

### Première Partie: Comprendre l'URDF

- 1) Dans votre catkin workspace, créer un package `udm_urdf` avec pour dépendance `rospy`, `roscpp`, `urdf`, `std_msgs`, `geometry_msgs`, `sensors_msgs`.

```
The following information may help to resolve the situation:  
  
The following packages have unmet dependencies:  
  python-catkin-tools : Depends: python-catkin-pkg (>= 0.2.9) but it is not installable  
                        Depends: python-osrf-pycommon but it is not installable  
E: Unable to correct problems, you have held broken packages.  
deeya@deeya-OMEN-by-HP-Laptop-17-cb0xxx:~/udm_urdf$ catkin_create_pkg udm_urdf rospy roscpp urdf std_  
msg geometry_msgs sensors_msgs  
Created file udm_urdf/package.xml  
Created file udm_urdf/CMakeLists.txt  
Created folder udm_urdf/include/udm_urdf  
Created folder udm_urdf/src  
Successfully created files in /home/deeya/udm_urdf/udm_urdf. Please adjust the values in package.xml.  
deeya@deeya-OMEN-by-HP-Laptop-17-cb0xxx:~/udm_urdf$
```

- 2) Dans le package `udm_urdf` créer un dossier `urdf`.

```
deeya@deeya-OMEN-by-HP-Laptop-17-cb0xxx:~/udm_urdf$ mkdir urdf
```

- 3) Dans le dossier `urdf` créer les fichiers suivants `urdf_cylinder.urdf`, `box.urdf`, `sphere.urdf`, `mesh.urdf`


```
deeya@deeya-OMEN-by-HP-Laptop-17-cb0xxx:~/udm_urdf/urdf$ touch urdf_cylinder.urdf box.urdf sphere.urdf  
f mesh.urdf  
deeya@deeya-OMEN-by-HP-Laptop-17-cb0xxx:~/udm_urdf/urdf$ ls  
box.urdf mesh.urdf sphere.urdf urdf_cylinder.urdf
```

- 4) Modifier `cylinder.urdf` afin de faire afficher un cylindre de hauteur 10cm et de rayon 5cm.


```
*cylinder.urdf  
~/catkin_ws/src/udm_urdf/urdf  
Save  
Open  
<?xml version="1.0"?>  
<robot name="cylinder">  
  <link name="base_link">  
    <visual>  
      <geometry>  
        <cylinder length="0.1" radius="0.05"/>  
      </geometry>  
    </visual>  
  </link>  
</robot>
```

## Seekcha Sungkur – TP 2

- 5) Modifier box.urdf pour afficher un pave de 10x15x5cm

```
Open ▾  box.urdf  
~/catkin_ws/src/udm_urdf/urdf  
  
<?xml version="1.0"?>  
<robot name="cylinder">  
  <link name="base_link">  
    <visual>  
      <geometry>  
        <box size="0.1 0.15 0.05"/>  
      </geometry>  
    </visual>  
  </link>  
</robot>
```

- 6) Modifier sphere.urdf pour afficher une sphere de rayon 10cm

```
Open ▾  sphere.urdf  
~/catkin_ws/src/udm_urdf/urdf  
  
<?xml version="1.0"?>  
<robot name="cylinder">  
  <link name="base_link">  
    <visual>  
      <geometry>  
        <sphere radius="0.1"/>  
      </geometry>  
    </visual>  
  </link>  
</robot>
```

- 7) On va maintenant modifier mesh.urdf

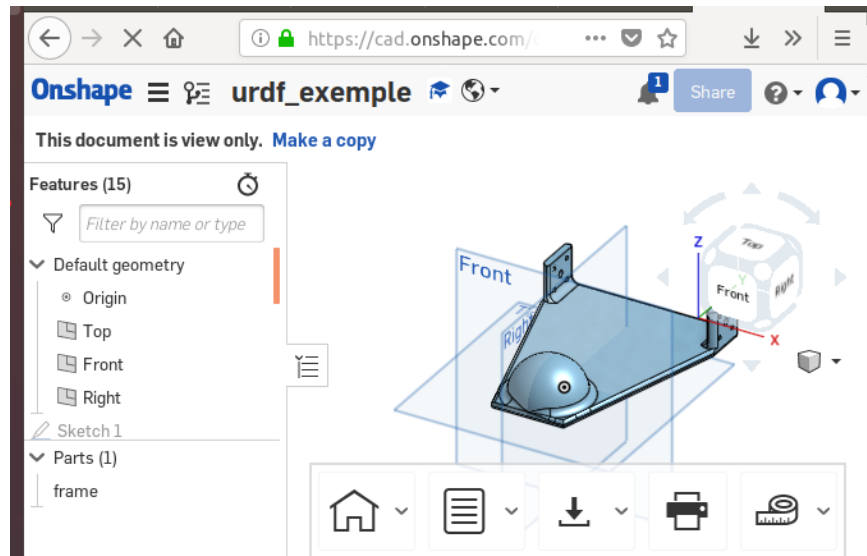
- a) Créer dans le package udm\_urdf un dossier mesh

```
deeya@deeya-VirtualBox:~/catkin_ws/src/udm_urdf$ mkdir mesh
```

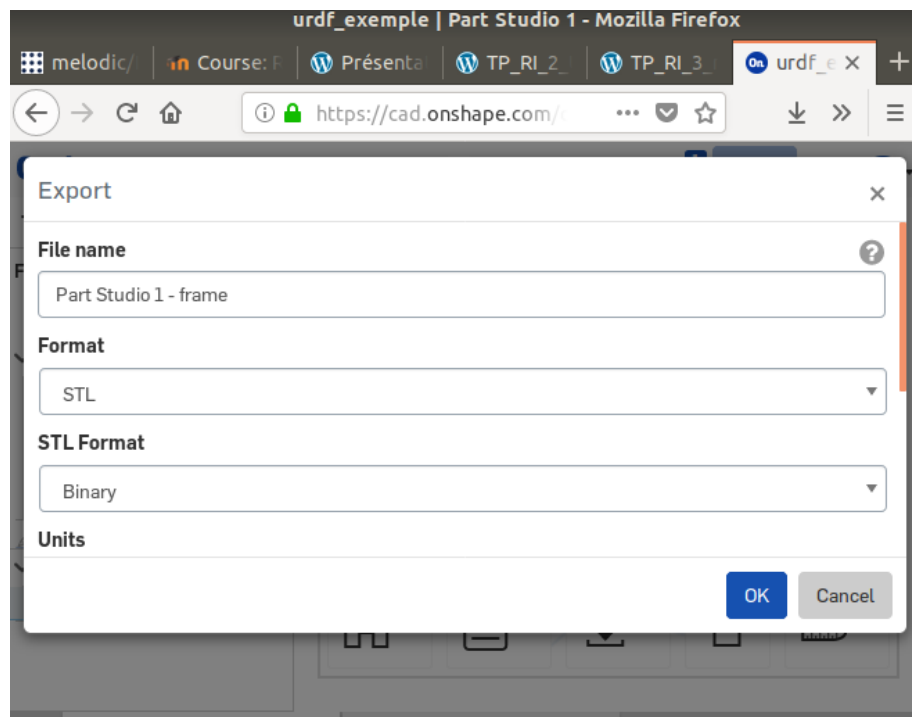
- b) Aller à l'adresse suivante:

<https://cad.onshape.com/documents/90c373b34c1f3bd23121eba5/w/84ce1e23bc20e257a735c19b/e/ab48b8ccbbbd88e1fb3d0565>

## Seekcha Sungkur – TP 2



c) Exporter le model 3D en stl (dans Parts clique gauche sur frame puis export)



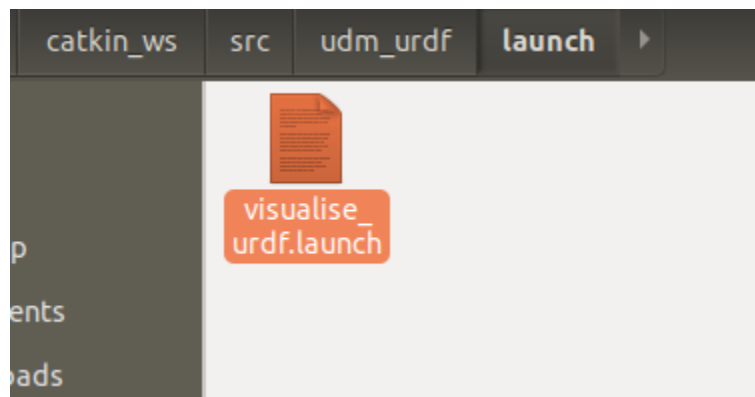
d) Puis modifier mesh.urdf afin d'afficher le model.

## Seekcha Sungkur – TP 2



```
<?xml version="1.0"?>
<robot name="mesh">
  <link name="base_link">
    <visual>
      <geometry>
        <mesh filename="package://udm_urdf/mesh/3d.stl" />
      </geometry>
    </visual>
  </link>
</robot>
```

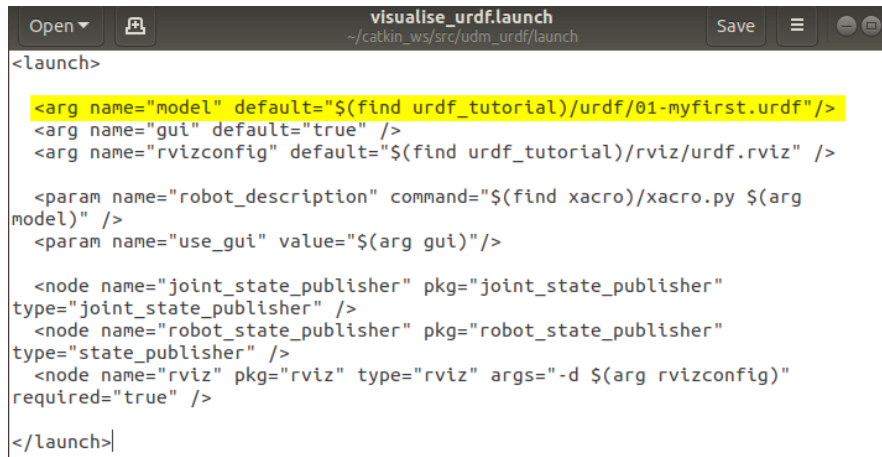
- e) Pour vérifier que les fichiers urdf sont correctes, utiliser l’outil `check_urdf`, si cela n’est pas installé, `sudo apt-get install liburdfdom-tools`.
- 8) Pour visualiser dans rviz, créer un dossier launch dans udm\_urdf
- a) Créer `visualize_urdf.launch`



- b) Ajouter un arg model qui aura pour valeur par défaut

```
”$(find udm_urdf)/urdf/cylinder.urdf”
```

## Seekcha Sungkur – TP 2



```
visualise_urdf.launch
~/catkin_ws/src/udm_urdf/launch

<launch>

  <arg name="model" default="$(find urdf_tutorial)/urdf/01-myfirst.urdf"/>
  <arg name="gui" default="true" />
  <arg name="rvizconfig" default="$(find urdf_tutorial)/rviz/urdf.rviz" />

  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />
  <param name="use_gui" value="$(arg gui)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rvizconfig)"
required="true" />

</launch>
```

- c) Ajouter un param robot\_description ajouter l'argument

command="\$(find xacro)/xacro.py \$(arg model)"



```
visualise_urdf.launch
~/catkin_ws/src/udm_urdf/launch

<launch>

  <arg name="model" default="$(find urdf_tutorial)/urdf/01-myfirst.urdf"/>
  <arg name="gui" default="true" />
  <arg name="rvizconfig" default="$(find urdf_tutorial)/rviz/urdf.rviz" />

  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />
  <param name="use_gui" value="$(arg gui)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rvizconfig)"
required="true" />

</launch>
```

- a) Ajouter le noeud joint\_state\_publisher\_gui qui se trouve dans le package du même nom

## Seekcha Sungkur – TP 2



```
<launch>

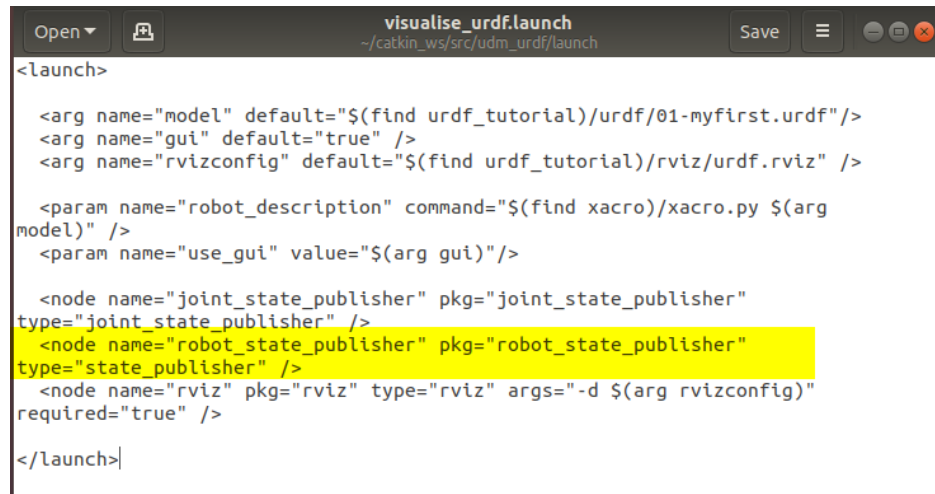
  <arg name="model" default="$(find urdf_tutorial)/urdf/01-myfirst.urdf"/>
  <arg name="gui" default="true" />
  <arg name="rvizconfig" default="$(find urdf_tutorial)/rviz/urdf.rviz" />

  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />
  <param name="use_gui" value="$(arg gui)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rvizconfig)"
required="true" />

</launch>
```

- b) Ajouter le noeud robot\_state\_publisher qui se trouve dans le package du même nom



```
<launch>

  <arg name="model" default="$(find urdf_tutorial)/urdf/01-myfirst.urdf"/>
  <arg name="gui" default="true" />
  <arg name="rvizconfig" default="$(find urdf_tutorial)/rviz/urdf.rviz" />

  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />
  <param name="use_gui" value="$(arg gui)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rvizconfig)"
required="true" />

</launch>
```

- c) Ajouter le noeud rviz qui se trouve dans le package du même nom avec args qui vaut “-d \$(find urdf\_tutorial)/rviz/urdf.rviz”

## Seekcha Sungkur – TP 2

```
visualise_urdf.launch
~/catkin_ws/src/udm_urdf/launch

<launch>

  <arg name="model" default="$(find urdf_tutorial)/urdf/01-myfirst.urdf"/>
  <arg name="gui" default="true" />
  <arg name="rvizconfig" default="$(find urdf_tutorial)/rviz/urdf.rviz" />

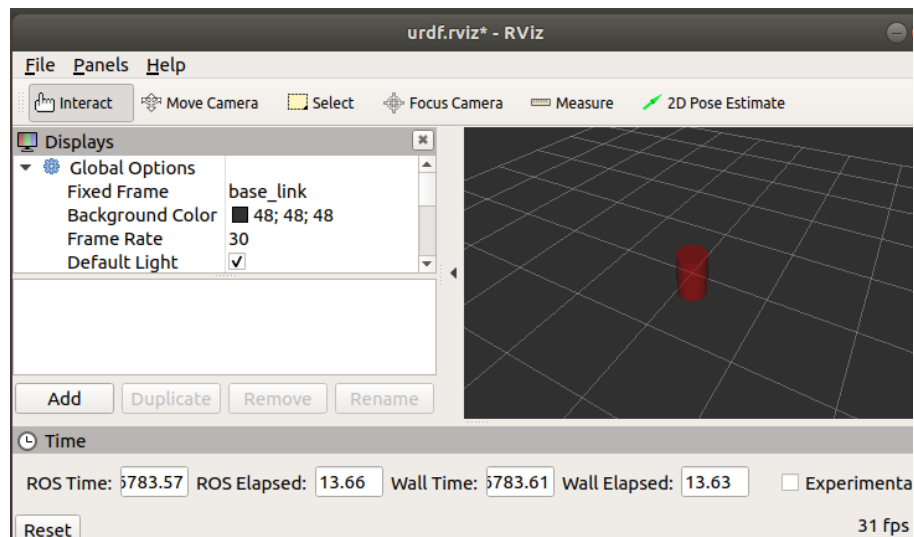
  <param name="robot_description" command="$(find xacro)/xacro.py $(arg
model)" />
  <param name="use_gui" value="$(arg gui)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rvizconfig)"
required="true" />

</launch>
```

d) Lancer le launch file

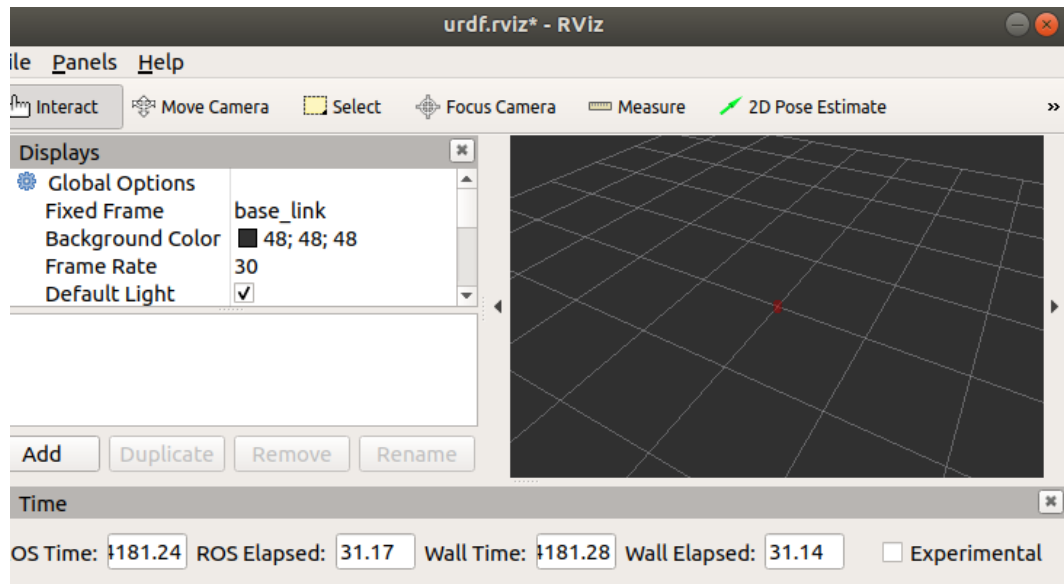
```
deeya@deeya-VirtualBox:~/catkin_ws$ source devel/setup.bash
deeya@deeya-VirtualBox:~/catkin_ws$ roslaunch udm_urdf visualise_urdf.launch
... logging to /home/deeya/.ros/log/0eb819ae-df9b-11ea-8b41-080027eb1150/roslau
nch-deeya-VirtualBox-9198.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
```



e) roslaunch udm\_urdf visualize\_urdf.launch model :=\$(find udm\_urdf)/urdf/cylinder.urdf

## Seekcha Sungkur – TP 2

```
deeya@deeya-VirtualBox:~/catkin_ws$ roslaunch udm_urdf visualise_urdf.launch mo
del:=$(find udm_urdf)/urdf/cylinder.urdf'
... logging to /home/deeya/.ros/log/87155ca4-dfc3-11ea-8b41-080027eb1150/roslau
nch-deeya-VirtualBox-9598.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.
```



### Partie Deux: Description de la main

- 9) Créer un fichier main.urdf

```
deeya@deeya-VirtualBox:~/catkin_ws/src/udm_urdf/urdf$ ls
box.urdf cylinder.urdf mesh.urdf sphere.urdf
deeya@deeya-VirtualBox:~/catkin_ws/src/udm_urdf/urdf$ touch main.urdf
deeya@deeya-VirtualBox:~/catkin_ws/src/udm_urdf/urdf$ ls
box.urdf cylinder.urdf main.urdf mesh.urdf sphere.urdf
```

- 10) Créer un box qui fait la taille de la paume d'une main

```
<link name="base_link">
  <visual>
    <geometry>
      <box size="0.10 0.04 0.11" />
    </geometry>
  </visual>
</link>
```

- 11) Ajouter une balise link avec pour nom index\_phalanx\_base



## Seekcha Sungkur – TP 2

```
<!-- Index -->
<link name="index_phalanx_base">
  <visual>
    <geometry>
      <cylinder length="0.04" radius="0.02" />
    </geometry>
    <material name="blue"/>
  </visual>
</link>
```

Ajouter une balise

joint avec le type fixe, la paume sera le parent et l'index\_dphalanx\_base sera le child.

```
<!-- Index -->
<link name="index_phalanx_base">
  <visual>
    <geometry>
      <cylinder length="0.04" radius="0.02" />
    </geometry>
    <material name="blue"/>
  </visual>
</link>
<link name="index_phalanx_middle">
  <visual>
    <geometry>
      <cylinder length="0.03" radius="0.009" />
    </geometry>
    <material name="red"/>
  </visual>
</link>
<link name="index_phalanx_end">
  <visual>
    <geometry>
      <cylinder length="0.03" radius="0.006" />
    </geometry>
    <material name="grey"/>
  </visual>
</link>
```

12) Ajouter une geometry cylinder qui fait la taille de la première phalange d'un index

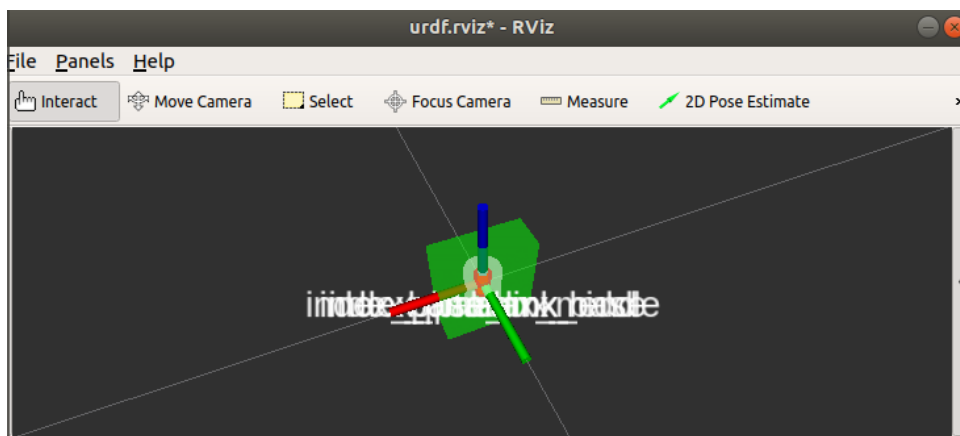
```
<joint name="base_link_to_index_base" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="base_link"/>
  <child link="index_phalanx_base"/>
  <limit effort="1000" upper="1.57" lower="-1.65" velocity="0.5"/>
  <origin xyz="0.03 0.0 0.05"/>
</joint>
<joint name="index_base_to_middle" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="index_phalanx_base"/>
  <child link="index_phalanx_middle"/>
  <limit effort="1000" upper="0" lower="-1.65" velocity="0.5"/>
  <origin xyz="0.0 0.0 0.03"/>
</joint>
<joint name="index_middle_to_end" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="index_phalanx_middle"/>
  <child link="index_phalanx_end"/>
  <limit effort="1000" upper="0" lower="-1.57" velocity="0.5"/>
  <origin xyz="0.0 0.0 0.025"/>
</joint>
```

13) Ajouter un matériel aux différentes pièces du système afin de le rendre plus lisible.

## Seekcha Sungkur – TP 2

```
Open  main.urdf  Save  ~/  
catkin_ws/src/udm_urdf/urdf  
  
<!-- Index -->  
<link name="index_phalanx_base">  
  <visual>  
    <geometry>  
      <cylinder length="0.04" radius="0.02" />  
    </geometry>  
    <material name="blue"/>  
  </visual>  
</link>  
  
<link name="index_phalanx_middle">  
  <visual>  
    <geometry>  
      <cylinder length="0.03" radius="0.009" />  
    </geometry>  
    <material name="red"/>  
  </visual>  
</link>  
  
<link name="index_phalanx_end">  
  <visual>  
    <geometry>  
      <cylinder length="0.03" radius="0.006" />  
    </geometry>  
    <material name="grey"/>  
  </visual>  
</link>
```

14) Visualiser votre urdf avec le launch file créé précédemment.



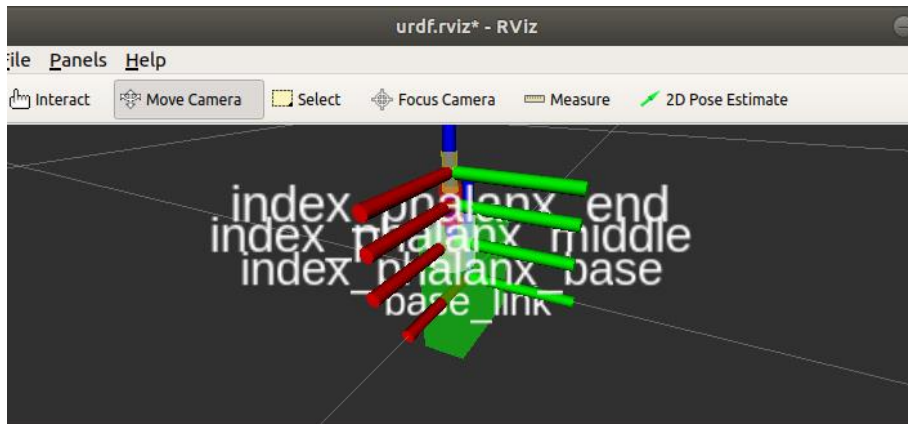
15) Dans le joint ajouter la balise origine qui aura deux arguments xyz et rpy le premier étant la position cartésienne de la phalange et le second l'orientation yaw pitch roll. Il faudra indiquer la position du joint

## Seekcha Sungkur – TP 2

```
<joint name="base_link_to_index_base" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="base_link"/>
  <child link="index_phalanx_base"/>
  <limit effort="1000" upper="1.57" lower="-1.65" velocity="0.5"/>
  <origin xyz="0.03 0.0 0.05"/>
</joint>
<joint name="index_base_to_middle" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="index_phalanx_base"/>
  <child link="index_phalanx_middle"/>
  <limit effort="1000" upper="0" lower="-1.65" velocity="0.5"/>
  <origin xyz="0.0 0.0 0.03"/>
</joint>
<joint name="index_middle_to_end" type="revolute">
  <axis xyz="1 0 0"/>
  <parent link="index_phalanx_middle"/>
  <child link="index_phalanx_end"/>
  <limit effort="1000" upper="0" lower="-1.57" velocity="0.5"/>
  <origin xyz="0.0 0.0 0.025"/>
</joint>
```

16) Dans le joint ajouter la position du joint relatif au parent.

17) Ajouter le reste des phalanges pour compléter l'index.



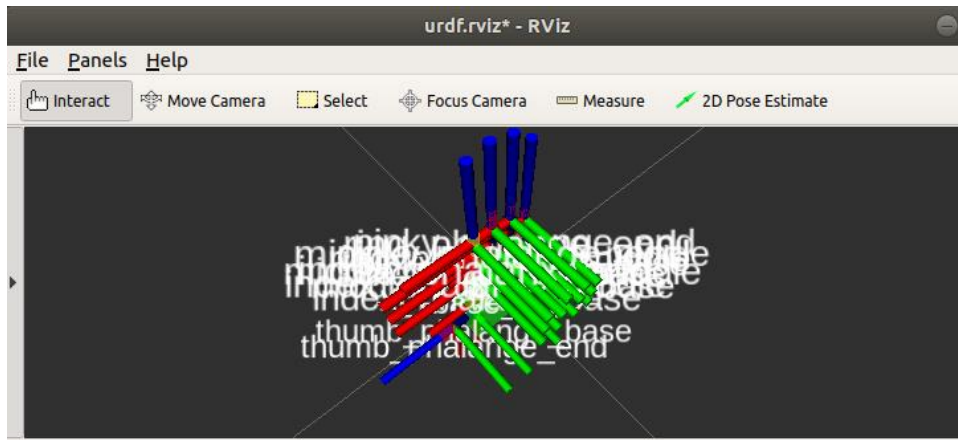
### Partie 3: Articuler le modèle

18) Modifier les joints du fichier main.urdf afin de mettre des revolute joint.

19) Lancer le launch file et tester les joints

20) Terminer la main en ajoutant les doigts manquants.

## Seekcha Sungkur – TP 2



21) Déposer sous Moodle un lien vers le projet GitHub qui contient le package udm\_urdf.