Motion Generator & GAZEBO & Action Engine

LEE GEON HEE
gunhee6392@gmail.com

Manual List:

1) ROS Package
2) Robocare data

1. ROS Package.

Package name:

1. kist_ws(in NAS)
2. action_engine(in git hub)

Packages lists(kist_ws):

Silbor3_2dnav

Silbot3_description

Silbot3_feasible_moveit

Silbot3_gazebo

Silbot3_motion_generation

Silbot3_msgs

Silbot3_omniwheels(incomplete)

Silbot3_slam

Silbot3_silbot3_teleoperation

Silbot3_tutorial
Silbot3_xmlparsing

### highlight is important function and used frequently.

Description(kist_ws):

1. Silbor3_2dnav

This is a package for implementing Navigation function.
It is basically used to using move_base function, therefore launch files and yaml files is described according to default format.

2. Silbot3_description

It is a package for silbot3 to have its modeling data which include stl file with mesh and URDF file(XACRO) describing relationship between joint and link.
It also includes launch file to run in the GAZEBO which means 3D virtual environment.

3. Silbot3_feasible_moveit

It is used to utilizing MoveIt API, but now it isn't used.

4. Silbot3_gazebo

It is a package to deploying various functions in the GAZEBO.

For using ros_control, It is added to yaml file regarding controller(ex. Position, trajectory controller) and launch file to run.

5. Silbot3_motion_generation

It is used to making motion generation program based on MoveIt!, but it is considerably transformed for implementing motion generation program and linked silbot3_xmlparsing package.

6. Silbot3_msgs

It is a package provided by the Robocare.
It included msg and service file for working Silbot3.

7. Silbot3_omniwheels(incomplete)

It is a package for other researcher to implement holonomic constraint which means omni-wheels mobility here and added for working navigation functions in the GAZEBO.

8. Silbot3_slam

It is a package for working mapping and map server.

9. Silbot3_teleoperation

It is a package for looking movement of Silbot3 in the GAZEBO and means "Teleoperation" function using Keyboard.

10. Silbot3_tutorial

It is a package for working silbot3 in the Robocare.
It is written by Python and C(++) according to various function, for example LED and movement… , It is just added to take possibility for working Silbot3 in real.

11. Silbot3_xmlparsing

It is a package which has Tinyxml library which helps to make xml files and makes xml files for working Silbot3.
Using Xml files, you can work Silbot3 in real and GAZEBO.

How to install the dependencies and libraries (it will be added later).

For launching the GAZEBO,

```
$ sudo apt-get install ros-kinetic-joint-state-controller*
```

For launching the MoveIt!

```
$ sudo apt-get install ros-kinetic-interative-maker
$ sudo apt-get install ros-kinetic-reconfigure
$ sudo apt-get install ros-kinetic-dynamic-reconfigure
```

And then you should copy from your_workspace/devel/silbot3_xmlparsing package to /opt/ros/kinetic/devel since dependency problem is not solved

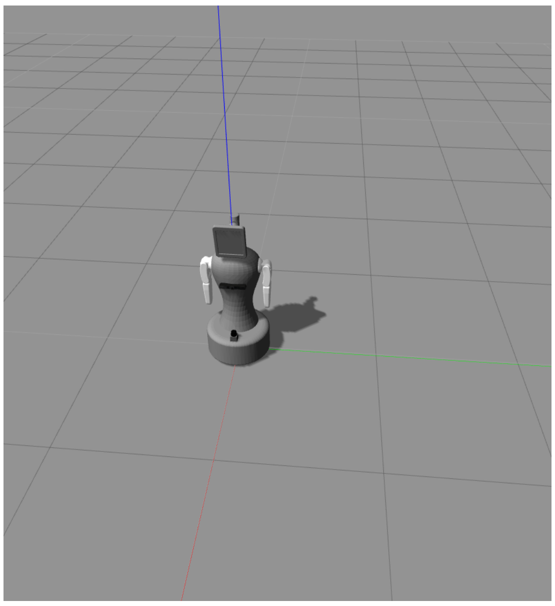
```
$ sudo cp –rp your_workspace/devel/silbot3_xmlparsing /opt/ros/kinetic/devel
```

How to launch each function.

1. GAZEBO.

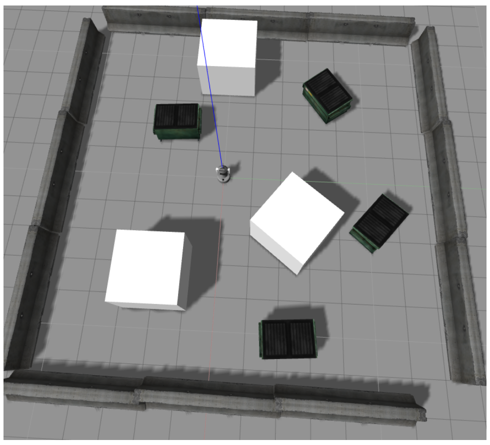
Load the model of Silbot3 in the GAZEBO.

```
$ roslauch silbot3_description only_silbot.launch
```



Load the model of Silbot3 with obstacles in the GAZEBO.

```
$ roslaunch silbot3_description silbot3_collision_xacro.launch
```



Load the model of Silbot3 with obstacle in the GAZEBO and controller.

Once you load the model,

```
$ roslaunch silbot3_description silbot3_collision_xacro.launch
```

if you want to implement position controller, you can write

```
$ roslaunch silbot3_gazebo silbot_arm_position_controller.launch
```

### if you want to check xml file, you should load the position controller.

if you want to implement position controller, you can write

```
$ roslaunch silbot3_gazebo silbot_arm_trajectory_controller.launch
```

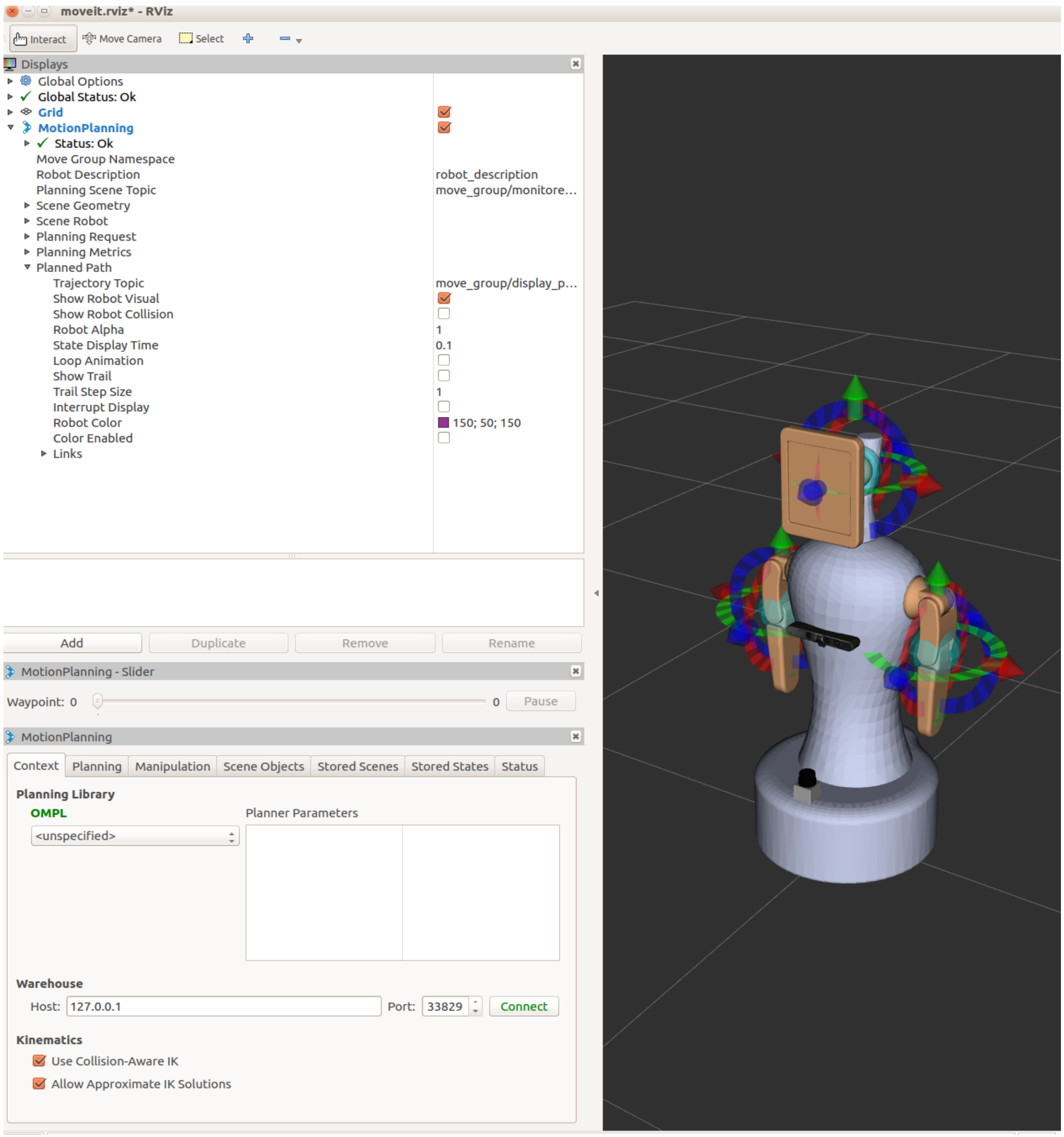
2. Motion Generation program.

It can make xml file in the Rviz using Silbot3.

First, you can see this GUI when you command as:

```
roslaunch silbot3_motion_generation motionGenerationProgram.launch
```

### Please check “Allow Approximate IK Solution” checkbox in left-bottom since it can give to control arm of silbot3 easily.



Global Options	
Global Status: Ok	
Grid	✓
MotionPlanning	✓
Status: Ok	
Move Group Namespace	
Robot Description	robot_description
Planning Scene Topic	move_group/monitore…
Scene Geometry	
Scene Robot	
Planning Request	
Planning Metrics	
Planned Path	
Trajectory Topic	move_group/display_p…
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	1
State Display Time	0.1
Loop Animation	☐
Show Trail	☐
Trail Step Size	1
Interrupt Display	☐
Robot Color	150; 50; 150
Color Enabled	☐
Links	

Waypoint: 0 0

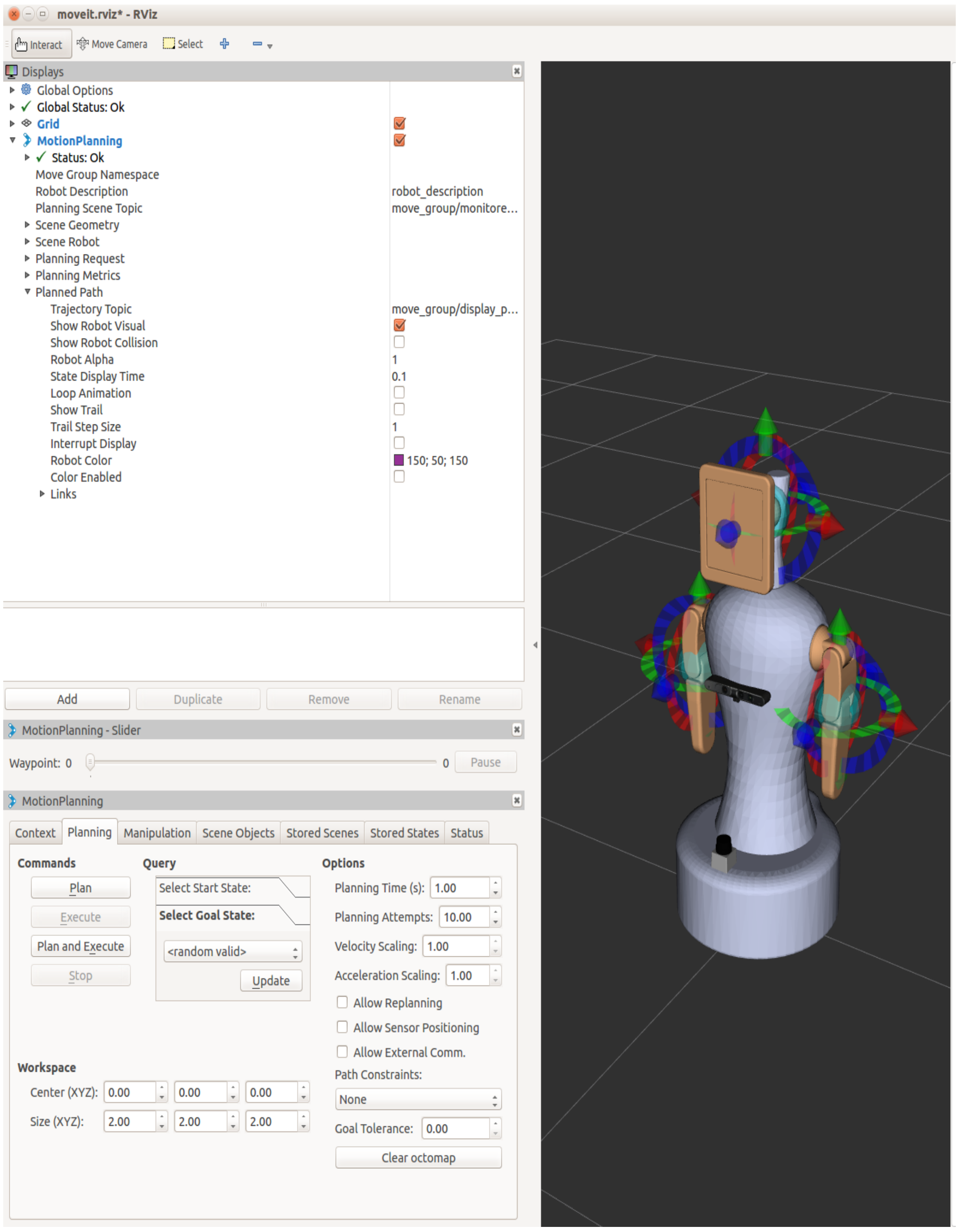
Planning Library
OMPL
<unspecified>
Planner Parameters

Warehouse
Host: 127.0.0.1 Port: 33829

Kinematics
✓ Use Collision-Aware IK
✓ Allow Approximate IK Solutions

Second, You can choice Planning tab in left-bottom for making xml file
In this case, you can control silbot3 using interactive markers

When you click the “Plan” button, you can see trajectory to which arm and head of silbot3 move according.
### Its process isn’t affect making a xml file.
When you click the “Execute” or “Plan and Execute” button, making a xml file immediately works according to the trajectory of “Plan” and depends on previous operation.



Global Options	
Global Status: Ok	
Grid	✓
MotionPlanning	✓
Status: Ok	
Move Group Namespace	
Robot Description	robot_description
Planning Scene Topic	move_group/monitore...
Scene Geometry	
Scene Robot	
Planning Request	
Planning Metrics	
Planned Path	
Trajectory Topic	move_group/display_p...
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	1
State Display Time	0.1
Loop Animation	☐
Show Trail	☐
Trail Step Size	1
Interrupt Display	☐
Robot Color	🟪 150; 50; 150
Color Enabled	☐
Links	

Waypoint: 0 — 0

Commands
Plan
Execute
Plan and Execute
Stop

Query
Select Start State:
Select Goal State:
<random valid>
Update

Options
Planning Time (s): 1.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
Path Constraints: None
Goal Tolerance: 0.00
Clear octomap

Workspace
Center (XYZ): 0.00 0.00 0.00
Size (XYZ): 2.00 2.00 2.00

Finally, you can modulate many parameters and determine file name and directory.

INITIALIZATION_FROM_YAML param confirms whether it is initialized using yaml file.

FILE_NAME param is file name which you want to take.

FILE_DIRECTORY is saved directory of file which you want to take.

rate param is the rate of /joint_state topic.

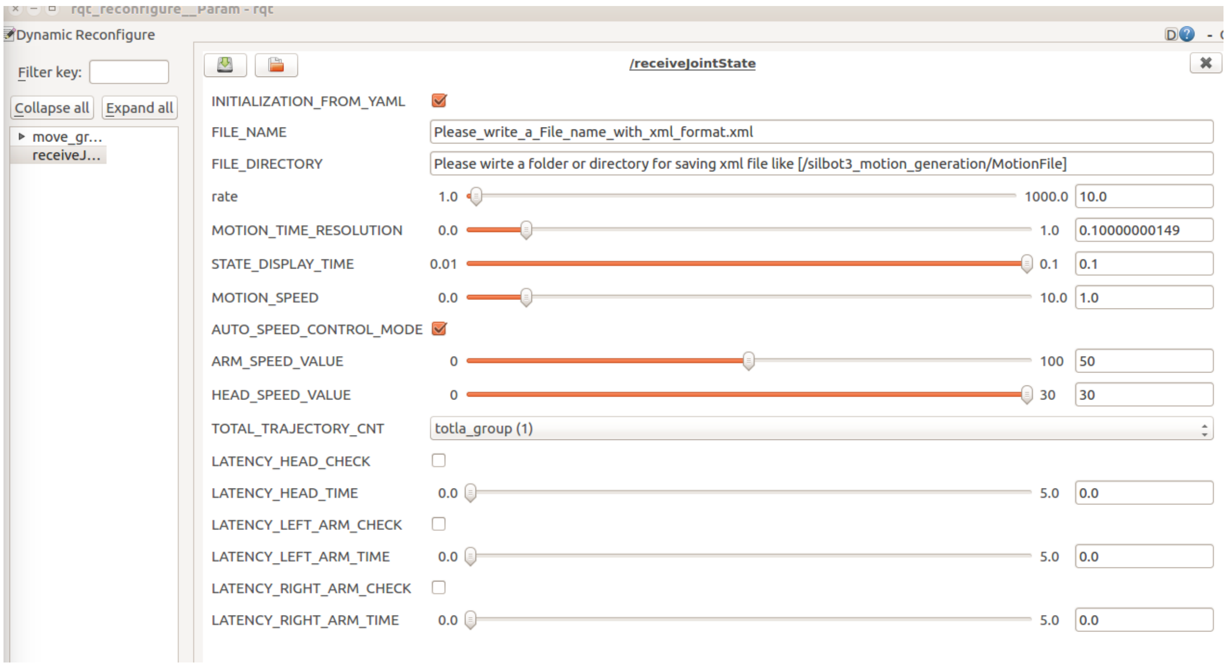
MOTION_TIME_RESOLUTION param is value of time gap of xml file.

AUTO_SPEED_CONTROL_MODE param is autonomous mode, if you select it, it autonomously decide param of speed value.

ARM_SPEED_VALUE, HEAD_SPEED_VALUE param is value to be described with speed value in xml file.

TOTAL_TRAJECTORY_CNT param is count how to work frequently, but it don't need to control.

LATENCY params provide latency function, it can be selected with group and should have amount of latency time in order to work



/receiveJointState

INITIALIZATION_FROM_YAML	✓		
FILE_NAME	Please_write_a_File_name_with_xml_format.xml		
FILE_DIRECTORY	Please wirte a folder or directory for saving xml file like [/silbot3_motion_generation/MotionFile]		
rate	1.0	1000.0	10.0
MOTION_TIME_RESOLUTION	0.0	1.0	0.10000000149
STATE_DISPLAY_TIME	0.01	0.1	0.1
MOTION_SPEED	0.0	10.0	1.0
AUTO_SPEED_CONTROL_MODE	✓		
ARM_SPEED_VALUE	0	100	50
HEAD_SPEED_VALUE	0	30	30
TOTAL_TRAJECTORY_CNT	totla_group (1)		
LATENCY_HEAD_CHECK	☐		
LATENCY_HEAD_TIME	0.0	5.0	0.0
LATENCY_LEFT_ARM_CHECK	☐		
LATENCY_LEFT_ARM_TIME	0.0	5.0	0.0
LATENCY_RIGHT_ARM_CHECK	☐		
LATENCY_RIGHT_ARM_TIME	0.0	5.0	0.0

3. Playing the created motion in GAZEBO.

In order to play motion of slilbot3 with xml file in GAZEBO, you will mostly use the silbot3_xmlparsing package.
Once you finish previous task such as loading model in GAZEBO and working position controller, you can check playing motion of Silbot3.

You should parse the xml file.
${prefix}: the directory of xml file you want
($(arg file name)): the file name of xml file you want

```
rosrun silbot3_xmlparsing silbot3_xmlparsing ${prefix}/($(arg file name))
```

You can see motion of Silbot3.

```
rosrun silbot3_xmlparsing trXml2Cmd
```

4. Classifying Xml file.

It is consisted of silbot3_motion_classification package, which makes modified xml files according to configuration file in the package.
Its package generally separates xml file containing data of head, left arm, right arm from xml file containing whole body motion using the configuration file

# Information of default name and directory of xml file is located at

- silbot3_motion_classification/launch/silbot3_motion_classification
- silbot3_motion_classification/src/createClassifiedXML.cpp
- silbot3_motion_generation/src/RCVJointState

After you work the motion generation program, you can get a xml file
(This is default setting)

```
roslaunch silbot3_motion_generation motionGenerationProgram.launch
```

#when you turn off Rviz tool, the xml file is instantly created.

default.xml

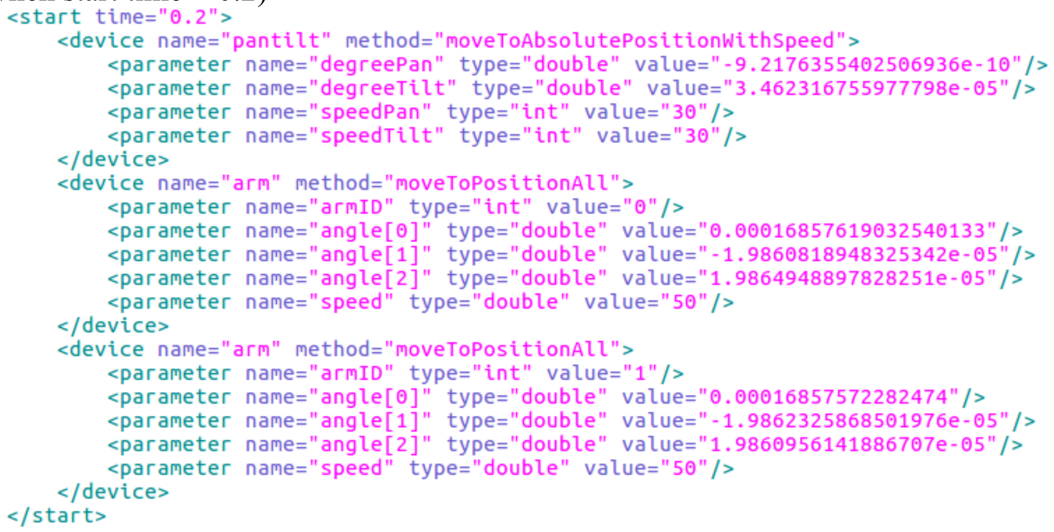
and then you can classify the xml file with the configuration file
(This is default setting)

```
roslaunch silbot3_motion_classification silbot3_motion_classification.launch
```

default1.xml default2.xml default3.xml

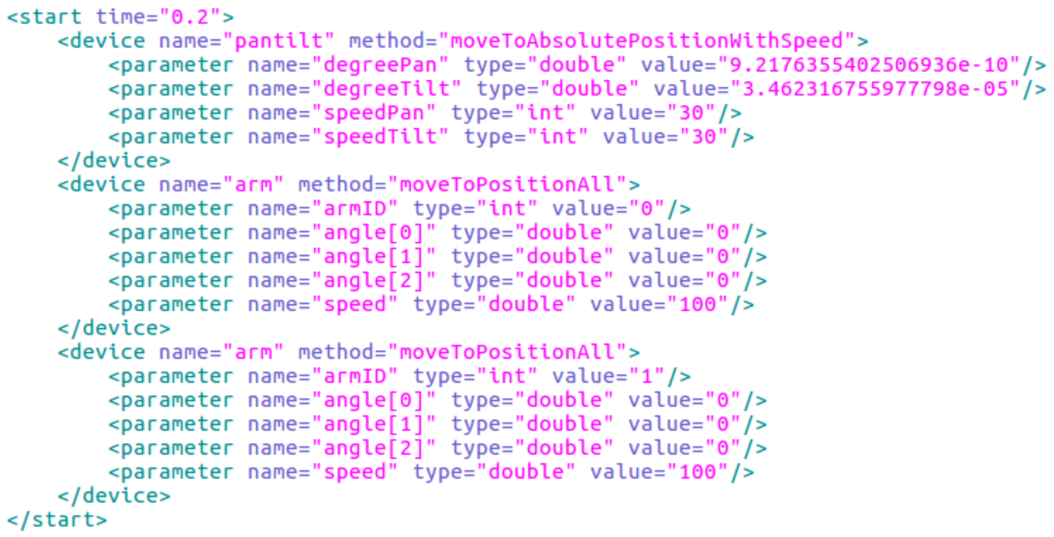
We can check how to separate xml file.

This is part of the original file
(when start time = 0.2)



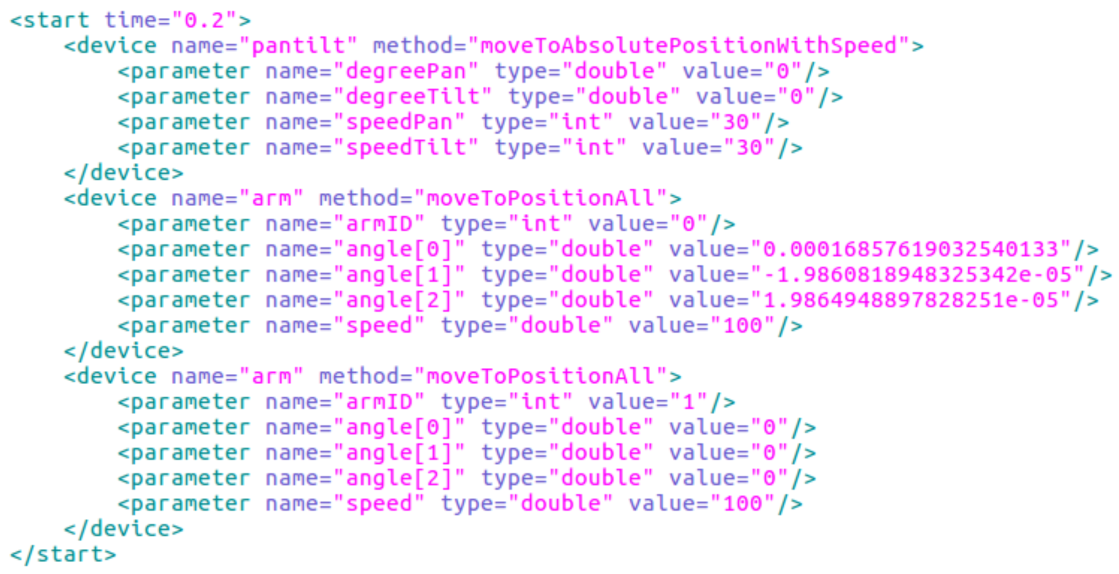
```xml
<start time="0.2">
    <device name="pantilt" method="moveToAbsolutePositionWithSpeed">
        <parameter name="degreePan" type="double" value="-9.2176355402506936e-10"/>
        <parameter name="degreeTilt" type="double" value="3.462316755977798e-05"/>
        <parameter name="speedPan" type="int" value="30"/>
        <parameter name="speedTilt" type="int" value="30"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="0"/>
        <parameter name="angle[0]" type="double" value="0.00016857619032540133"/>
        <parameter name="angle[1]" type="double" value="-1.9860818948325342e-05"/>
        <parameter name="angle[2]" type="double" value="1.9864948897828251e-05"/>
        <parameter name="speed" type="double" value="50"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="1"/>
        <parameter name="angle[0]" type="double" value="0.00016857572282474"/>
        <parameter name="angle[1]" type="double" value="-1.9862325868501976e-05"/>
        <parameter name="angle[2]" type="double" value="1.9860956141886707e-05"/>
        <parameter name="speed" type="double" value="50"/>
    </device>
</start>
```

This is a part of the classificated file associated with head
(default1.xml)



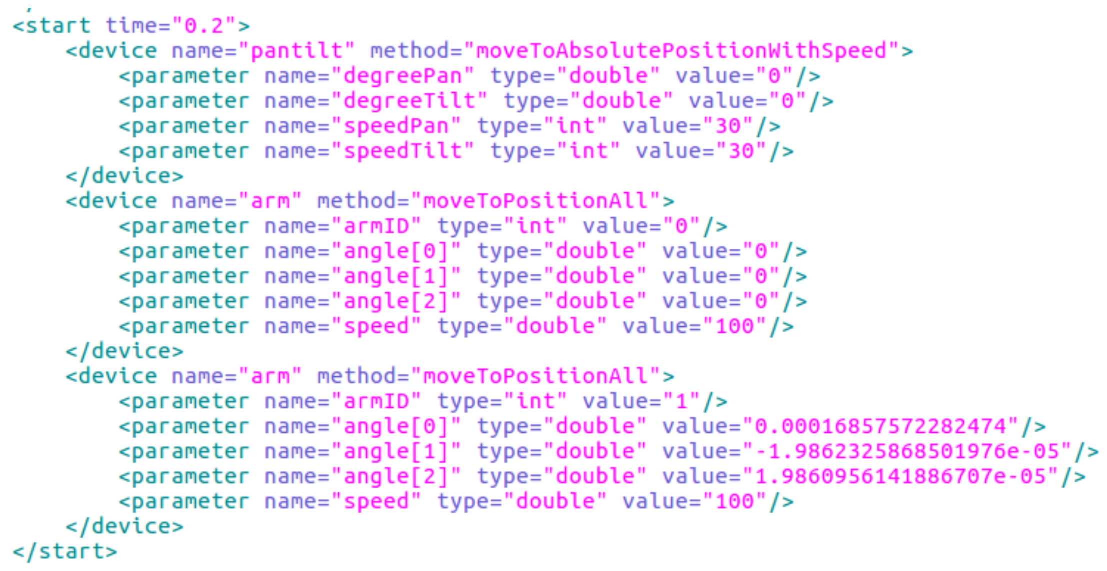
```xml
<start time="0.2">
    <device name="pantilt" method="moveToAbsolutePositionWithSpeed">
        <parameter name="degreePan" type="double" value="9.2176355402506936e-10"/>
        <parameter name="degreeTilt" type="double" value="3.462316755977798e-05"/>
        <parameter name="speedPan" type="int" value="30"/>
        <parameter name="speedTilt" type="int" value="30"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="0"/>
        <parameter name="angle[0]" type="double" value="0"/>
        <parameter name="angle[1]" type="double" value="0"/>
        <parameter name="angle[2]" type="double" value="0"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="1"/>
        <parameter name="angle[0]" type="double" value="0"/>
        <parameter name="angle[1]" type="double" value="0"/>
        <parameter name="angle[2]" type="double" value="0"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
</start>
```

This is a part of the classificated file associated with left arm (default2.xml)



```xml
<start time="0.2">
    <device name="pantilt" method="moveToAbsolutePositionWithSpeed">
        <parameter name="degreePan" type="double" value="0"/>
        <parameter name="degreeTilt" type="double" value="0"/>
        <parameter name="speedPan" type="int" value="30"/>
        <parameter name="speedTilt" type="int" value="30"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="0"/>
        <parameter name="angle[0]" type="double" value="0.00016857619032540133"/>
        <parameter name="angle[1]" type="double" value="-1.9860818948325342e-05"/>
        <parameter name="angle[2]" type="double" value="1.9864948897828251e-05"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="1"/>
        <parameter name="angle[0]" type="double" value="0"/>
        <parameter name="angle[1]" type="double" value="0"/>
        <parameter name="angle[2]" type="double" value="0"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
</start>
```

This is a part of the classificated file associated with right arm (default3.xml)



```xml
<start time="0.2">
    <device name="pantilt" method="moveToAbsolutePositionWithSpeed">
        <parameter name="degreePan" type="double" value="0"/>
        <parameter name="degreeTilt" type="double" value="0"/>
        <parameter name="speedPan" type="int" value="30"/>
        <parameter name="speedTilt" type="int" value="30"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="0"/>
        <parameter name="angle[0]" type="double" value="0"/>
        <parameter name="angle[1]" type="double" value="0"/>
        <parameter name="angle[2]" type="double" value="0"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
    <device name="arm" method="moveToPositionAll">
        <parameter name="armID" type="int" value="1"/>
        <parameter name="angle[0]" type="double" value="0.00016857572282474"/>
        <parameter name="angle[1]" type="double" value="-1.9862325868501976e-05"/>
        <parameter name="angle[2]" type="double" value="1.9860956141886707e-05"/>
        <parameter name="speed" type="double" value="100"/>
    </device>
</start>
```

you can identify previous task.

you should execute launch files regarding GAZEBO and XML parsing package in order to check whether it is made perfectly.

The First, you start to launch GAZEBO and load model of silbot3

```
roslaunch silbot3_description only_silbot.launch
```

The Second, you start to launch Controller
(In this case, you should use position controller)

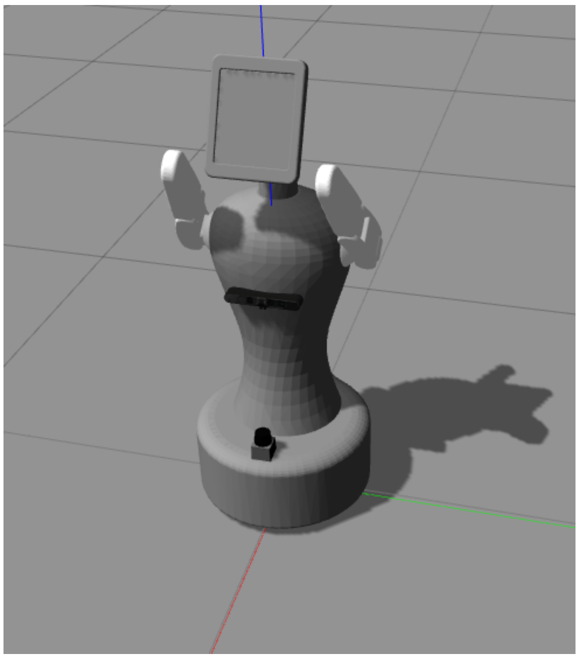
```
roslaunch silbot3_gazebo silbot_arm_position_controller.launch
```

The Third, you must parse xml file for change xml data into ROS topic
(In this case, Its command processes original Xml file)

```
rosrun silbot3_xmlparsing xmlparsing /home/kist/MotnFile/default.xml
```

And then you should transform xml topic data to the commands which can control silbot3 in GAZEBO

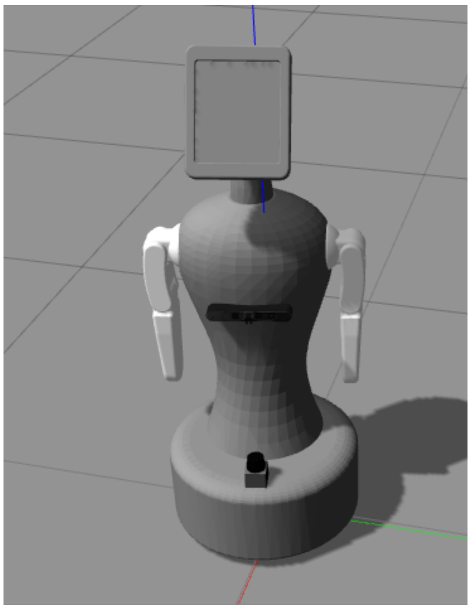
```
rosrun silbot3_xmlparsing trXml2Cmd
```



(In this case, its command processes the Xml file classificated with head)

```
rosrun silbot3_xmlparsing xmlparsing /home/kist/MotnFile/default1.xml
```

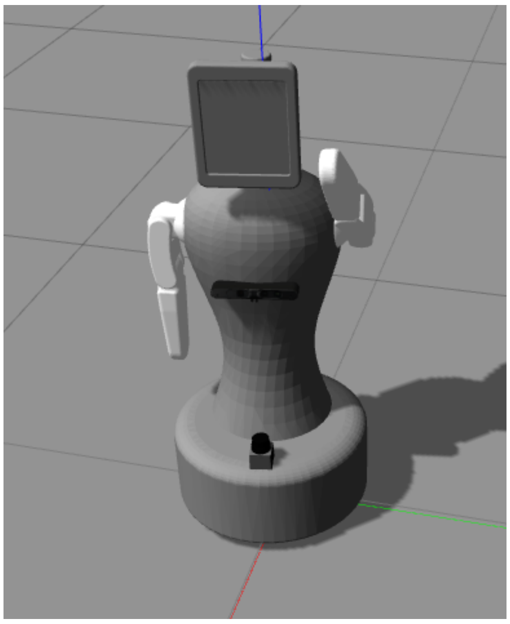
```
rosrun silbot3_xmlparsing trXml2Cmd
```



(In this case, its command processes the Xml file classificated with left arm)

```
rosrun silbot3_xmlparsing xmlparsing /home/kist/MotnFile/default2.xml
```

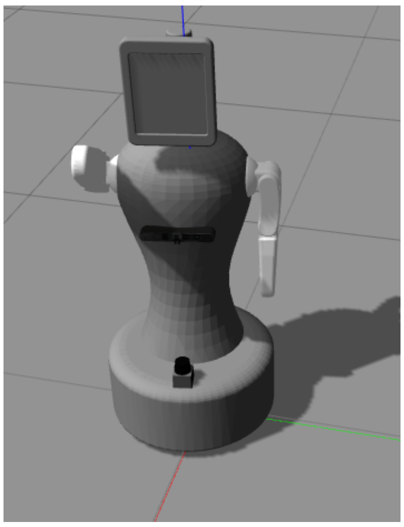
```
rosrun silbot3_xmlparsing trXml2Cmd
```



(In this case, its command processes the Xml file classificated with right arm)

```
rosrun silbot3_xmlparsing xmlparsing /home/kist/MotnFile/default4.xml
```

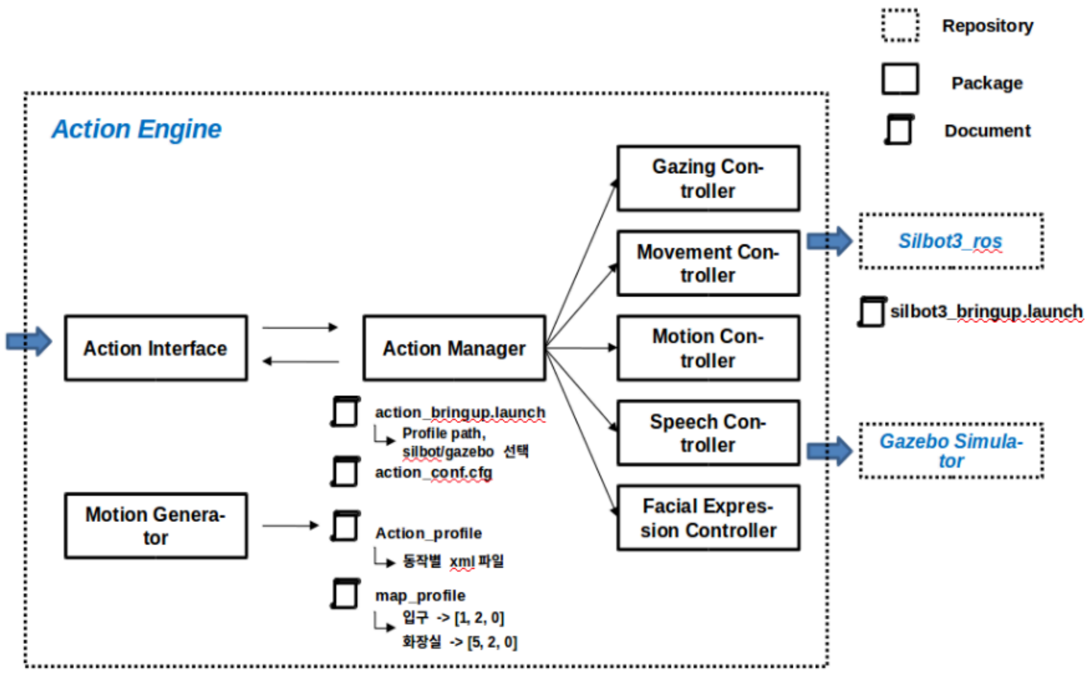
```
rosrun silbot3_xmlparsing trXml2Cmd
```



5. Action Engine.

The objective of action engine framework is to implement the natural action like human. Initial source is loaded in the https://github.com/Geonhee-LEE/action_engine.git.
It is just the temporal and initial structure which means not perfect and complete.
What if you want to change the different structure whether you are a successor or not, you can do.

Let me introduce the structure of action engine. The first thing is whole structure of action engine containing each manager and packages and so on as follow.



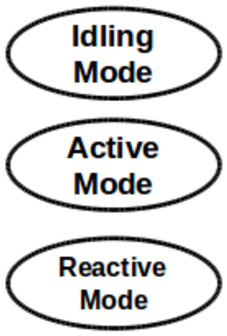
The action interface package implements the buffer which convert the input of framework or package to the topic we want. In brief, it connects outside function and action engine.
I didn't make the action interface package and I used just rostopic pub command to check.

The action manager package which have two modules, there are action manager class and manager interface class, is the most important thing among these package since the action manager should manage controllers. The manager interface class of action manager have a function which load many variables to the Parameter server which it is provided by ROS As I said, it is currently not completed to decide the many variables associated with gazing and movement and motion and speech and facial.
The many variables are separated by each controller as follow:

```
movement : { forward : 0, backward: 0, right: 0, left: 0, rotation : 0, map_x : 0, map_y : 0 }
motion : { intention  : "greeting", deictic : 0, iconic : 0, metaphoric : 0, beat : 0 , polite : 0}
gazing :  { head : false, torso : false}
speech : { contents : "hello" }
face_expression : { emotion : "happy", arouse : 0, valence : 0, stance : 0}
```

Each controller has to work using the variables according to its element, for example the movement controller should move the forward by 1m when movement parameter is {forward: 1} or move to the x=1 of map coordinate when movement parameter is {map_x:1} using the movement controller. You have to add suitable parameters for using the controller.

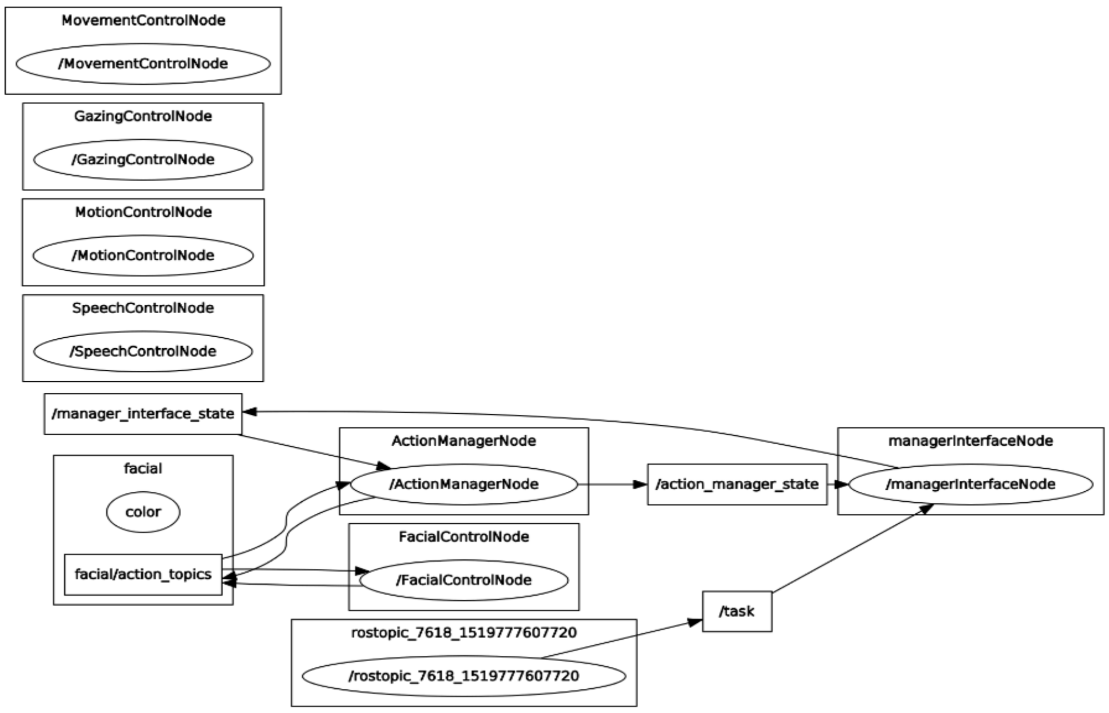
Another important concept is action mode which is consist of idling mode and active mode and reactive mode. In idling mode case, the robot should naturally act such as looking around. In active mode case, the robot should conduct the mission according to interaction or situation such as greeting, pointing, grasp. In reactive mode case, the robot should listen to object or human for receiving input such as gazing.



If you receive input, as a result, the manager interface class of action manager package will select action mode and decide parameters in order to load to the Parameter server of ROS. But now it is implemented as a dummy form.

The second thing is the action manager class of the action manager package. It implements between action manager package and controllers using actionlib. I thought that controller is affected by the kind of action mode, therefore I decided controllers which will work according to action mode. For example, in reactive mode case, it needs to work gazing controller and facial controller. In idling mode, it needs to work motion controller and facial controller. In active mode, it needs to work facial and speech and motion and movement controller.

Finally, I implemented very simple task for checking the action engine as follow:



The action manager class constantly interact with the manager interface class with own condition and the manager interface class receive /task input which contains "active mode" as dummy. The action manager class connects to the facial controller with actionlib. But, I wanted to implement connection between manager interface and several controllers together. So, I recommend you to fine multi-client and multi-server concepts. As I result, I can check different greeting motion using silbot3 according to different polite parameters.

+ In my opinion, does we need gazing controller in the action engine?
+ it needs to change many functions, which are placed in

manager_interface.cpp:

```
int8_t manager_interface::selectActionMode(silbot3_action_manager::inputTask taskMsg)
void anager_interface::transInput2Struct(silbot3_action_manager::inputTask, Action_struct *)
```

actionManager.cpp

```
void actionManager::realizeMode()
void actionManager::treatAction(int8_t current_state , int8_t desiring_mode)
void actionManager::manageController( int8_t working_mode )
```

+ every controller

How to launch each function.

The action manager function is launched as follow:

```
roslaunch silbot3_action_manager silbot3_action_manager.launch
roslaunch silbot3_action_manager silbot3_manager_interface.launch
```

The controller's functions is launched as follow:

```
roslaunch silbot3_action_manager silbot3_action_controller.launch
```

4. Robocare data

saved in the NAS of Cjsteam

(smb://161.122.114.157/cjsteam/Users/이건희/2018 New Silbot File)