity_acceleration_controller

အထက်ပါ Controller တွေကို အသုံးပြုဖို့အတွက် Plugins တွေကို ဒီ Link မှာ (https://github.com/ros-controls/ros_controllers) ဒေါင်းလို့လည်းရသလို terminal ကနေလည်း install လို့ရကြောင်းပါ ။

Enter following line in terminal to install ,

```
sudo apt-get install ros-noetic-ros-control ros-noetic-ros-controllers
```

3 - Types of Hardware Interface

1. **Joint Command Interface** - ဒီ Interface ကတော့ controller ကနေ command ပေးလိုက်တဲ့ command အမျိုးအစားပေါ်မူတည်ပြီး hardware joint တွေဆီကို send လုပ်ပေးပါတယ် ။ command အမျိုးအစားဆိုတာကတော့ အောက်မှာပါတဲ့ သုံးခုပေါ့ ။
 - **Effort Joint Interface** - for commanding effort-based joints.
 - **Velocity Joint Interface** - for commanding velocity-based joints.
 - **Position Joint Interface** - for commanding position-based joints.
2. **Joint State Interfaces** - ဒီ Interface ကတော့ hardware ဆီကနေ feedback ပြန်ရမယ့် Joint States တွေကို controller ဆီ ပြန်ပို့ပေးတာပါ ။ ဥပမာနဲ့ မြင်အောင်ပြောရရင် PID Controller ပါ ။ controller နဲ့ သုံးမယ့် real or fake hardware ကြား loop ဖြစ်နေတာပါ ။
3. **Actuator Command Interfaces** - ဒီ Interface ကတော့ Actuator အတွက် Hardware Interface ဖြစ်ပါတယ် ၊ အပေါ်မှာပြောခဲ့တဲ့ Joint Command Interface နဲ့ အများအားဖြင့် တူတူပဲရေးပါတယ် ။ အမျိုးအစားဆိုတာကတော့ အောက်မှာပါတဲ့ သုံးခုပေါ့ ။
 - **Effort Actuator Interface**
 - **Velocity Actuator Interface**
 - **Position Actuator Interface**

ဒီ အောက်က interface တွေကတော့ မိမိသုံးတဲ့အပေါ်မူတည်ပြီး ကွဲပြားမှာဖြစ်တဲ့အတွက် အသေးစိတ်မပြောတော့ပါဘူး (ပျင်းလို့) ။

4. Actuator State Interfaces

5. Force-torque sensor Interface

6. IMU sensor Interface

အမှန်တော့ ကျနော် ခုပြောခဲ့တဲ့ Types of Controllers နဲ့ Types of Hardware Interface တွေက အပေါ်ကပြခဲ့တဲ့ ပုံကို နားလည်တာနဲ့ အကုန်လုံး ရှင်းရှင်းလင်းလင်းနဲ့ကို သိသွားမှာပါ ။

အပေါ်မှာ သေချာပြောပြခဲ့တဲ့ ROS controller နဲ့ Hardware Interface တွေ ကို Gazebo မှာ အသုံးချနိုင်ဖို့ဆို ROS မှာရှိတဲ့ **ros_control package** ကို နားလည်ဖို့လိုပါတယ် ။

4 - ros_control Package

ros_control ဆိုတာကတော့ metapackage ကြီးပါ။ သူ့ထဲမှာ တခြားသော packages တွေ ပါဝင်ပါတယ်။

1. **control_toolbox** - ဒီ Package ကတော့ controller တွေရေးတဲ့အခါ သုံးတဲ့ tools တွေ ပါဝင်ပါတယ်။
2. **controller_interface** - ဒီ Package ကတော့ controller တွေအတွက် interface base class ပါဝင်ပါတယ်။
3. **controller_manager** - ဒီ Package ကတော့ controller တွေကို load,unload,start,stop လုပ်တာပါ။ ။ရှင်းအောင်ပြောရရင် controller တွေကို ခိုင်းတဲ့ package ပါ။
4. **controller_manager_msgs** - ဒီ Package ကတော့ controller_manager အတွက် လိုအပ်တဲ့ message တွေ service တွေကို provide လုပ်ပေးပါတယ်။
5. **hardware_interface** - ဒီ package ကတော့ အပေါ်မှာ ရှင်းပြခဲ့ပြီးလို့ သိပြီထင်ပါတယ် ၊ controller တွေက command ပေးတာကို real hardware သို့ fake hardware(in simulation) ကိုလုပ်ဆောင်နိုင်ဖို့ ကြားခံအဖြစ် ဆောင်ရွက်ပေးတဲ့ Package ပါ။ (**Note – Vice Versa**)
6. **transmission_interface** - ဒီ Package ကတော့ hardware interface အတွက်လိုအပ်တဲ့ interface classes တွေပါဝင်ပါတယ်။

Package တွေများလို့ စိတ်ညစ်မသွားပါနဲ့ ၊ ကျနော်က ပြည့်ပြည့်စုံစုံဖြစ်အောင်လို့ ပြောပြတာပါ။ ။

Coding မှာအဓိကသုံးမယ့် controllers တွေနဲ့ဆိုင်တဲ့ အရေးကြီးတဲ့ Package တွေကို ပြောပါမယ် ။
အောက်မှာကြည့်ပါ ။

5 - ROS Controller Packages

1. **joint_position_controller** - ဒီ Package ကိုရှင်းမယ်ထင်ပါတယ် ၊ အပေါ်မှာပြောခဲ့တဲ့ Types of Controllers မှာပါတဲ့ **position_controllers** အမျိုးအစားပါ ။
2. **joint_state_controller** - ဒီ Package ကိုရှင်းမယ်ထင်ပါတယ် ၊ အပေါ်မှာပြောခဲ့တဲ့ Types of Controllers မှာပါတဲ့ **joint_state_controller** အမျိုးအစားပါ ။
3. **joint_effort_controller** - ဒီ Package ကိုရှင်းမယ်ထင်ပါတယ် ၊ အပေါ်မှာပြောခဲ့တဲ့ Types of Controllers မှာပါတဲ့ **efforts_controllers** အမျိုးအစားပါ ။

Program ပိုင်းကို သေသေချာချာ နားလည်နိုင်ဖို့အတွက် လိုအပ်တဲ့ Information မှန်သမျှကို ရှင်းပြပြီးပြီဖြစ်တဲ့အတွက် Gazebo မှာ မိမိတို့ Robot Model ကို ဘယ်လို Control လုပ်မလဲဆိုတာကို ဆက်ရှင်းပါမယ်

6 - ros_control with Gazebo

Gazebo ထဲမှာ robot_controllers တွေကို ကျနော်တို့ အပေါ်မှာ အသေးစိတ်ပြောခဲ့တဲ့ ros_control နဲ့ အသုံးပြုပြီး robot ရဲ့ joint တွေကို actuate လုပ်စေမှာဖြစ်ပါတယ်။

မှတ်ချက် - Gazebo သည် Standalone Simulation Software ဖြစ်တဲ့အတွက် သေချာလေ့လာမယ်ဆို အချိန်တွေအများကြီးပေးရမှာပါ။ ကျနော်တို့ ခု ဒီ pdf မှာတော့ ROS နဲ့အတူ Simulation လုပ်ဆောင်နိုင်ဖို့အတွက် လိုအပ်တဲ့ Information တွေနဲ့ example ကိုပဲရှင်းပြပေးသွားမှာပါ။ စိတ်ပါရင် ကိုယ့်ဘာသာ further study လုပ်ကြည့်ပါ။

6.a – What is Transmission ?

ဒီ Tag ကတော့ ကျနော်တို့ robot မှာသုံးမယ့် Actuator(Motor) နဲ့ Joint ကို link လုပ်ပေးတာပဲ ဖြစ်ပါတယ်။ ပြီးတော့ mechanical coupling ကိုလည်း ဖော်ပြပေးပါတယ်။

ဒီနေရာမှာ Joint နဲ့ Actuator မတူဖူးလားလို့မေးစရာရှိပါတယ်။ Actuator ကို Motor လို့နောက်မှာ ဆက်သုံးပါမယ်။ Motor ဆိုတာ Joint ကို actuate လုပ်တဲ့အရာဖြစ်ပါတယ်။ Joint က actuate ဖြစ်မှ Joint နဲ့ချိတ်ဆက်ထားတဲ့ Robot Links တွေက ရွေ့လျားမှာဖြစ်ပါတယ်။

Motor က Joint ကို actuate လုပ်ဖို့ဆို coupling လုပ်ဖို့လိုပါတယ်။ coupling ဆိုတာကတော့ motor နဲ့ joint ကြားမှာထားတဲ့ component တစ်ခုပဲ ဖြစ်ပါတယ်။ သေချာသိချင်ရင် Google or YouTube မှာ ရှာကြည့်နိုင်ပါတယ်။ ဒီ link လေးမှာ ပြောပြတာလေးကတော့ ကောင်းတယ်ဗျ

(<https://www.smlease.com/entries/mechanism/what-is-mechanical-coupling-types-applications/#:~:text=Mechanical%20Coupling%20are%20used%20to%20connect%20driver%20and,rear%20axle%20and%20connected%20using%20a%20universal%20joint.>)

ရည်ရွယ်ချက်ကတော့ Power Transmission လုပ်ဖို့ပါ။ ။

Power ရဲ့ equation က $P = W/t$

$W = Fd$ ဖြစ်တဲ့အတွက် $P = Fd/t$

d/t က velocity ဖြစ်တဲ့အတွက် နောက်ဆုံး Equation က

$$P = F \times v$$

ပြောချင်တာကတော့ Power Transmission လုပ်တယ်ဆိုတာက Effort (force or torque) နဲ့ velocity တွေကို joint controller တွေဆီ transmit လုပ်ပေးတာပါ ၊ ဒါမှ effort-based controller တွေ velocity, position based controller တွေကို လိုသလို အသုံးပြုနိုင်မှာပါ။ ။

6.b – Transmission Tag in URDF

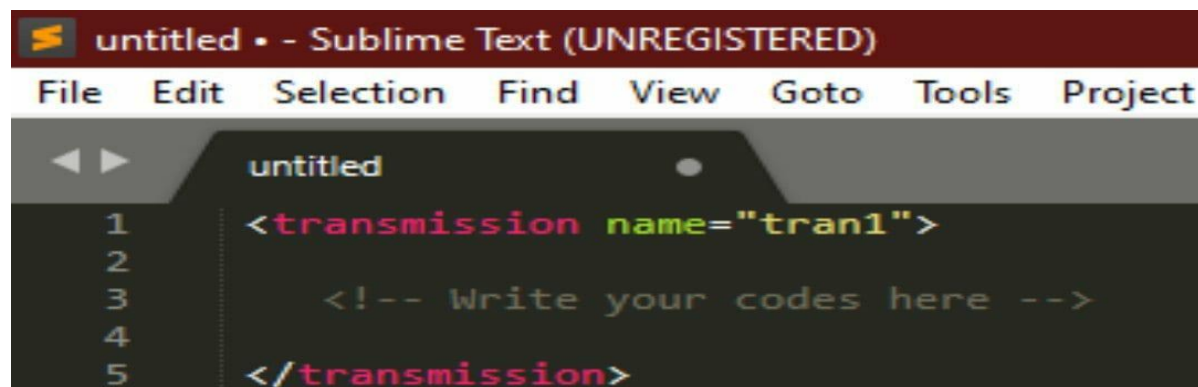
Syntax - `<transmission>.....</transmission>`

အပေါ်မှာရှင်းပြခဲ့တဲ့ ရည်ရွယ်ချက်နဲ့ Transmission Tag ကိုသုံးရခြင်းဖြစ်ပါတယ်။ ။

Attribute of Transmission Tag

1. name

(name ဆိုတာကတော့ မိမိသုံးမယ့် transmission ရဲ့ နာမည်ပါ ၊ ဥပမာအနေနဲ့ အောက်ကပုံလေးမှာ ကြည့်ပါ)



```
untitled • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project

untitled
1 <transmission name="tran1">
2
3   <!-- Write your codes here -->
4
5 </transmission>
```

Elements of Transmission Tag

- (i) - **<type>**.....**</type>** tag
- (ii) - **<joint>**.....**</joint>** tag
- (iii) - **<actuator>**.....**</actuator>** tag

- (i) - **<type>**.....**</type>** Tag (one occurrence)

ဒီ Tag မှာတော့ မိမိသုံးမယ့် Transmission အမျိုးအစားကို ရေးပေးရမှာပါ။ လောလောဆယ် ရှိတာကတော့ transmission_interface/SimpleTransmission ပါ။ ကျနော် လောလောဆယ် ဒါပဲတွေ့ဖူးပါတယ်။

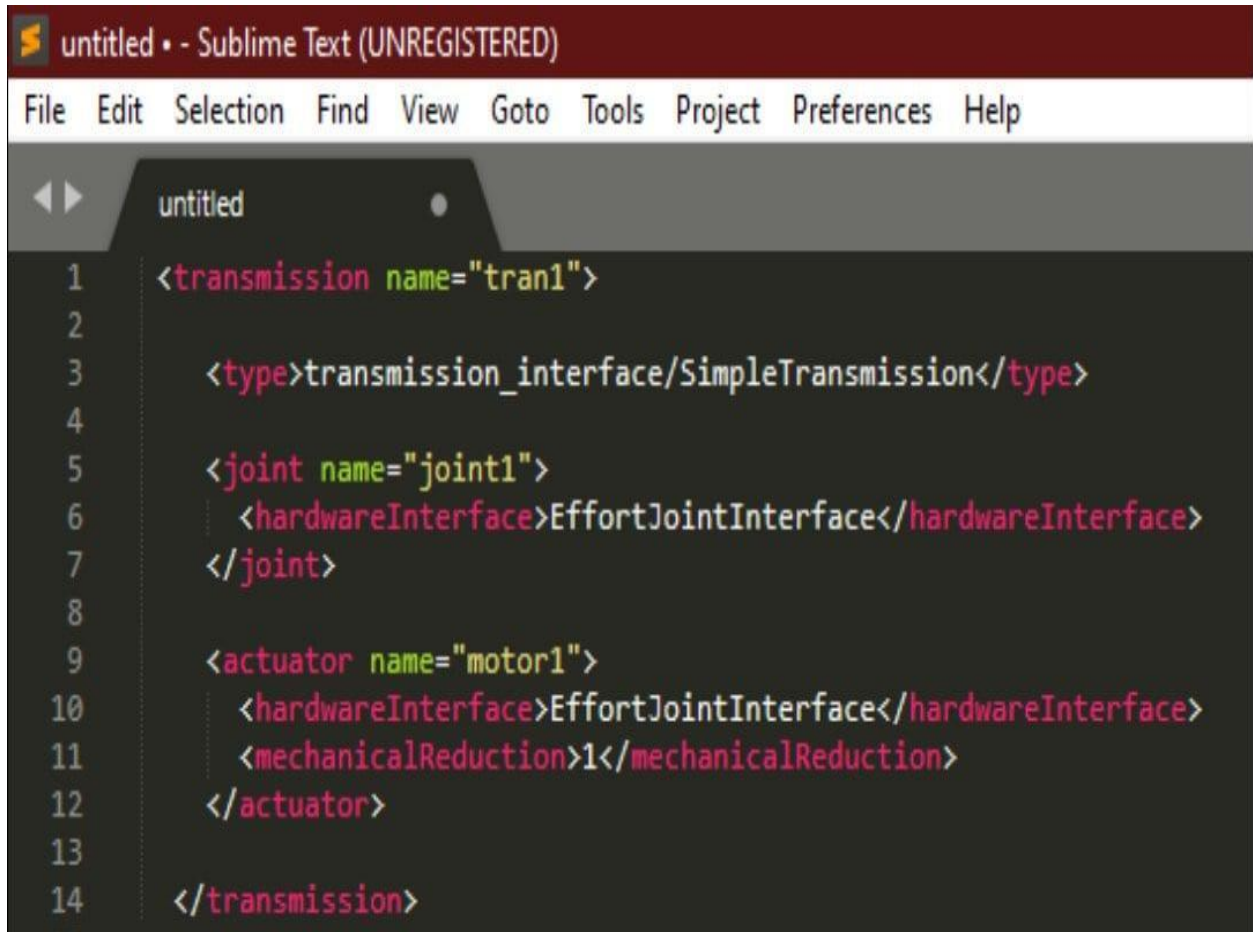
- (ii) - **<joint>**.....**</joint>** Tag (one or more occurrences)

ဒီ Tag မှာတော့ မိမိ transmission ကို အသုံးချမယ့် joint နဲ့ hardware interface ကို ရေးပေးရမှာပါ။ hardware interface ကို အပေါ်မှာ သေချာရှင်းပြပြီးပါပြီ။ ထိုသို့ အသုံးချဖို့ joint name အတွက် name ဆိုတဲ့ parameter နဲ့ hardware interface အတွက် **<hardwareInterface>**.....**<hardwareInterface>** ဆိုတဲ့ tag လေးသေးသေးလေးရှိပါတယ်။

- (iii) - **<actuator>**.....**</actuator>** Tag (one or more occurrences)

ဒီ Tag မှာတော့ မိမိ transmission ကို အသုံးချမယ့် actuator နဲ့ hardware interface ကို ရေးပေးရမှာပါ။ hardware interface ကို အပေါ်မှာ သေချာရှင်းပြပြီးပါပြီ။ ထိုသို့ အသုံးချဖို့ actuator name အတွက် name ဆိုတဲ့ parameter နဲ့ hardware interface အတွက် **<hardwareInterface>**.....**</hardwareInterface>** ဆိုတဲ့ tag လေးသေးသေးလေးရှိပါတယ်။ သူ့မှာ **<mechanicalReduction>**.....**</mechanicalReduction>** ဆိုတဲ့ tag လေးလည်းရှိသေးတယ်။ Mechanical Reduction က ဘာလဲ ကျနော် သေချာမသိလို့ ရမ်းမချွီးချင်ပါဖူး။

အပေါ်မှာ ရှင်းပြခဲ့တဲ့ transmission tag တစ်ခုလုံးကို ရှင်းရှင်းလင်းလင်း ဒီအောက်က example မှာ ကြည့်နိုင်ပါတယ် ။



```
untitled • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

<> untitled
1 <transmission name="tran1">
2
3   <type>transmission_interface/SimpleTransmission</type>
4
5   <joint name="joint1">
6     <hardwareInterface>EffortJointInterface</hardwareInterface>
7   </joint>
8
9   <actuator name="motor1">
10    <hardwareInterface>EffortJointInterface</hardwareInterface>
11    <mechanicalReduction>1</mechanicalReduction>
12  </actuator>
13
14 </transmission>
```

6.c – Add gazebo_control_plugin to URDF

gazebo tag ထဲမှာ tag များစွာရှိသော်လည်း ဒီ pdf မှာတော့ အဓိကသုံးမယ့် plugin tag ကိုပဲ ပြောသွားမှာပါ။

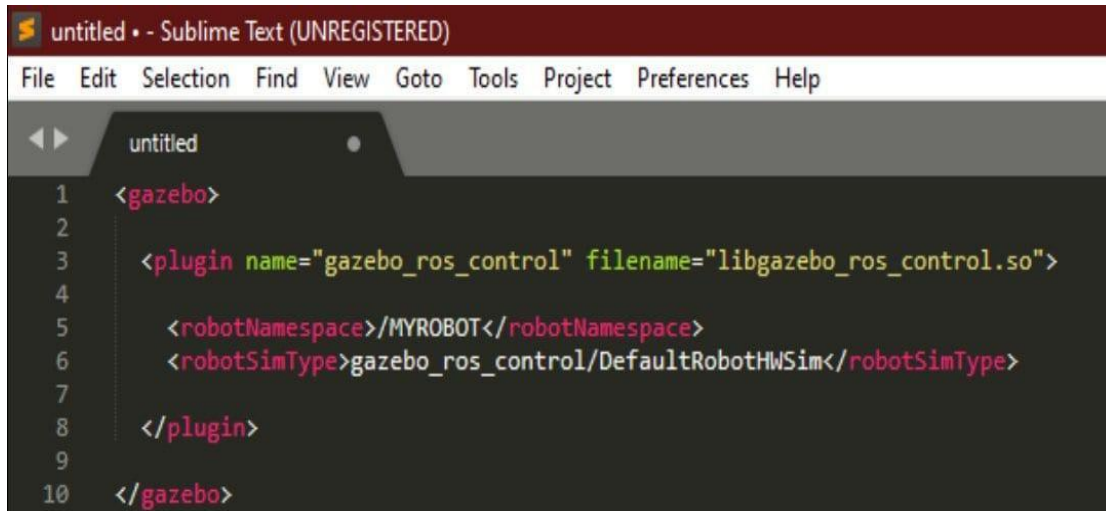
<plugin>.....**</plugin>** Tag

ဒီမှာတော့ controller manager နဲ့ transmission tag ထဲမှာ သတ်မှတ်ခဲ့တဲ့ hardware interface ကို load လုပ်ပေးမယ့် plugin အကြောင်းပြောမှာပါ။ သူ့မှာ child tag လေး လေးခုရှိပါတယ်။ သူ့မှာ name ဆိုတဲ့ parameter နဲ့ filename ဆိုတဲ့ parameter နှစ်ခုရှိပါတယ်။ name parameter ကတော့ plugin name ဖြစ်ပြီး filename parameter ကတော့ အသုံးချမယ့် plugin ရဲ့ so file ဖြစ်ပါတယ်။ တနည်းအားဖြင့် controller file ပါပဲ။ မိမိအသုံးချမယ့် robot အပေါ်မူတည်ပြီး controller file ကွဲပြားသွားမှာပါ။

- **<robotNamespace>**.....**</robotNamespace>** - ဒီ child tag မှာတော့ မိမိ robot name ကို ရေးပေးရမှာပါ။ URDF အစမှာရေးခဲ့တဲ့ root tag ကြီးဖြစ်တဲ့ robot tag ရဲ့ name ပါ။
- **<controlPeriod>**.....**</controlPeriod>** ဒီ child tag မှာတော့ မိမိ အသုံးချမယ့် controller ကို update လုပ်မယ့် period ကိုရေးပေးရမှာပါ။ unit က second ပါ။
- **<robotParam>**.....**</robotParam>** ဒီ child tag မှာတော့ မိမိတို့ parameter server ပေါ်မှာရှိတဲ့ robot_description ရဲ့ location ကိုရေးရမှာပါ။ launch file ထဲမှာရေးပေးနေကြ `"/robot_description"` ပါပဲ။
- **<robotSimType>**.....**</robotSimType>** မိမိတို့ robot ကို simulation မယ့် interface pluginlib name ပါ။ default ကတော့ `"DefaultRobotHWSim"` ပါ။ သေချာသိချင်ရင် ဒီ link မှာ သွားကြည့်နိုင်ပါတယ်

(http://gazebosim.org/tutorials/?tut=ros_control#Defaultgazebo_ros_controlBehavior)

အပေါ်မှာ ရှင်းပြခဲ့တဲ့ plugin tag အသုံးချပုံကို ရှင်းရှင်းလင်းလင်း ဒီအောက်က example မှာ ကြည့်နိုင်ပါတယ် ။



```
untitled - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

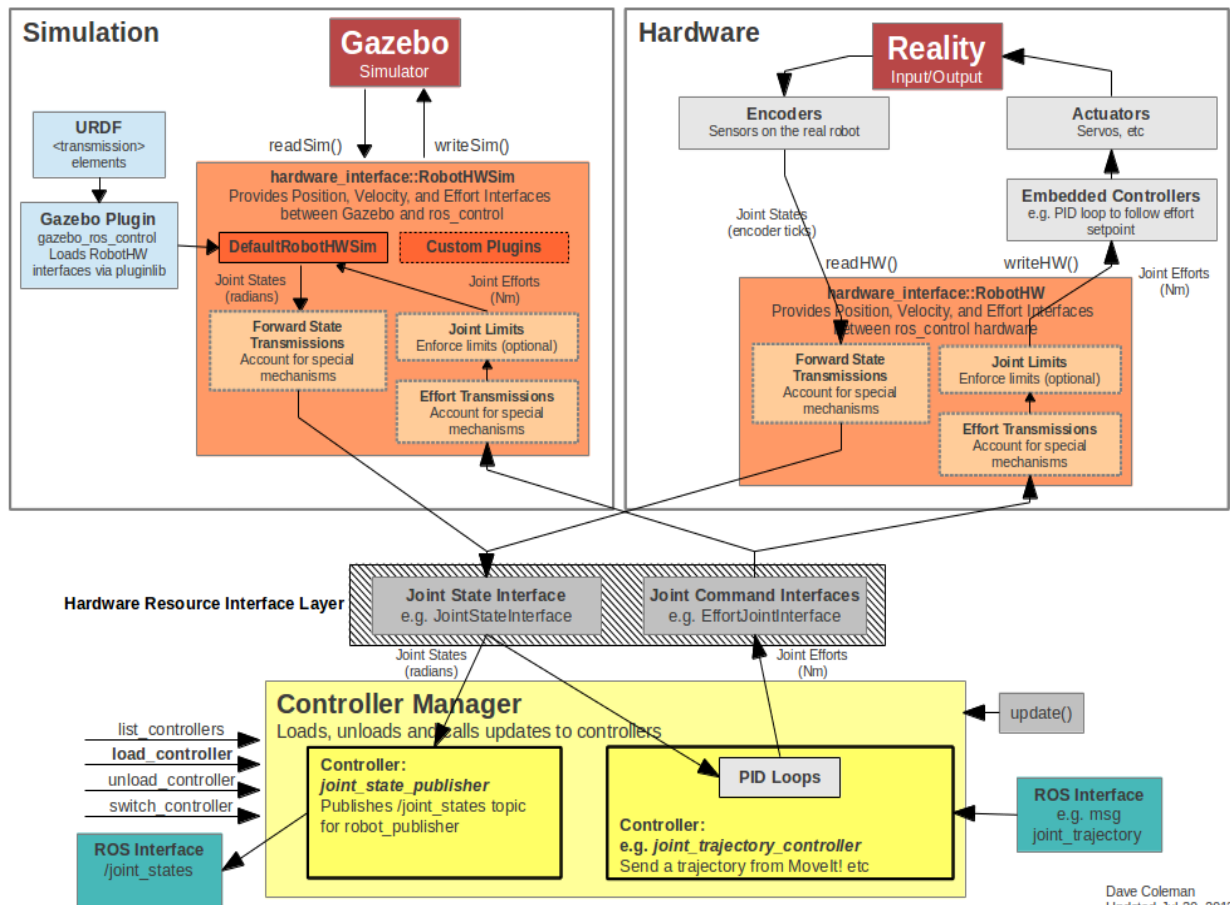
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/MYROBOT</robotNamespace>
    <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
  </plugin>
</gazebo>
```

6.d – Example using RRBot (Revolute-Revolute Manipulator Robot)

အစကတော့ ကျနော် ဒီ pdf ထဲ ထည့်မလို့ပါပဲ ၊ ကျနော်ပြောတာထက် အသုံးချပုံ ပြည့်ပြည့်စုံစုံကို ဒီ link မှာ သွားကြည့်လိုက်ရင် ပိုရှင်းသွားပါလိမ့်မယ် ။ အဲ့ link ကို နှိပ်လိုက်ပြီးရင် link ရဲ့ လက်ရှိရောက်နေတဲ့နေရာကနေ အောက်ပိုင်းအဆုံးထိဆို အသုံးချပုံ example ပါပါတယ် (http://gazebosim.org/tutorials/?tut=ros_control#RRBotExample)

အပေါ်မှာ ပြောခဲ့တဲ့အကြောင်းအရာအားလုံးကို နားလည်သွားတဲ့အခါ အောက်မှာပါတဲ့ပုံက သင့်ကို ပြီးပြည့်စုံစေမှာပါ

GAZEBO + ROS + ros_control



Dave Coleman
Updated Jul 30, 2013

Source – gazebosim.org

-----End-----