SMART FARMING DUAL-ROBOTIC ARM

Muhammad Noor Sabri Bin Md Yusoff

School of Electrical and Electronic Engineering
Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia

1. INTRODUCTION

1.1 OBJECTIVES

1. To solve the problem faced by the agriculture sector
2. To increase the number of crops yielded
3. To provide a cheaper system for agriculture sector.

1.2 PROBLEM DEFINITION

Conventional farming has proved to occupy large amount of **space** to compensate for the passages so that the farmers to work on. This affects the overall annual yield of the crops. To overcome this issue, the unmanned dual-robotic arms will be deployed to exploit the maximum potential in the field.

1.3 PROBLEM SOLUTION

To make a dual-robotic arm system which can be very helpful in the future. Not only that this system saves time and **space**, it is also cheaper when comparing with the amount that every employer had to pay their farmers. This system is also very effective as it is non-human and need no rest.

1.4 SOUND METHODOLOGY

The current agriculture technique will be observed and revised to make it more efficient. A proper analysis by using kinematic analysis of the dual-robotic arm system will be done. This system will utilize ROS (kinetic) control as a software backbone to run itself. Simulation of this system will be done in ROS gazebo.

1.5 TIMELINE

	APRIL		MAY				JUNE			
	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Problem Statement/Solution	🟥									
Analyse the system	🟨	🟨								
Install ROS (kinetic)	🟩	🟩	🟩	🟩						
Explore ROS (kinetic)	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Simulation	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Demonstration										⬛

Table 1 Timeline of the work

2. METHODOLOGY

2.1 THE DUAL-ROBOTIC ARM

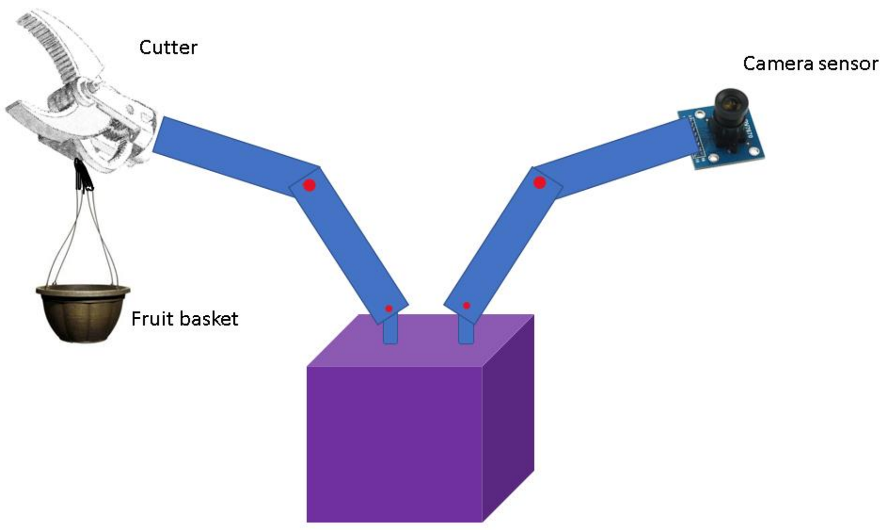


Figure 1 The dual-robotic arm system

Figure 1 shows the dual-robotic arm system of this work. One of the arms is for sensing and the other is for plucking. The robot will sense the ripe fruit and the other arm should pluck it if it ripened. This is a general workability of the dual-robotic system.

2.2 KINEMATIC ANALYSIS

- Kinematic analysis for 1 arm (same for the other arm)

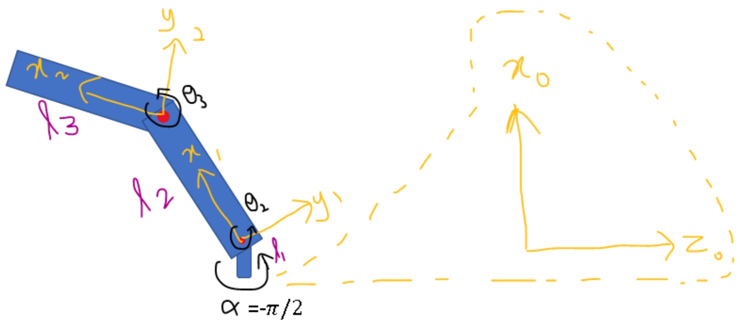
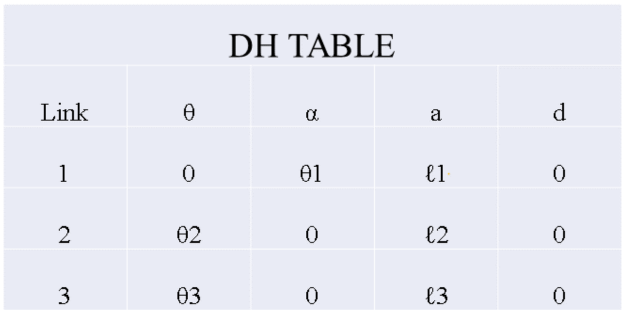


Figure 2 The kinematic diagram of the work

Figure 2 shows that the robotic arm has 3 degree of freedom and it is the same for the other arm.

2.3 DENAVIT- HARTENBERG PARAMETERS



DH TABLE

Link	θ	α	a	d
1	0	θ1	ℓ1	0
2	θ2	0	ℓ2	0
3	θ3	0	ℓ3	0

Table 2 D-H Table

2.3 JACOBIAN DERIVATION

Transformation Matrices

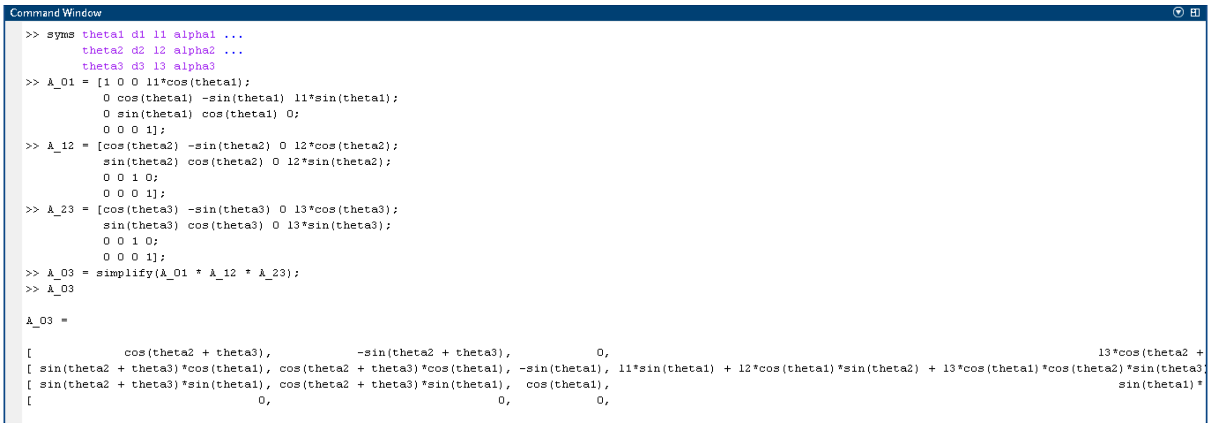
$$ {}^{0}T_{1}= \begin{bmatrix} 1 & 0 & 0 & l_1c_1 \\ 0 & c_1 & -s_1 & l_1s_1 \\ 0 & s_1 & c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}T_{2}= \begin{bmatrix} c_2 & -s_2 & 0 & l_2c_2 \\ s_2 & c_2 & 0 & l_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}T_{3}= \begin{bmatrix} c_3 & -s_3 & 0 & l_3c_3 \\ s_3 & c_3 & 0 & l_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Since,

$$ {}^{0}T_{3}= {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} $$



```
>> syms theta1 d1 l1 alpha1 ...
        theta2 d2 l2 alpha2 ...
        theta3 d3 l3 alpha3
>> A_01 = [1 0 0 l1*cos(theta1);
           0 cos(theta1) -sin(theta1) l1*sin(theta1);
           0 sin(theta1) cos(theta1) 0;
           0 0 0 1];
>> A_12 = [cos(theta2) -sin(theta2) 0 l2*cos(theta2);
           sin(theta2) cos(theta2) 0 l2*sin(theta2);
           0 0 1 0;
           0 0 0 1];
>> A_23 = [cos(theta3) -sin(theta3) 0 l3*cos(theta3);
           sin(theta3) cos(theta3) 0 l3*sin(theta3);
           0 0 1 0;
           0 0 0 1];
>> A_03 = simplify(A_01 * A_12 * A_23);
>> A_03

A_03 =

[             cos(theta2 + theta3),            -sin(theta2 + theta3),            0,                                                                   l3*cos(theta2 +
[ sin(theta2 + theta3)*cos(theta1), cos(theta2 + theta3)*cos(theta1), -sin(theta1), l1*sin(theta1) + l2*cos(theta1)*sin(theta2) + l3*cos(theta1)*cos(theta2)*sin(theta3
[ sin(theta2 + theta3)*sin(theta1), cos(theta2 + theta3)*sin(theta1),  cos(theta1),                                                                    sin(theta1)*
[                                0,                                0,            0,
```

Figure 3 MATLAB matrix calculation.

${}^{0}T_{3}$ is obtained by calculating the matrix multiplication in MATLAB as shown in figure 3.

Therefore,

$$ {}^{0}T_{3} = \begin{bmatrix} c_{23} & -s_{23} & 0 & l_3c_{23} + l_1c_1 + l_2c_2 \\ s_{23}c_1 & c_{23}c_1 & -s_1 & l_1s_1 + l_2c_1s_2 + l_3c_1c_2s_3 + l_3c_1c_3s_2 \\ s_{23}s_1 & c_{23}s_1 & c_1 & l_3s_1s_{23} + l_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

By taking the partial derivative of the translational matrix

$\frac{\partial P_x}{\partial \theta_1} = -l_1 s_1$

$\frac{\partial P_x}{\partial \theta_2} = -l_2 s_{23} - l_2 s_2$

$\frac{\partial P_x}{\partial \theta_3} = -l_2 s_{23}$

$\frac{\partial P_y}{\partial \theta_1} = l_1 c_1 - l_2 s_1 s_2 - l_3 s_1 c_2 c_3 - l_3 s_1 s_2 c_3$

$\frac{\partial P_y}{\partial \theta_2} = l_2 c_1 c_2 - l_3 c_1 s_2 s_3 + l_3 c_1 c_2 c_3$

$\frac{\partial P_y}{\partial \theta_3} = l_3 c_1 c_2 c_3 - l_3 c_1 s_2 s_3$

$\frac{\partial P_z}{\partial \theta_1} = l_3 c_1 s_{23} + l_2 c_1 s_2$

$\frac{\partial P_z}{\partial \theta_2} = l_3 s_1 c_{23} + l_2 s_1 c_2$

$\frac{\partial P_z}{\partial \theta_3} = l_3 s_1 c_{23}$

According to analytical

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_A} & \frac{\partial p_x}{\partial \theta_B} & \frac{\partial p_x}{\partial \theta_C} \\ \frac{\partial p_y}{\partial \theta_A} & \frac{\partial p_y}{\partial \theta_B} & \frac{\partial p_y}{\partial \theta_C} \\ \frac{\partial p_z}{\partial \theta_A} & \frac{\partial p_z}{\partial \theta_B} & \frac{\partial p_z}{\partial \theta_C} \end{bmatrix}$$

Figure 4 Analytical Jacobian Matrix

Based on figure 4, the final Jacobian Matrix is,

$$J = \begin{bmatrix} -l_1 s_1 & -l_2 s_{23} - l_2 s_2 & -l_2 s_{23} \\ l_1 c_1 - l_2 s_1 s_2 - l_3 s_1 c_2 c_3 - l_3 s_1 s_2 c_3 & l_2 c_1 c_2 - l_3 c_1 s_2 s_3 + l_3 c_1 c_2 c_3 & l_3 c_1 c_2 c_3 - l_3 c_1 s_2 s_3 \\ l_3 c_1 s_{23} + l_2 c_1 s_2 & l_3 s_1 c_{23} + l_2 s_1 c_2 & l_3 s_1 c_{23} \end{bmatrix}$$

3. RESULTS & DISCUSSIONS

3.1 Dynamic Analysis

3.1.1 Path Planning

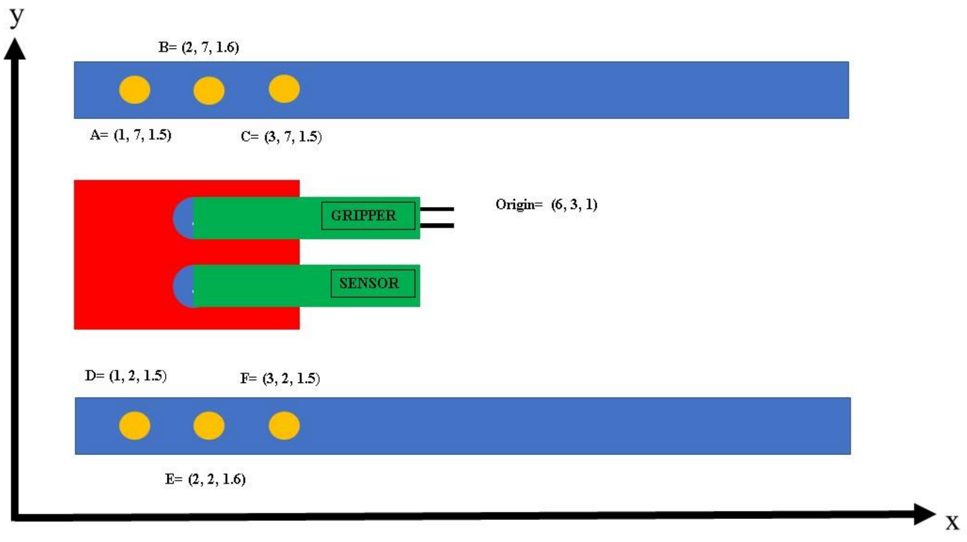
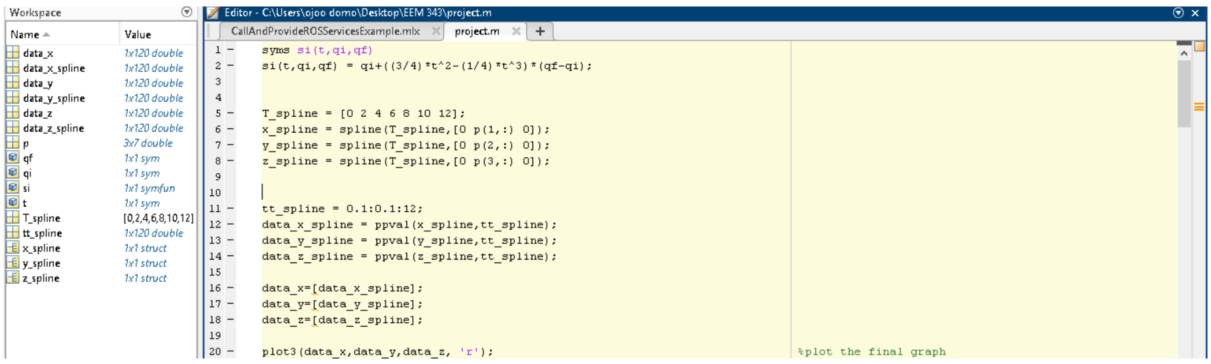


Figure 5 The orientation of the robot and fruits for analysis

The robot will begin to pluck the fruits at these given coordinates,

Initial End effector	[6,3,1]
Fruit A	[1,7,1.5]
Fruit B	[2,7,1.6]
Fruit C	[3,7,1.5]
Fruit D	[1,2,1.5]
Fruit E	[2,2,1.6]
Fruit F	[3,2,1.5]

Assuming all the fruits ripened, both of the robotic arm will have the same trajectory. **Assuming the time needed for each segment** is 2 seconds, the robot should be able to sense all the fruits within 12 seconds and pluck it within another 12 seconds. MATLAB is used to plot a path to visualize the trajectory for both of the robotic arm. Assuming all the fruits are ripened and pluckable, the trajectory of the arms should be the same.


Workspace
Editor - C:\Users\ojoo domo\Desktop\EEM 343\project.m
CallAndProvideROSServicesExample.mlx
project.m

Name	Value
data_x	1x120 double
data_x_spline	1x120 double
data_y	1x120 double
data_y_spline	1x120 double
data_z	1x120 double
data_z_spline	1x120 double
p	3x7 double
qf	1x1 sym
qi	1x1 sym
si	1x1 symfun
t	1x1 sym
T_spline	[0,2,4,6,8,10,12]
tt_spline	1x120 double
x_spline	1x1 struct
y_spline	1x1 struct
z_spline	1x1 struct

```
syms si(t,qi,qf)
si(t,qi,qf) = qi+((3/4)*t^2-(1/4)*t^3)*(qf-qi);


T_spline = [0 2 4 6 8 10 12];
x_spline = spline(T_spline,[0 p(1,:) 0]);
y_spline = spline(T_spline,[0 p(2,:) 0]);
z_spline = spline(T_spline,[0 p(3,:) 0]);


tt_spline = 0.1:0.1:12;
data_x_spline = ppval(x_spline,tt_spline);
data_y_spline = ppval(y_spline,tt_spline);
data_z_spline = ppval(z_spline,tt_spline);

data_x=[data_x_spline];
data_y=[data_y_spline];
data_z=[data_z_spline];

plot3(data_x,data_y,data_z, 'r');                                    %plot the final graph
```

Figure 6 MATLAB code for trajectory planning

Figure 6 shows the MATLAB code to plot the 3-Dimensional plot of the trajectory of the arms. Assuming all the fruits are ripened, the arms should have the same trajectory. The sensor arm will move first to sense all the fruit. After that, the gripper arm will move with the same trajectory to pluck the fruits one by one. The coordinates of the fruits are keyed in the 'p' variable and spline is used to make a curved path instead of just a straight line. In **figure 7,** the result of the trajectory was plotted by using MATLAB

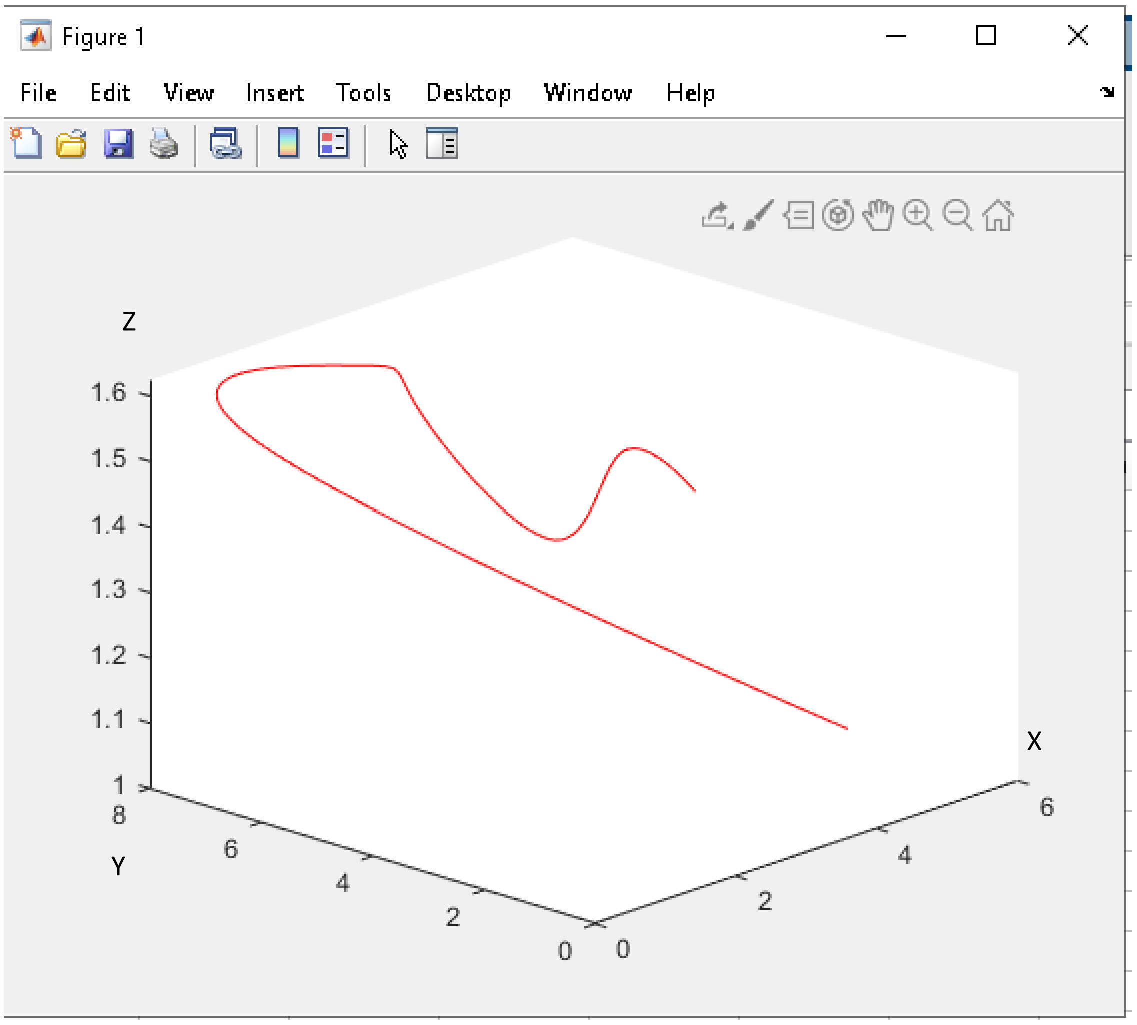


Figure 7 The trajectory of the robotic arms

3.1.2 Lagrangian Equation

$X_D = \frac{l_3}{2} c_{23} + l_1 c_1 + l_2 c_2$

$Y_D = l_1 s_1 + l_2 c_1 s_2 + \frac{l_3}{2} c_1 c_2 s_3 + \frac{l_3}{2} c_1 c_3 s_2$

$Z_D = \frac{l_3}{2} s_1 s_{23} + l_2 s_2$

Velocity

$v^2_D = \dot{X}^2_D + \dot{Y}^2_D + \dot{Z}^2_D$

= D2*sin(theta1)*((l3*cos(theta2 + theta3))/2 + l2*cos(theta2)) + D1*cos(theta1)*((l3*sin(theta2 + theta3))/2 + l2*sin(theta2)) + (D3*l3*cos(theta2 + theta3)*sin(theta1))/2)^2 + (D2*((l3*sin(theta2 + theta3))/2 + l2*sin(theta2)) + (D3*l3*sin(theta2 + theta3))/2 + D1*l1*sin(theta1))^2 + (D2*(l2*cos(theta1)*cos(theta2) + (l3*cos(theta1)*cos(theta2)*cos(theta3))/2 - (l3*cos(theta1)*sin(theta2)*sin(theta3))/2) -

D1*(l2*sin(theta1)*sin(theta2) - l1*cos(theta1) + (l3*cos(theta2)*sin(theta1)*sin(theta3))/2 + (l3*cos(theta3)*sin(theta1)*sin(theta2))/2) + D3*((l3*cos(theta1)*cos(theta2)*cos(theta3))/2 - (l3*cos(theta1)*sin(theta2)*sin(theta3))/2))^2

Where, D1, D2, D3 is $\dot{\theta}_1$, $\dot{\theta}_2$, $\dot{\theta}_3$

Kinetic Energy =

(D1^2*l1^2*m1)/6 + (l1^2*m1*(D1^2 + D2^2))/6 + (L3^2*m3*((D2 + D3)^2 + D1^2))/24

Potential Energy = (981*m3*(L1 + sin(theta2 + theta3)*(l2 + l3/2)))/100 + (981*l1*m1)/200 + (981*m2*(l1 + (l2*sin(theta2))/2))/100

Lagrangian Equation = (D1^2*l1^2*m1)/6 - (981*l1*m1)/200 - (981*m2*(l1 + (l2*sin(theta2))/2))/100 - (981*m3*(L1 + sin(theta2 + theta3)*(l2 + l3/2)))/100 + (l1^2*m1*(D1^2 + D2^2))/6 + (L3^2*m3*((D2 + D3)^2 + D1^2))/24.

All of these parameters are calculated by using MATLAB and the codes are available in the appendices.

3.2 ROS Simulation

3.2.1 Block Diagram

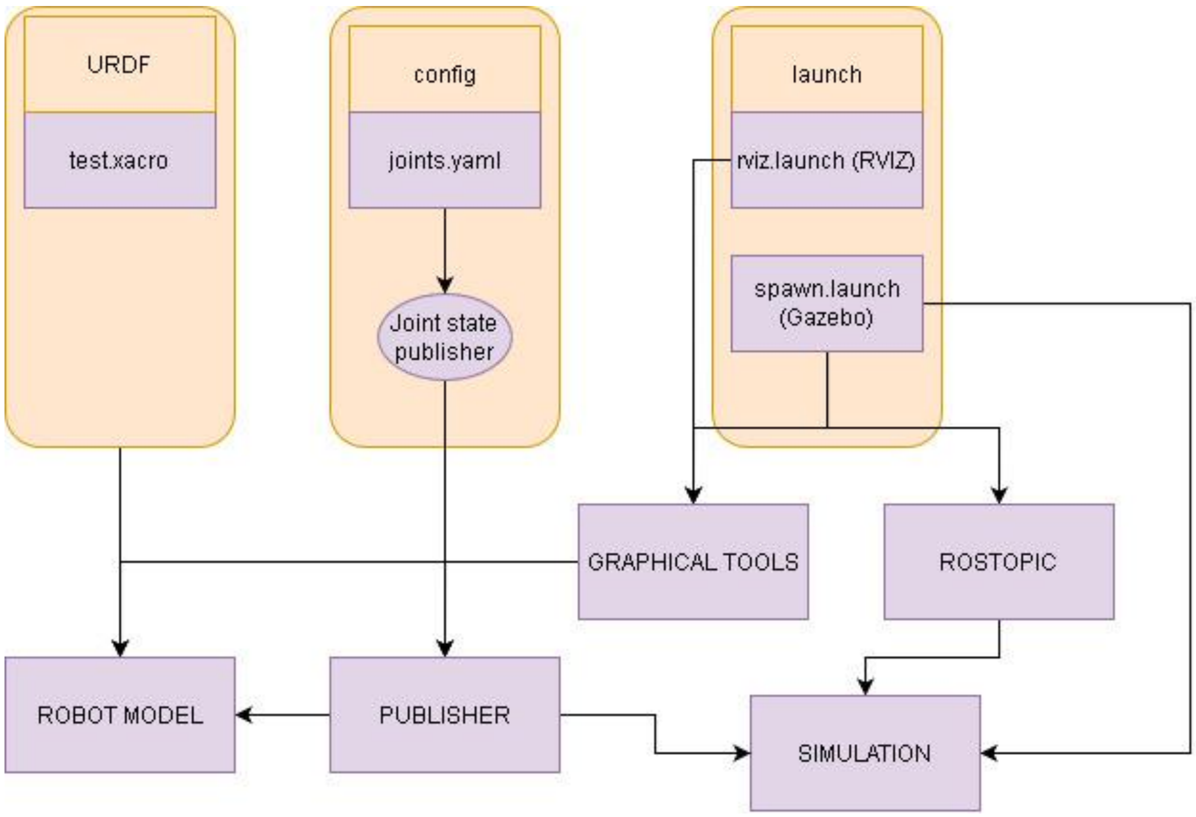
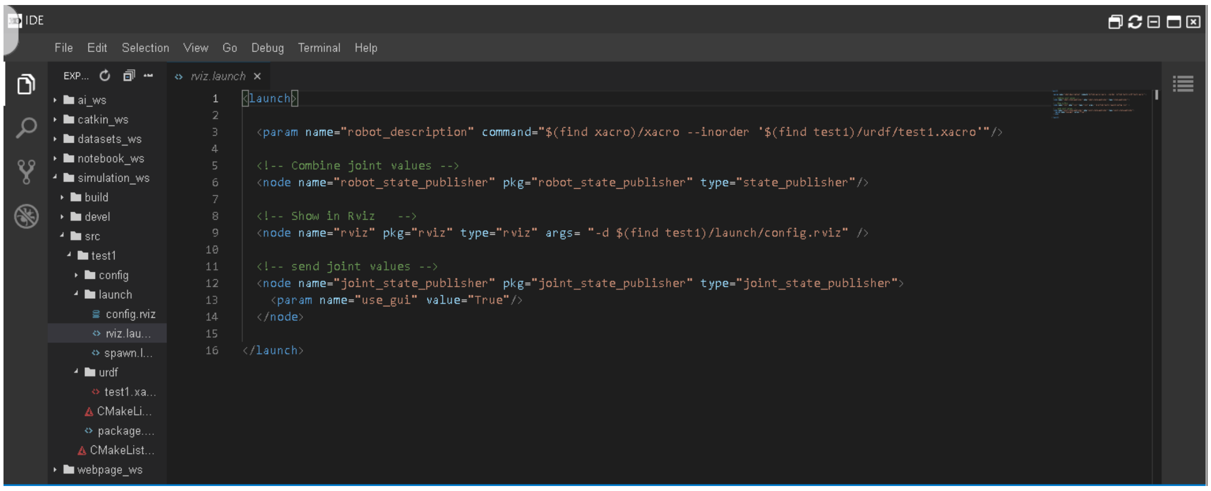


Figure 8 The block diagram of this project

In the URDF folder, there is a main program code for this robot which is test.xacro. This code contains all the constrains, mass and inertia of the robot. In the config folder, all the joints in the robot was defined to be simulated in Gazebo. In the launch folder, there are two launch files which is for the RVIZ and Gazebo. ROS Development Studios is used for this project.

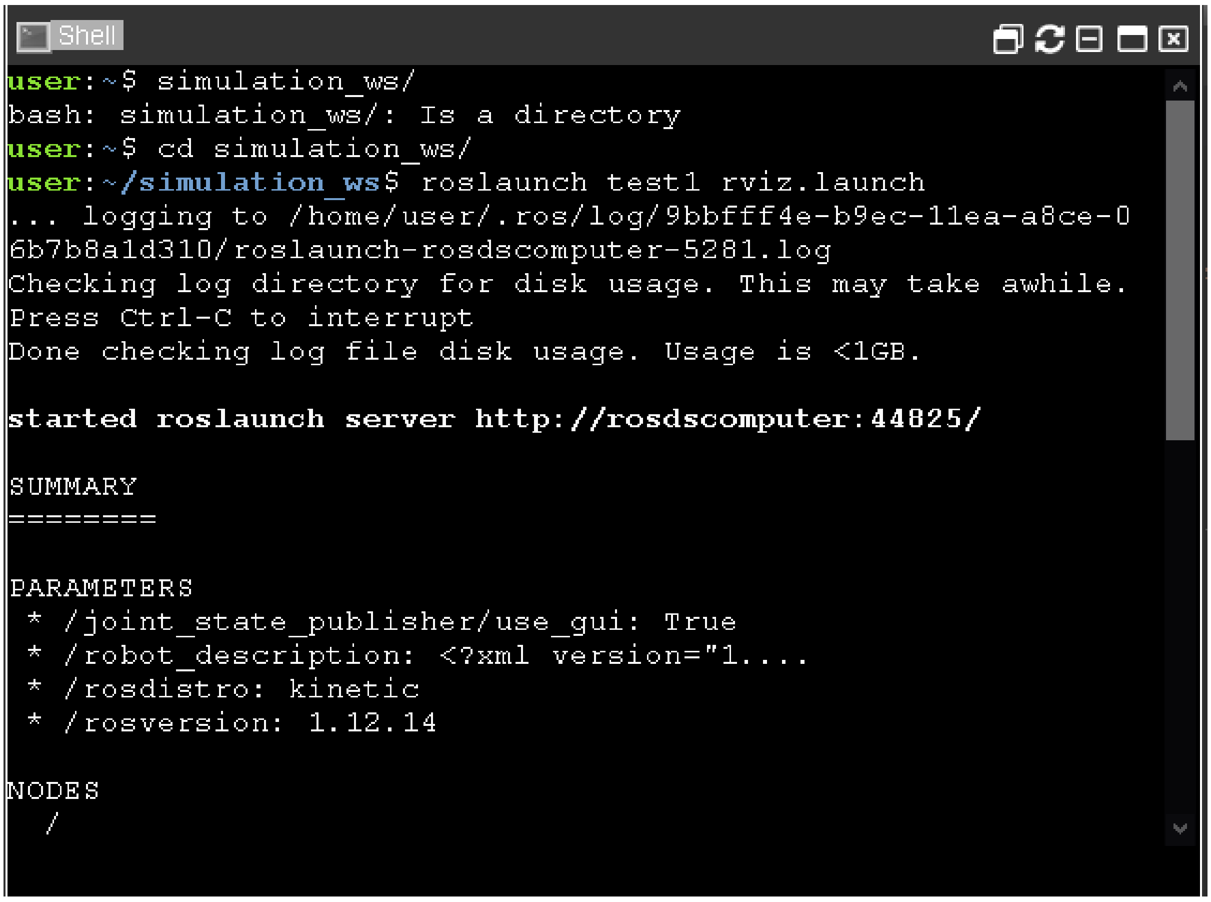
3.2.2 RVIZ

In RVIZ, the robot model was shown and it is to ease us for troubleshooting the robot's code in URDF. **Figure 9** shows the rviz.launch file. This file is necessary to spawn a robot model in RVIZ. This file needs to be called as shown in **figure 10**. **Figure 11** shows the robot model in RVIZ. Not only that it can visualize the robot at one specific position, RVIZ can also move the links of the robot analogous using the Joint-state publisher as shown in **figure 13. Figure 12** shows the joint-state publisher.



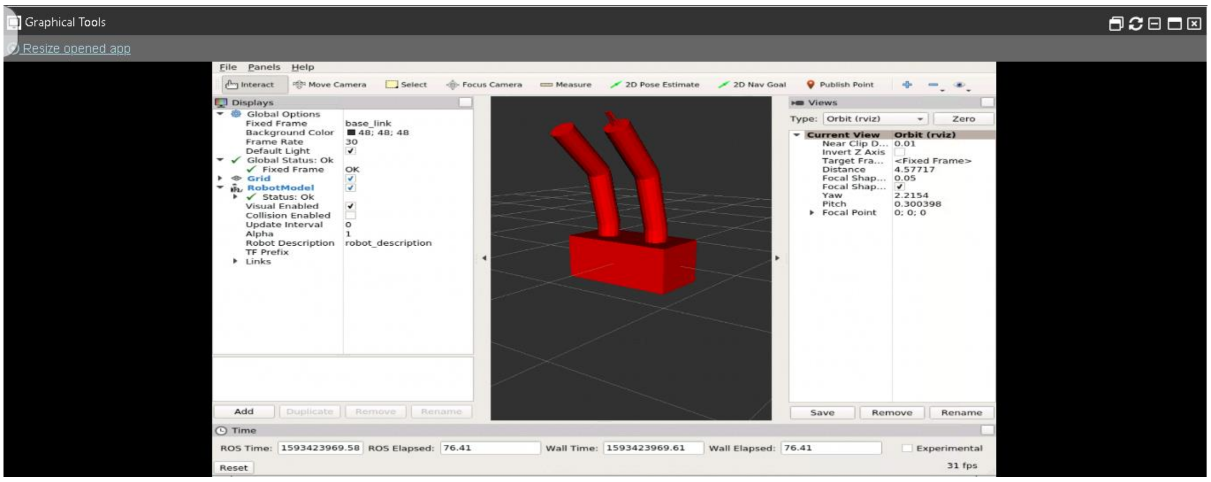
```
<launch>

  <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find test1)/urdf/test1.xacro'"/>

  <!-- Combine joint values -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher"/>

  <!-- Show in Rviz   -->
  <node name="rviz" pkg="rviz" type="rviz" args= "-d $(find test1)/launch/config.rviz" />

  <!-- send joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="use_gui" value="True"/>
  </node>

</launch>
```

Figure 9 The rviz.launch file



```
user:~$ simulation_ws/
bash: simulation_ws/: Is a directory
user:~$ cd simulation_ws/
user:~/simulation_ws$ roslaunch test1 rviz.launch
... logging to /home/user/.ros/log/9bbfff4e-b9ec-11ea-a8ce-0
6b7b8a1d310/roslaunch-rosdscomputer-5281.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rosdscomputer:44825/

SUMMARY
========

PARAMETERS
 * /joint_state_publisher/use_gui: True
 * /robot_description: <?xml version="1....
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
```

Figure 10 Spawning robot in the RVIZ

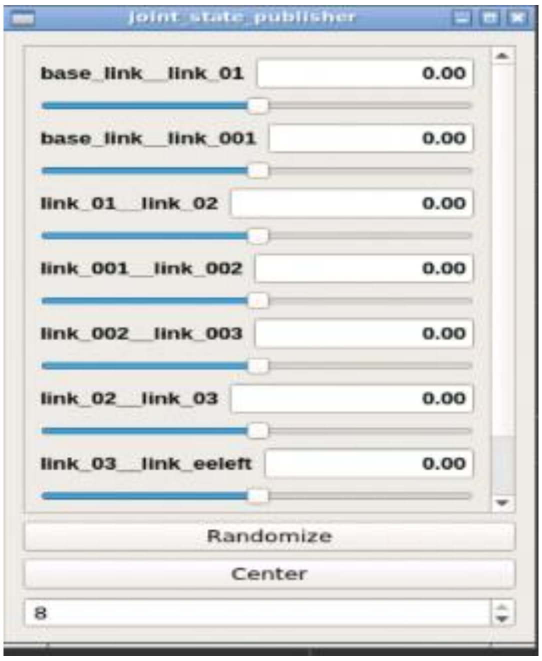


```
Displays
Global Options
  Fixed Frame            base_link
  Background Color       48; 48; 48
  Frame Rate             30
  Default Light          ✓
Global Status: Ok
  Fixed Frame            OK
Grid                     ✓
RobotModel               ✓
  Status: Ok
  Visual Enabled         ✓
  Collision Enabled
  Update Interval        0
  Alpha                  1
  Robot Description      robot_description
  TF Prefix
  Links
```

```
Views
Type: Orbit (rviz)    Zero
Current View     Orbit (rviz)
  Near Clip D...   0.01
  Invert Z Axis
  Target Fra...    <Fixed Frame>
  Distance         4.57717
  Focal Shap...    0.05
  Focal Shap...    ✓
  Yaw              2.2154
  Pitch            0.300398
  Focal Point      0; 0; 0
```

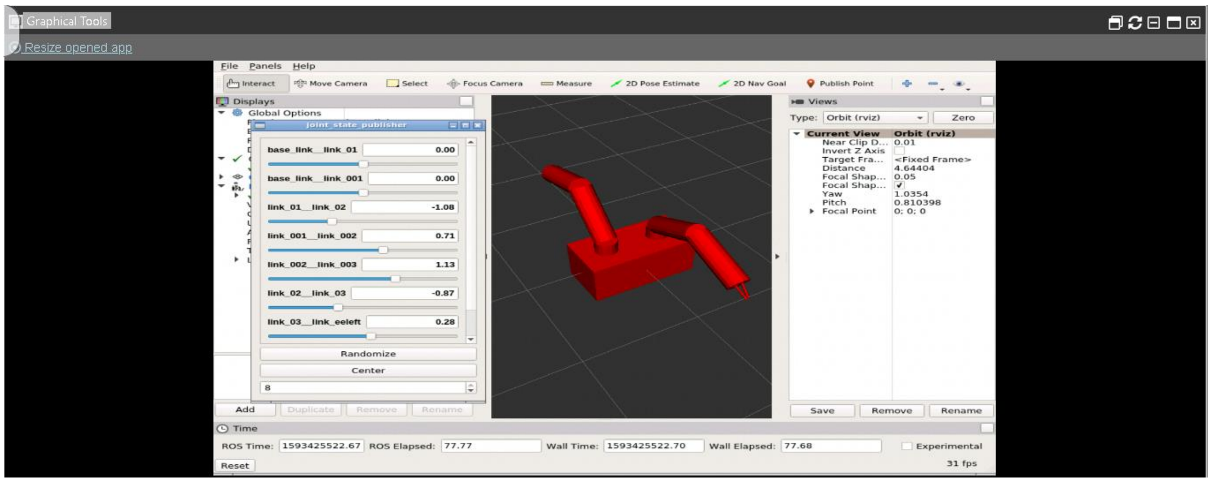
```
Time
ROS Time: 1593423969.58  ROS Elapsed: 76.41  Wall Time: 1593423969.61  Wall Elapsed: 76.41  Experimental
31 fps
```

Figure 11 The dual robotic arm in RVIZ



```
base_link__link_01      0.00
base_link__link_001     0.00
link_01__link_02        0.00
link_001__link_002      0.00
link_002__link_003      0.00
link_02__link_03        0.00
link_03__link_eeleft    0.00
8
```

Figure 12 The joint-state publisher

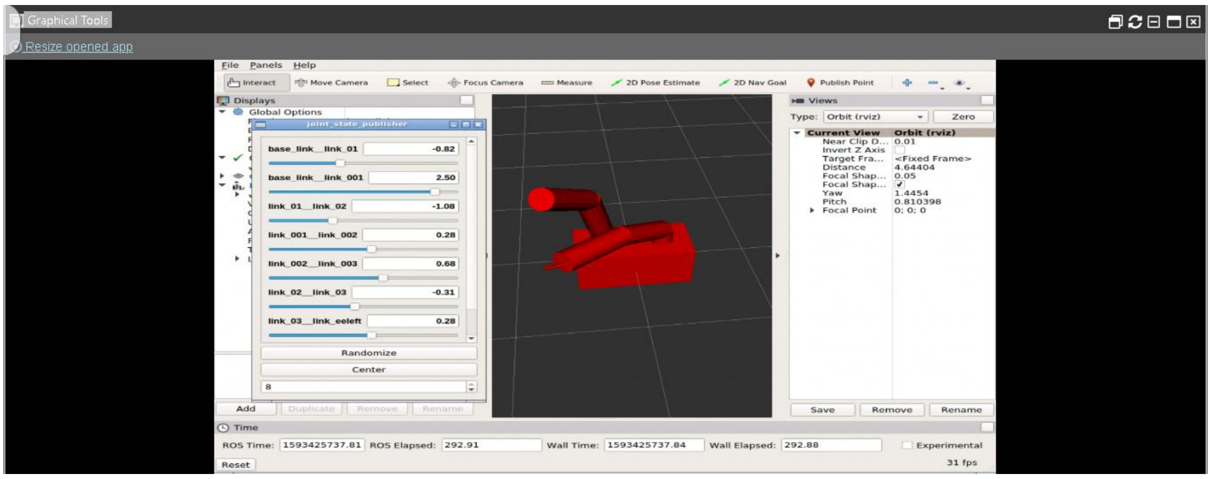

joint_state_publisher

base_link__link_01 0.00
base_link__link_001 0.00
link_01__link_02 -1.08
link_001__link_002 0.71
link_002__link_003 1.13
link_02__link_03 -0.87
link_03__link_eeleft 0.28

Type: Orbit (rviz)
Current View: Orbit (rviz)
Near Clip D... 0.01
Target Fra... <Fixed Frame>
Distance 4.64404
Focal Shap... 0.05
Yaw 1.0354
Pitch 0.810398
Focal Point 0; 0; 0

ROS Time: 1593425522.67 ROS Elapsed: 77.77 Wall Time: 1593425522.70 Wall Elapsed: 77.68

31 fps

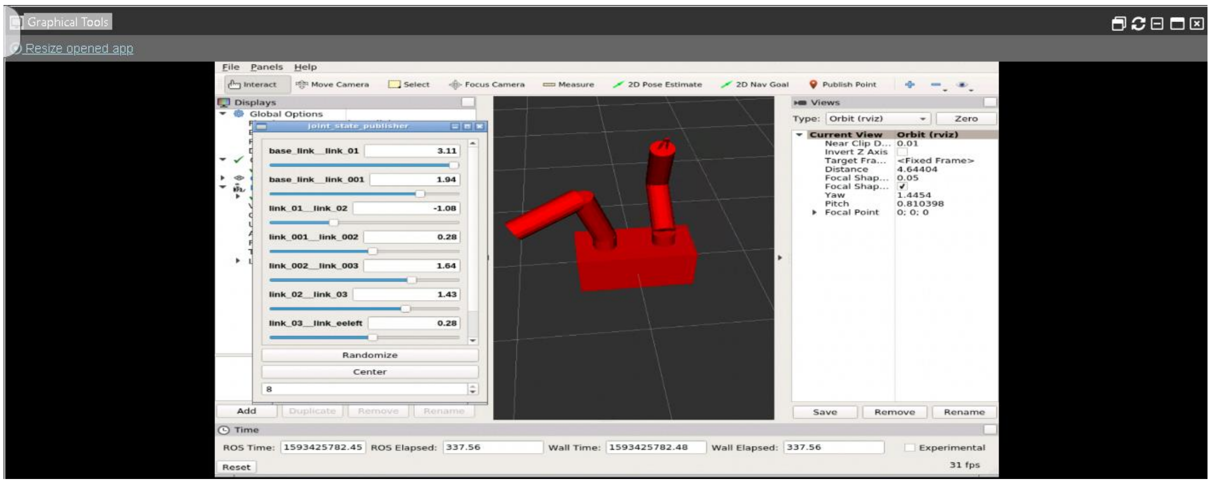

joint_state_publisher

base_link__link_01 -0.82
base_link__link_001 2.50
link_01__link_02 -1.08
link_001__link_002 0.28
link_002__link_003 0.68
link_02__link_03 -0.31
link_03__link_eeleft 0.28

Type: Orbit (rviz)
Current View: Orbit (rviz)
Near Clip D... 0.01
Target Fra... <Fixed Frame>
Distance 4.64404
Focal Shap... 0.05
Yaw 1.4454
Pitch 0.810398
Focal Point 0; 0; 0

ROS Time: 1593425737.81 ROS Elapsed: 292.91 Wall Time: 1593425737.84 Wall Elapsed: 292.88

31 fps


joint_state_publisher

base_link__link_01 3.11
base_link__link_001 1.94
link_01__link_02 -1.08
link_001__link_002 0.28
link_002__link_003 1.64
link_02__link_03 1.43
link_03__link_eeleft 0.28

Type: Orbit (rviz)
Current View: Orbit (rviz)
Near Clip D... 0.01
Target Fra... <Fixed Frame>
Distance 4.64404
Focal Shap... 0.05
Yaw 1.4454
Pitch 0.810398
Focal Point 0; 0; 0

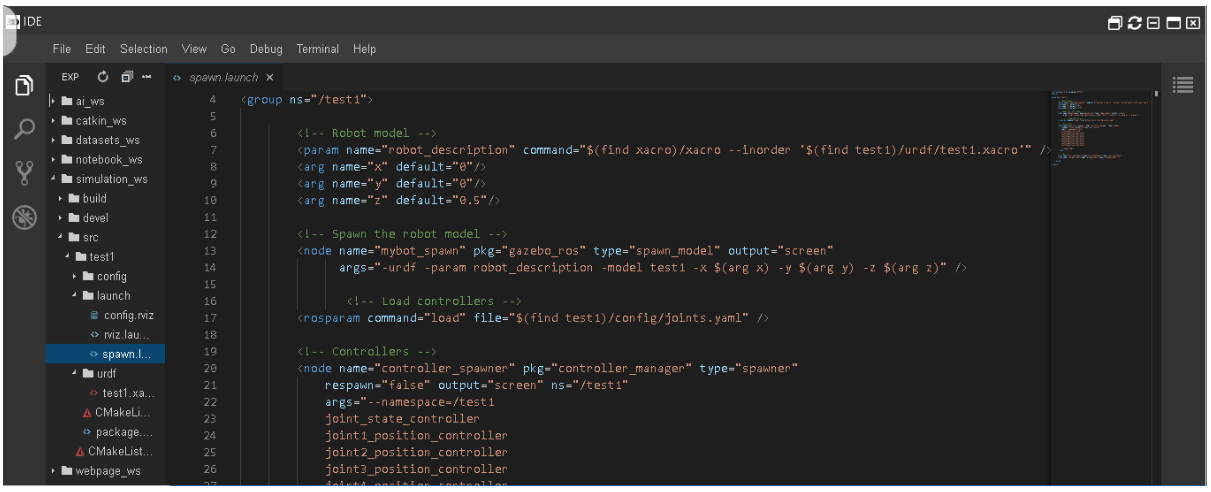
ROS Time: 1593425782.45 ROS Elapsed: 337.56 Wall Time: 1593425782.48 Wall Elapsed: 337.56

31 fps

Figure 13 Various position of the robotic arm

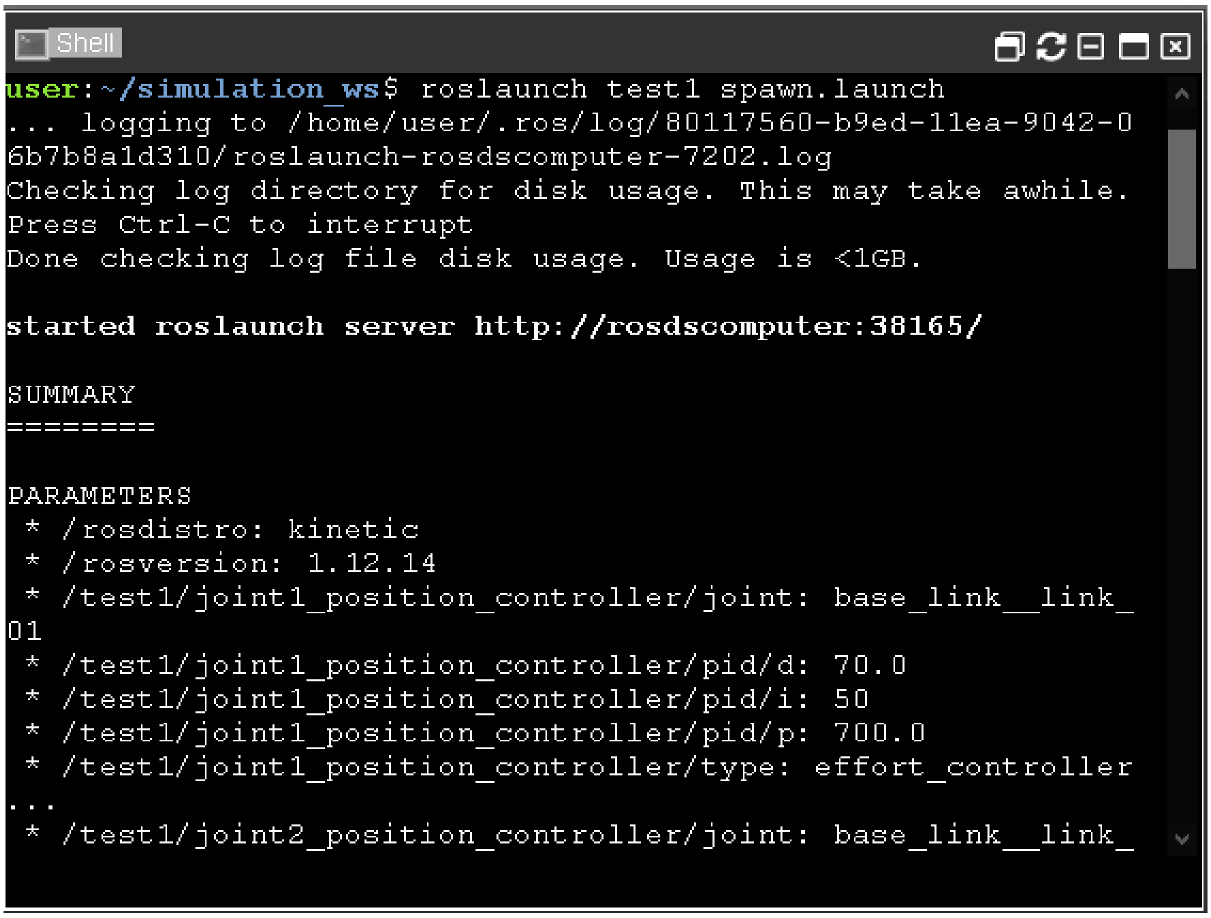
3.2.3 Gazebo

This platform is more suitable for simulating the robot. In here we can easily make a world to visualize our robot at work. **Figure 14** shows the spawn.launch file to spawn the robot model inside the Gazebo. In order to call the file, we need to call the function as shown in **Figure 15. Figure 16** shows the robot model in the Gazebo world. This configuration can be changed by putting some values in radian to the rqt publisher as shown in **Figure 18. Figure 17** shows the rqt publisher used to adjust the robot configuration.



```
<group ns="/test1">

        <!-- Robot model -->
        <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find test1)/urdf/test1.xacro'" />
        <arg name="x" default="0"/>
        <arg name="y" default="0"/>
        <arg name="z" default="0.5"/>

        <!-- Spawn the robot model -->
        <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
              args="-urdf -param robot_description -model test1 -x $(arg x) -y $(arg y) -z $(arg z)" />

               <!-- Load controllers -->
        <rosparam command="load" file="$(find test1)/config/joints.yaml" />

        <!-- Controllers -->
        <node name="controller_spawner" pkg="controller_manager" type="spawner"
            respawn="false" output="screen" ns="/test1"
            args="--namespace=/test1
            joint_state_controller
            joint1_position_controller
            joint2_position_controller
            joint3_position_controller
```

Figure 14 Part of spawn.launch file



```
user:~/simulation_ws$ roslaunch test1 spawn.launch
... logging to /home/user/.ros/log/80117560-b9ed-11ea-9042-0
6b7b8a1d310/roslaunch-rosdscomputer-7202.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rosdscomputer:38165/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14
 * /test1/joint1_position_controller/joint: base_link__link_
01
 * /test1/joint1_position_controller/pid/d: 70.0
 * /test1/joint1_position_controller/pid/i: 50
 * /test1/joint1_position_controller/pid/p: 700.0
 * /test1/joint1_position_controller/type: effort_controller
...
 * /test1/joint2_position_controller/joint: base_link__link_
```

Figure 15 Spawning robot in Gazebo

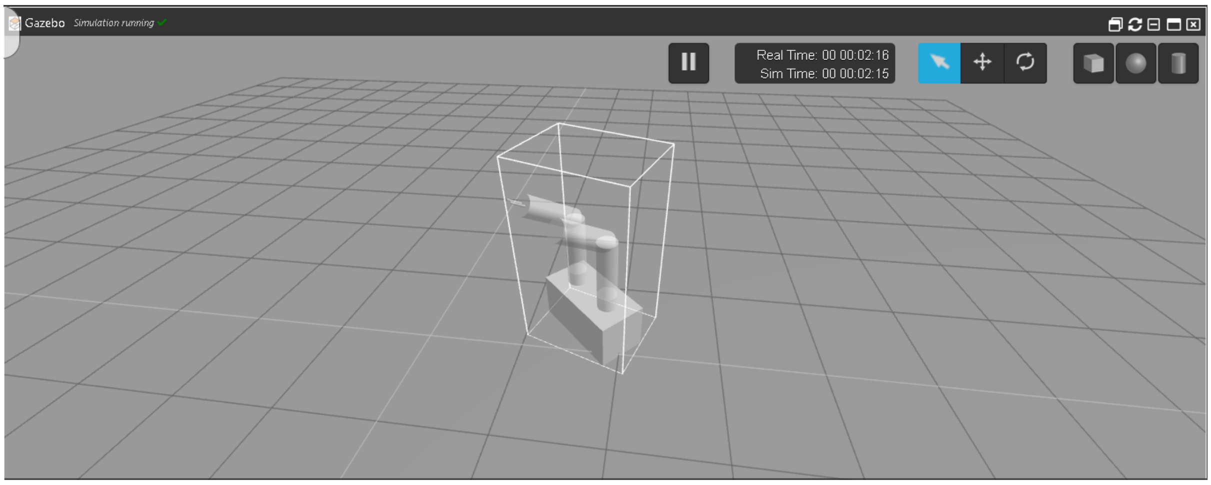
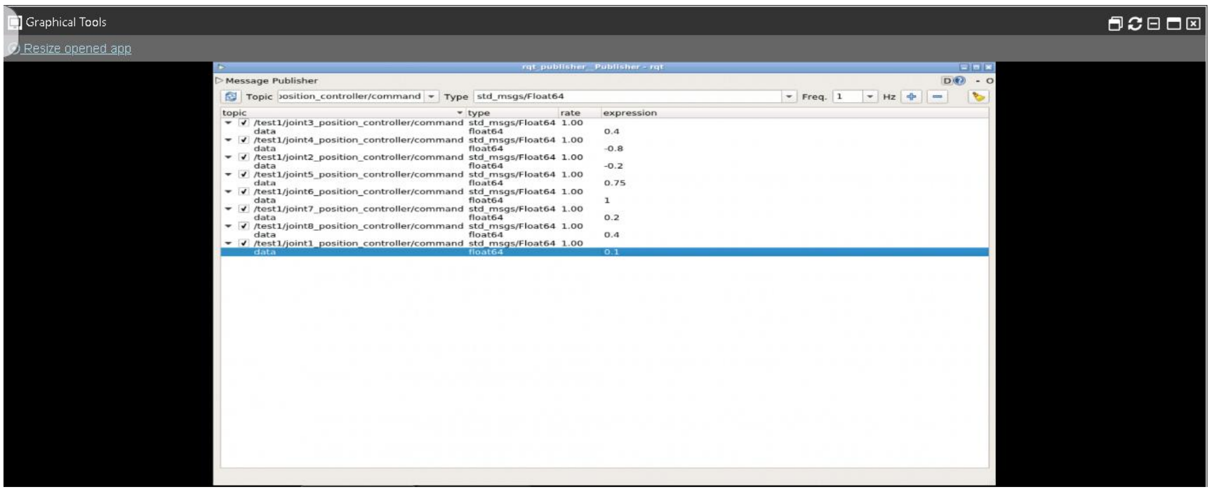


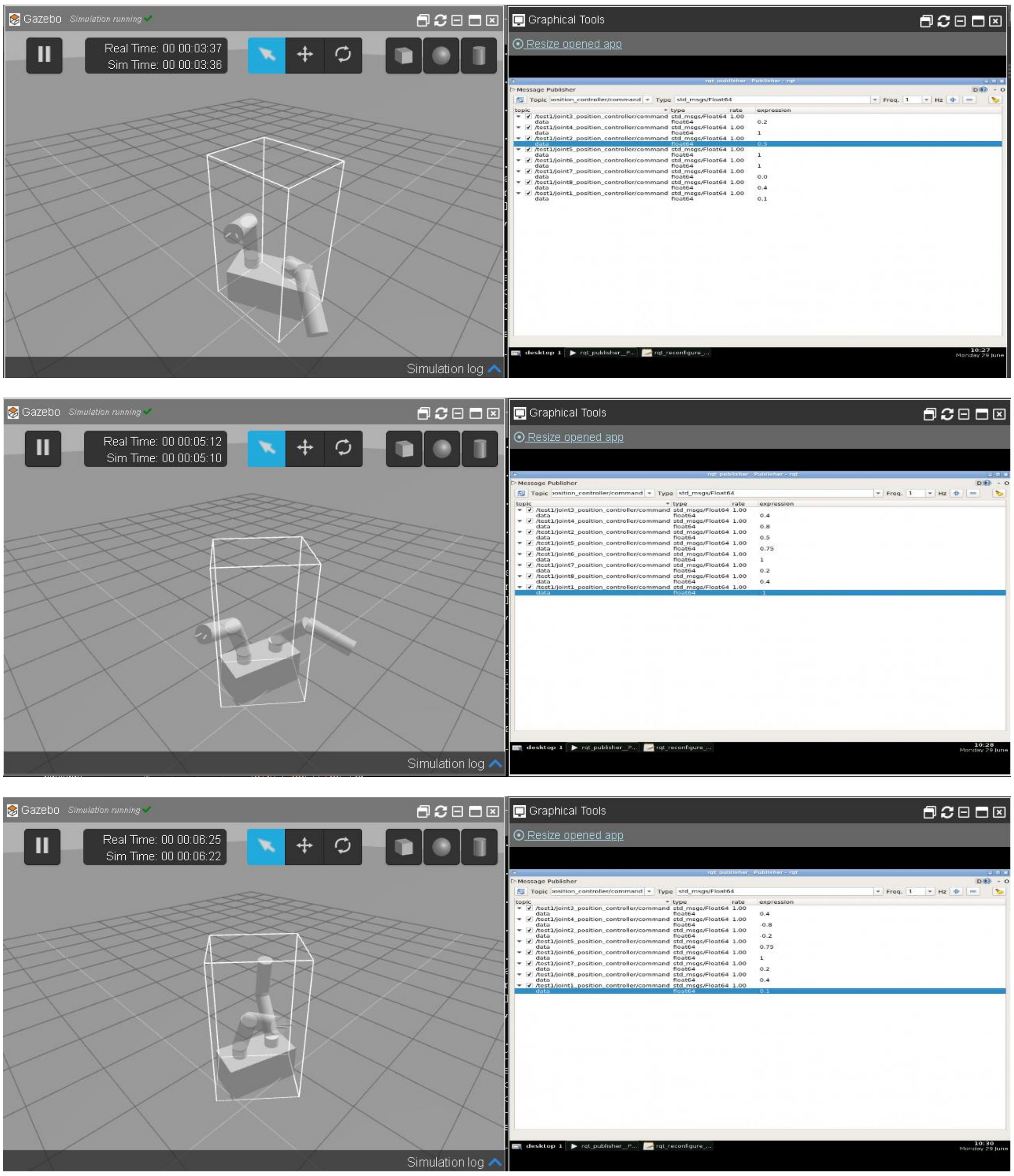
Figure 16 The dual robotic arm in Gazebo



Topic: position_controller/command Type: std_msgs/Float64 Freq.: 1 Hz

topic	type	rate	expression
✓ /test1/joint3_position_controller/command	std_msgs/Float64	1.00	
data	float64		0.4
✓ /test1/joint4_position_controller/command	std_msgs/Float64	1.00	
data	float64		-0.8
✓ /test1/joint2_position_controller/command	std_msgs/Float64	1.00	
data	float64		-0.2
✓ /test1/joint5_position_controller/command	std_msgs/Float64	1.00	
data	float64		0.75
✓ /test1/joint6_position_controller/command	std_msgs/Float64	1.00	
data	float64		1
✓ /test1/joint7_position_controller/command	std_msgs/Float64	1.00	
data	float64		0.2
✓ /test1/joint8_position_controller/command	std_msgs/Float64	1.00	
data	float64		0.4
✓ /test1/joint1_position_controller/command	std_msgs/Float64	1.00	
data	float64		0.1

Figure 17 The rqt publisher



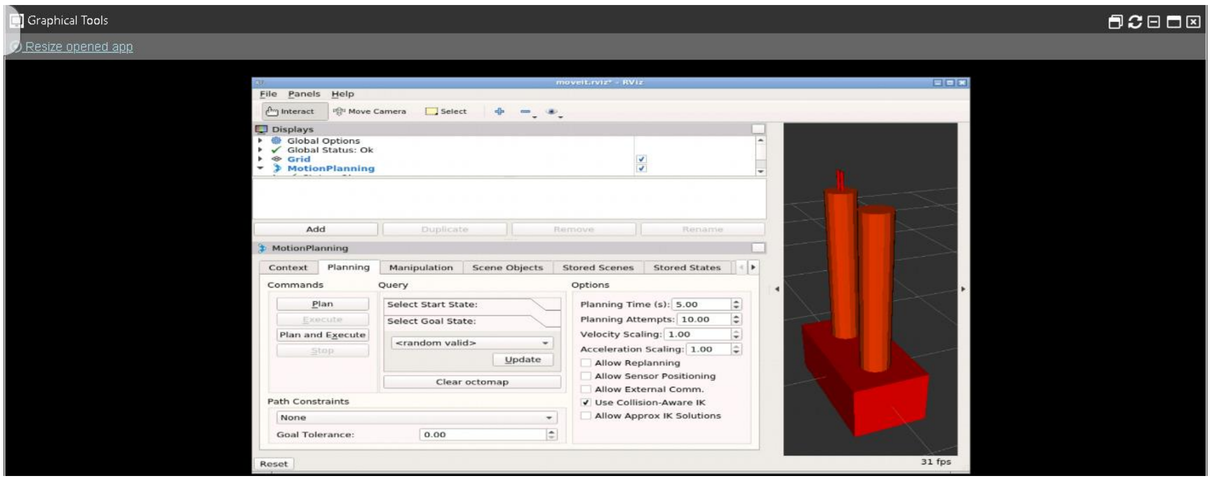
```
Message Publisher
Topic  osition_controller/command   Type  std_msgs/Float64   Freq.  1   Hz
topic                                        type              rate   expression
/test1/joint3_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.2
/test1/joint4_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
/test1/joint2_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.5
/test1/joint5_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
/test1/joint6_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
/test1/joint7_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.0
/test1/joint8_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.4
/test1/joint1_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.1
```

```
Message Publisher
Topic  osition_controller/command   Type  std_msgs/Float64   Freq.  1   Hz
topic                                        type              rate   expression
/test1/joint3_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.4
/test1/joint4_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.8
/test1/joint2_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.5
/test1/joint5_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.75
/test1/joint6_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
/test1/joint7_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.2
/test1/joint8_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.4
/test1/joint1_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
```

```
Message Publisher
Topic  osition_controller/command   Type  std_msgs/Float64   Freq.  1   Hz
topic                                        type              rate   expression
/test1/joint3_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.4
/test1/joint4_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  -0.8
/test1/joint2_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  -0.2
/test1/joint5_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.75
/test1/joint6_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  1
/test1/joint7_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.2
/test1/joint8_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.4
/test1/joint1_position_controller/command   std_msgs/Float64  1.00
  data                                       float64                  0.1
```

Figure 18 Various configurations of the robotic arm controlled by rqt publisher

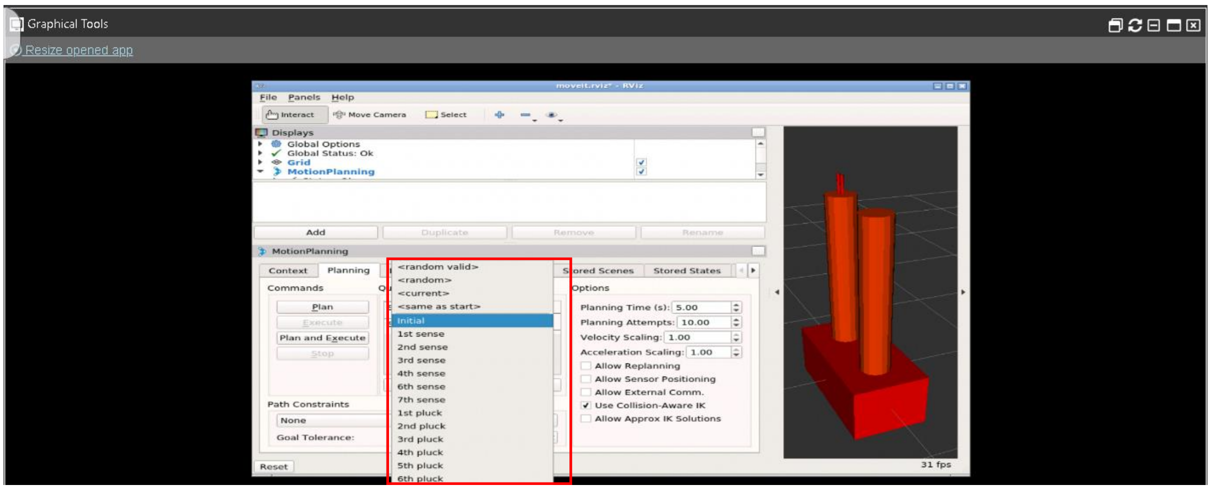
3.2.3 Moveit



Global Status: Ok
Select Start State:
Select Goal State: <random valid>
None
Goal Tolerance: 0.00
Planning Time (s): 5.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
✓ Use Collision-Aware IK
Allow Approx IK Solutions
31 fps

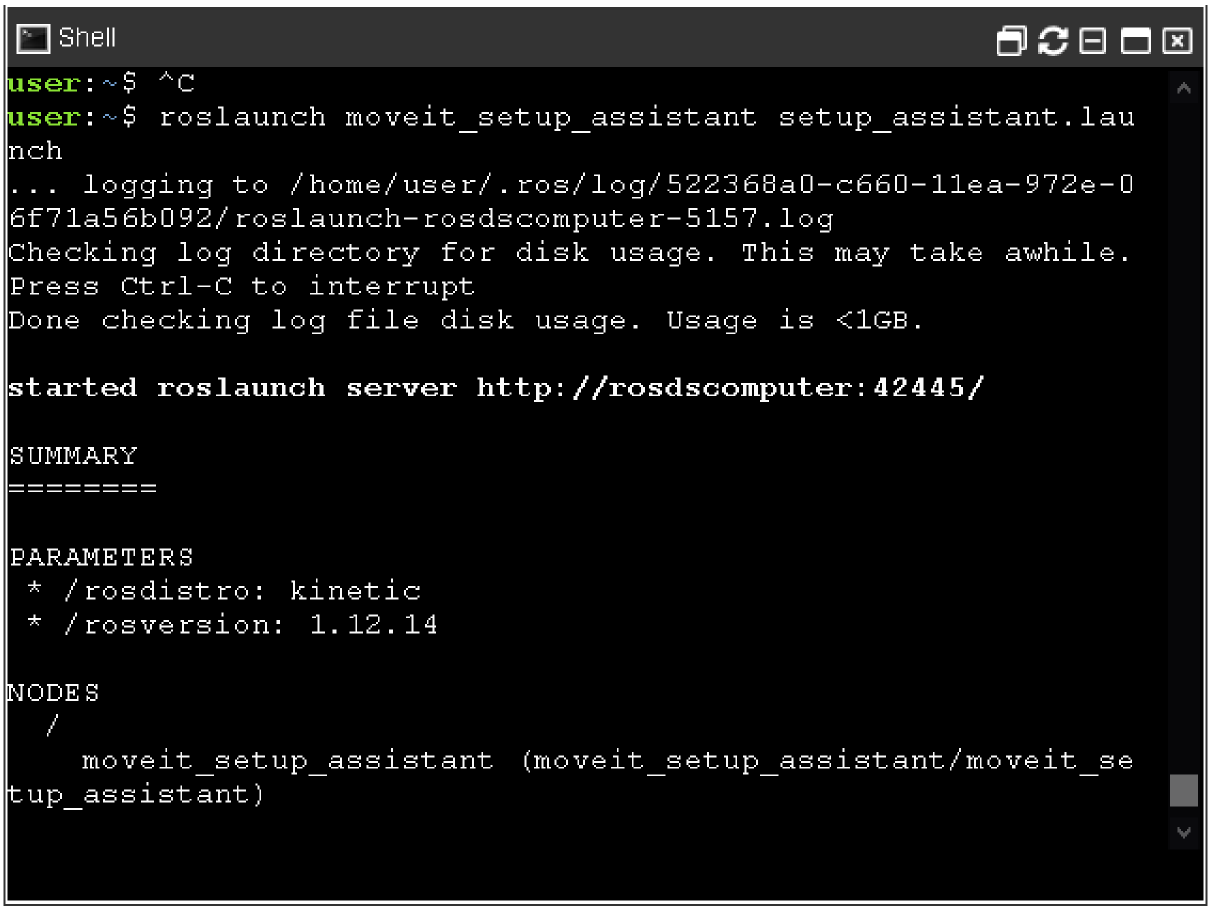
Figure 19 Moveit interface

We can simulate the robot in moveit by selecting the goal state of what we desire as shown in **Figure 20**. The goal state can be altered by using moveit setup assistant as shown in **Figure 21** and **Figure 22**.



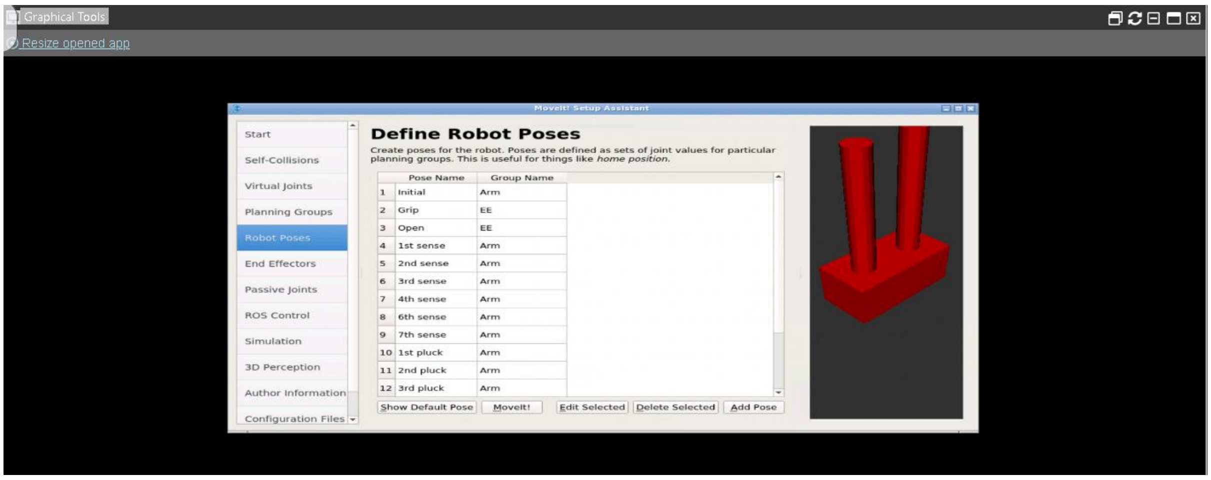
Global Status: Ok
None
Goal Tolerance:
Planning Time (s): 5.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
✓ Use Collision-Aware IK
Allow Approx IK Solutions
31 fps

Figure 20 Goal state selection



```
user:~$ ^C
user:~$ roslaunch moveit_setup_assistant setup_assistant.lau
nch
... logging to /home/user/.ros/log/522368a0-c660-11ea-972e-0
6f71a56b092/roslaunch-rosdscomputer-5157.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rosdscomputer:42445/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    moveit_setup_assistant (moveit_setup_assistant/moveit_se
tup_assistant)
```

Figure 21 Moveit setup assistant command



Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position*.

	Pose Name	Group Name
1	Initial	Arm
2	Grip	EE
3	Open	EE
4	1st sense	Arm
5	2nd sense	Arm
6	3rd sense	Arm
7	4th sense	Arm
8	6th sense	Arm
9	7th sense	Arm
10	1st pluck	Arm
11	2nd pluck	Arm
12	3rd pluck	Arm

Figure 22 Moveit Robot Poses

We can actually define the robot position as much as we want to accomplish our job for example plucking fruits. I have included 6 position of sensing arm and 6 position of gripping arm and an initial state of the robot. These 13 poses is enough for the robot to accomplish my work as shown in **Figure 5**.

4. APPENDIX

4.1 MATLAB Source Code

4.1.1 Jacobian Calculations

```
syms theta1 d1 l1 alpha1 ...
     theta2 d2 l2 alpha2 ...
     theta3 d3 l3 alpha3
>> A_01 = [1 0 0 l1*cos(theta1);
        0 cos(theta1) -sin(theta1) l1*sin(theta1);
        0 sin(theta1) cos(theta1) 0;
        0 0 0 1];
>> A_12 = [cos(theta2) -sin(theta2) 0 l2*cos(theta2);
        sin(theta2) cos(theta2) 0 l2*sin(theta2);
        0 0 1 0;
        0 0 0 1];
>> A_23 = [cos(theta3) -sin(theta3) 0 l3*cos(theta3);
        sin(theta3) cos(theta3) 0 l3*sin(theta3);
        0 0 1 0;
        0 0 0 1];
>> A_03 = simplify(A_01 * A_12 * A_23);
>> A_03
```

```
A_03 =

[          cos(theta2 + theta3),          -sin(theta2 + theta3),          0,
l3*cos(theta2 + theta3) + l1*cos(theta1) + l2*cos(theta2)]
```

[sin(theta2 + theta3)*cos(theta1), cos(theta2 + theta3)*cos(theta1), -sin(theta1), l1*sin(theta1) + l2*cos(theta1)*sin(theta2) + l3*cos(theta1)*cos(theta2)*sin(theta3) + l3*cos(theta1)*cos(theta3)*sin(theta2)]

[sin(theta2 + theta3)*sin(theta1), cos(theta2 + theta3)*sin(theta1), cos(theta1), sin(theta1)*(l3*sin(theta2 + theta3) + l2*sin(theta2))]

[0,   0,   0,
1]

4.1.2 Trajectory of the robot

```
syms si(t,qi,qf)
si(t,qi,qf) = qi+((3/4)*t^2-(1/4)*t^3)*(qf-qi);


T_spline = [0 2 4 6 8 10 12];
x_spline = spline(T_spline,[0 p(1,:) 0]);
y_spline = spline(T_spline,[0 p(2,:) 0]);
z_spline = spline(T_spline,[0 p(3,:) 0]);


tt_spline = 0.1:0.1:12;
data_x_spline = ppval(x_spline,tt_spline);
data_y_spline = ppval(y_spline,tt_spline);
data_z_spline = ppval(z_spline,tt_spline);

data_x=[data_x_spline];
data_y=[data_y_spline];
data_z=[data_z_spline];

plot3(data_x,data_y,data_z, 'r');
```

4.1.3 Lagrangian Equation

syms l1 theta1 D1 ...

    l2 theta2 D2 ...

    l3 theta3 D3 ...

>> X = (l3/2)*cos(theta2 + theta3) + l1*cos(theta1) + l2*cos(theta2);

Y = l1*sin(theta1) + l2*cos(theta1)*sin(theta2) + (l3/2)*cos(theta1)*cos(theta2)*sin(theta3) + (l3/2)*cos(theta1)*cos(theta3)*sin(theta2);

Z = sin(theta1)*((l3/2)*sin(theta2 + theta3) + l2*sin(theta2));

dX1 = diff(X,theta1);

dX2 = diff(X,theta2);

dX3 = diff(X,theta3);

dX = dX1+dX2+dX3;

```
dY1 = diff(Y,theta1);

dY2 = diff(Y,theta2);

dY3 = diff(Y,theta3);

dZ1 = diff(Z,theta1);

dZ2 = diff(Z,theta2);

dZ3 = diff(Z,theta3);

dY = dY1+dY2+dY3;

dZ = dZ1+dZ2+dZ3;

dXYZ = dX+dY+dZ;

>> dX = (dX1*D1) + (dX2*D2) + (dX3*D3);

dY = (dY1*D1) + (dY2*D2) + (dY3*D3);

dZ = (dZ1*D1) + (dZ2*D2) + (dZ3*D3);

V2 = (dX^2) + (dY^2) + (dZ^2);

>> V2


V2 =


(D2*sin(theta1)*((l3*cos(theta2 + theta3))/2 + l2*cos(theta2)) +
D1*cos(theta1)*((l3*sin(theta2 + theta3))/2 + l2*sin(theta2)) + (D3*l3*cos(theta2 +
theta3)*sin(theta1))/2)^2 + (D2*((l3*sin(theta2 + theta3))/2 + l2*sin(theta2)) +
(D3*l3*sin(theta2 + theta3))/2 + D1*l1*sin(theta1))^2 + (D2*(l2*cos(theta1)*cos(theta2) +
(l3*cos(theta1)*cos(theta2)*cos(theta3))/2 - (l3*cos(theta1)*sin(theta2)*sin(theta3))/2) -
D1*(l2*sin(theta1)*sin(theta2) - l1*cos(theta1) + (l3*cos(theta2)*sin(theta1)*sin(theta3))/2 +
(l3*cos(theta3)*sin(theta1)*sin(theta2))/2) + D3*((l3*cos(theta1)*cos(theta2)*cos(theta3))/2
- (l3*cos(theta1)*sin(theta2)*sin(theta3))/2))^2


>> K1 = 0.5* ((1/3)*m1*(l1)^2)*(D1)^2;

Unrecognized function or variable 'm1'.


Did you mean:

>> syms m1 m2 m3

>> K1 = 0.5* ((1/3)*m1*(l1)^2)*(D1)^2;

>> K2 = 0.5* ((1/3)*m1*(l1)^2)*((D1)^2+(D2)^2);
>> K3= 0.5*((1/12)*m3*(L3)^2)*(D1^2+(D2+D3)^2);
>> K = K1 + K2 + K3

K =

(D1^2*l1^2*m1)/6 + (l1^2*m1*(D1^2 + D2^2))/6 + (L3^2*m3*((D2 + D3)^2 + D1^2))/24

>> K3 = 0.5*((1/12)*m3*(L3)^2)*(D1^2+(D2+D3)^2) + (0.5* m3 * V2);
>> K

K =

(D1^2*l1^2*m1)/6 + (l1^2*m1*(D1^2 + D2^2))/6 + (L3^2*m3*((D2 + D3)^2 + D1^2))/24

>> g = 9.81;
>> P1 = m1*g*(l1/2);
>> P2 = m2*g*((l1) + (l2/2)*sin(theta2));
>> P3 = m3*g*((L1) + (l2+l3/2)*sin(theta2+theta3));
>> P = P1+P2+P3

P =

(981*m3*(L1 + sin(theta2 + theta3)*(l2 + l3/2)))/100 + (981*l1*m1)/200 + (981*m2*(l1 +
(l2*sin(theta2))/2))/100

>> L = K-P

L =
```

(D1^2*l1^2*m1)/6 - (981*l1*m1)/200 - (981*m2*(l1 + (l2*sin(theta2))/2))/100 - (981*m3*(L1 + sin(theta2 + theta3)*(l2 + l3/2)))/100 + (l1^2*m1*(D1^2 + D2^2))/6 + (L3^2*m3*((D2 + D3)^2 + D1^2))/24

4.2 ROS Development Studios Source Code

4.2.1 xacro file (URDF)

```
<?xml&nbspversion ="1.0"?>

<robot name ="test" xmlns:xacro="http://www.ros.org/wiki/xacro">
<!--base&nbspbox-->
    <link name ="base_link">
  <inertial>
  <mass value ="12.5"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0"/>
  <inertia ixx="0.5208" ixy="0" ixz="0" iyy="1.302" iyz="0" izz="1.302"/>
</inertial>
    <collision>
    <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0" />
      <geometry>
       <box size="1.0&nbsp0.5&nbsp0.5" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0" />
      <geometry>
       <box size="1.0&nbsp0.5&nbsp0.5" />
      </geometry>
     </visual>
    </link>


<!--link1&nbsplefty-->
<link name ="link_01">
<inertial>
  <mass value ="0.236"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.075"/>
  <inertia ixx="0.0010325" ixy="0" ixz="0" iyy="0.0010325" iyz="0" izz="0.0011
8"/>
</inertial>
 <collision>
 <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.1" length="0.15" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.1" length="0.15" />
      </geometry>
     </visual>
    </link>

<!--joint&nbsp0-->
<joint name ="base_link__link_01" type = "revolute">
     <axis xyz="0&nbsp0&nbsp1"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="-0.25&nbsp0&nbsp0.25"/>
     <parent link="base_link"/>
     <child link = "link_01"/>
    </joint>
<!--transmission&nbsp1-->
<transmission name="trans_base_link__link_01">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_link__link_01">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_base_link__link_01">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link1&nbspright-->
<link name ="link_001">
<inertial>
  <mass value ="0.236"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.075"/>
  <inertia ixx="0.0010325" ixy="0" ixz="0" iyy="0.0010325" iyz="0" izz="0.0011
8"/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.1" length="0.15" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
```

```
       <cylinder radius ="0.1" length="0.15" />
      </geometry>
     </visual>
    </link>

<!--joint1&nbspright-->
<joint name ="base_link__link_001" type = "revolute">
     <axis xyz="0&nbsp0&nbsp1"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0.25&nbsp0&nbsp0.25"/>
     <parent link="base_link"/>
     <child link = "link_001"/>
    </joint>
<!--transmission&nbsp1.1-->
<transmission name="trans_base_link__link_001">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="base_link__link_001">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_base_link__link_001">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link&nbsp2&nbsplefty-->
<link name ="link_02">
<inertial>
  <mass value ="0.942"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.3"/>
  <inertia ixx="0.030615" ixy="0" ixz="0" iyy="0.030615" iyz="0" izz="0.00471"
/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </visual>
    </link>
```

```
<!--joint&nbsp2&nbsplefty-->
<joint name ="link_01__link_02" type = "revolute">
     <axis xyz="1&nbsp0&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.15"/>
     <parent link="link_01"/>
     <child link = "link_02"/>
    </joint>
<!--transmission2-->
<transmission name="trans_link_01__link_02">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_01__link_02">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_link_01__link_02">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link&nbsp2&nbspright-->
<link name ="link_002">
<inertial>
  <mass value ="0.942"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.3"/>
  <inertia ixx="0.030615" ixy="0" ixz="0" iyy="0.030615" iyz="0" izz="0.00471"
/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </visual>
    </link>

<!--joint&nbsp2&nbspright-->
<joint name ="link_001__link_002" type = "revolute">
     <axis xyz="1&nbsp0&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.15"/>
     <parent link="link_001"/>
     <child link = "link_002"/>
    </joint>
<!--transmission2.1-->
<transmission name="trans_link_001__link_002">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_001__link_002">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_link_001__link_002">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link&nbsp3&nbspright-->
<link name ="link_003">
<inertial>
  <mass value ="0.942"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.3"/>
  <inertia ixx="0.030615" ixy="0" ixz="0" iyy="0.030615" iyz="0" izz="0.00471"
/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </visual>
    </link>

<!--joint&nbsp3&nbspright-->
<joint name ="link_002__link_003" type = "revolute">
     <axis xyz="1&nbsp0&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.6"/>
     <parent link="link_002"/>
     <child link = "link_003"/>
```

```xml
    </joint>
<!--transmission3-->
<transmission name="trans_link_002__link_003">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_002__link_003">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_link_002__link_003">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link&nbsp3&nbsplefty-->
<link name ="link_03">
<inertial>
  <mass value ="0.942"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.3"/>
  <inertia ixx="0.030615" ixy="0" ixz="0" iyy="0.030615" iyz="0" izz="0.00471"
/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.3" />
      <geometry>
       <cylinder radius ="0.1" length="0.6" />
      </geometry>
     </visual>
    </link>

<!--joint&nbsp3&nbsplefty-->
<joint name ="link_02__link_03" type = "revolute">
     <axis xyz="1&nbsp0&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.6"/>
     <parent link="link_02"/>
     <child link = "link_03"/>
    </joint>


<!--transmission3.1-->
```

```
<transmission name="trans_link_02__link_03">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_02__link_03">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_link_02__link_03">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<!--link ee lefty-->
<link name ="link_eeleft">
<inertial>
  <mass value ="0.00004712"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="-0.03&nbsp0&nbsp0.075"/>
  <inertia ixx="1" ixy="0" ixz="0" iyy="1" iyz="0" izz="1"/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.01" length="0.15" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.01" length="0.15" />
      </geometry>
     </visual>
    </link>

<!--joint ee lefty-->
<joint name ="link_03__link_eeleft" type = "revolute">
     <axis xyz="0&nbsp1&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="-0.03&nbsp0&nbsp0.6"/>
     <parent link="link_03"/>
     <child link = "link_eeleft"/>
    </joint>


<!--transmission4-->
<transmission name="trans_link_03__link_eeleft">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_03__link_eeleft">
```

```xml
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
<actuator name="motor_link_03__link_eeleft">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>


<!--link&nbspee&nbspright-->
<link name ="link_eeright">
<inertial>
  <mass value ="0.00004712"/>
  <origin rpy ="0&nbsp0&nbsp0" xyz ="0&nbsp0&nbsp0.075"/>
  <inertia ixx="1" ixy="0" ixz="0" iyy="1" iyz="0" izz="1"/>
</inertial>
<collision>
<origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.01" length="0.15" />
      </geometry>
     </collision>
     <visual>
      <origin ryp="0&nbsp0&nbsp0" xyz="0&nbsp0&nbsp0.075" />
      <geometry>
       <cylinder radius ="0.01" length="0.15" />
      </geometry>
     </visual>
    </link>

<!--joint&nbspee&nbspright-->
<joint name ="link_03__link_eeright" type = "revolute">
     <axis xyz="0&nbsp1&nbsp0"/>
     <limit effort ="1000.0" lower="-3.14" upper="3.14" velocity="0.5"/>
     <origin rpy="0&nbsp0&nbsp0" xyz="0.03&nbsp0&nbsp0.6"/>
     <parent link="link_03"/>
     <child link = "link_eeright"/>
    </joint>


<!--transmission4.1-->
<transmission name="trans_link_03__link_eeright">
 <type>transmission_interface/SimpleTransmission</type>
  <joint name="link_03__link_eeright">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
</joint>
```

```
<actuator name="motor_link_03__link_eeright">
 <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface
>
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
</transmission>

<gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    </plugin>
</gazebo>

</robot>
```

5. GITHUB LINK

https://github.com/muhdsabri448/EEM343-Dual-Arm-Robotic-Sabri-