RBE-500

Final Assignment Part 1

Group 9: Abhishek Jain, Paurvi Dixit, Surya Murugavel Ravishankar

Objective:

1. Write a node for **forward kinematics** that takes joint positions as input and outputs end effector position and orientation.
2. Write a node for **inverse kinematics** that takes pose of end effector as input and returns joint positions as output.
3. Write a third node that communicates with the 2 nodes made above by:-
    - Taking joint positions from gazebo
    - Sending them to forward kinematics node and taking output(pose) from forward node
    - Supplying it to inverse kinematics node to verify the calculations.

Implementation:

For the purpose of this assignment,

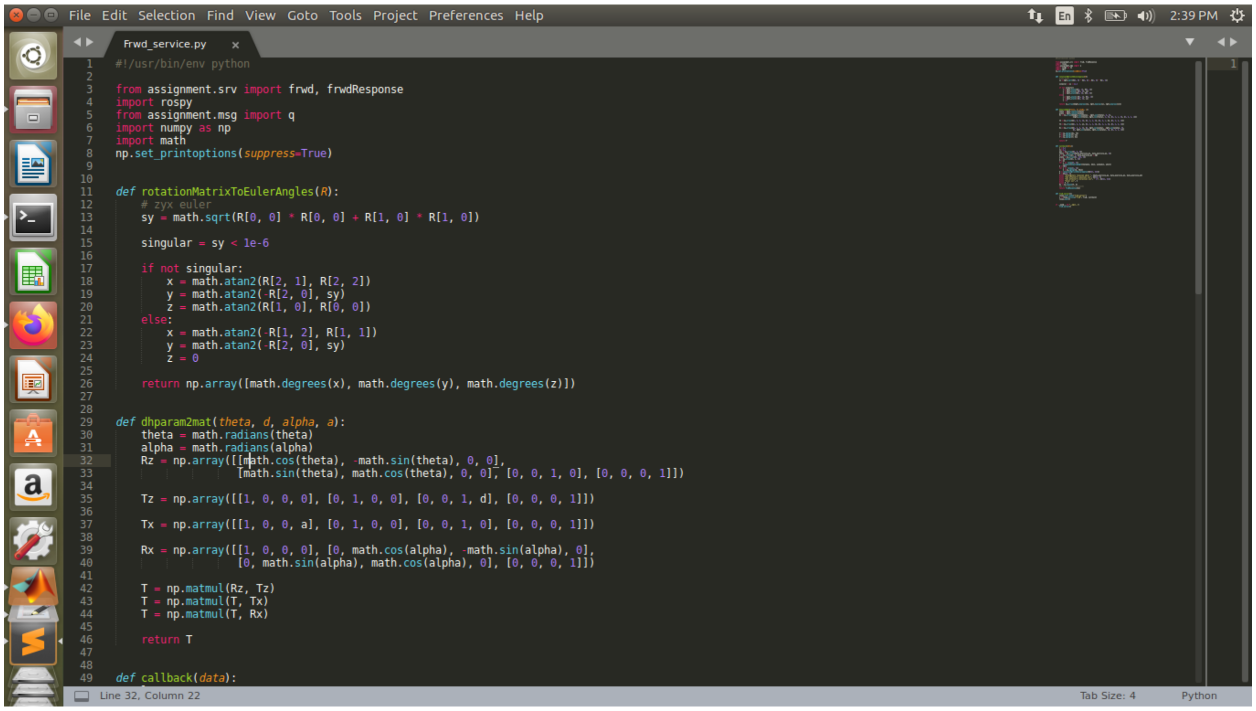
- 2 msg files are declared:
    1. q.msg - 3 variables q1, q2 & q3 with float64 as datatype
    2. array.msg - array T
- 2 service types are being used:
    1. Fwrd.srv - takes qservice (msg type q) as input; gives T(msg type array) as output.
    2. Inv.srv - takes T(msg type array) as input; gives qservice (msg type q) as output.

Que 1:

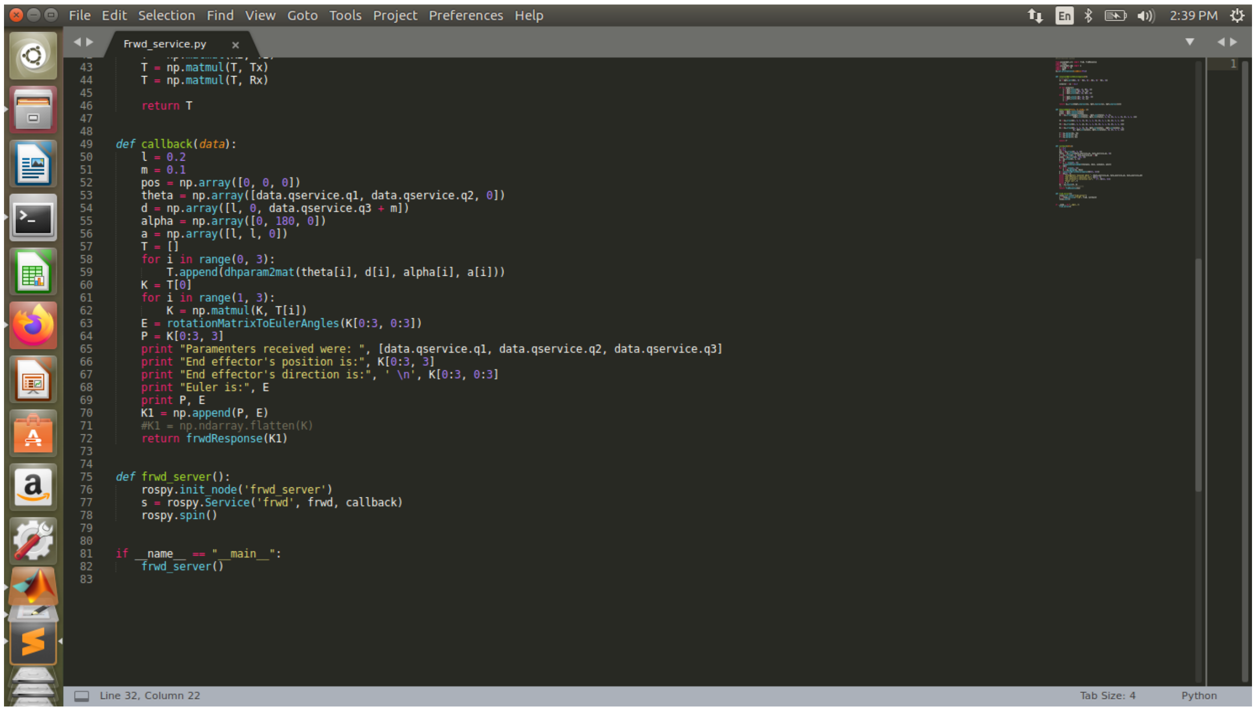
The objective is to write a node that takes joint positions as input and gives end effector pose as output.

- **Frwd_server** provides the service named **'frwd'** which has **'frwd'** as service type. All the requests are passed to callback function. It takes 'data 'as input which has q1, q2, q3.
- It begins with taking joint parameters(in degrees) are inputs and defining dh-parameters.
- To generate transformation matrix, a function named **dhparam2mat()** is called that takes dh-parameters as input.
- Another function, **rotationMatrixToEulerAngles()** is called which returns the value of Euler angles corresponding to the given transformation matrix.
- 6-dimensional vector with the position of the end effector in the cartesian space, and orientation as Euler angles(in degrees) is printed on the terminal as output.

Below is the code snippet of node - **Frwd_service.py**

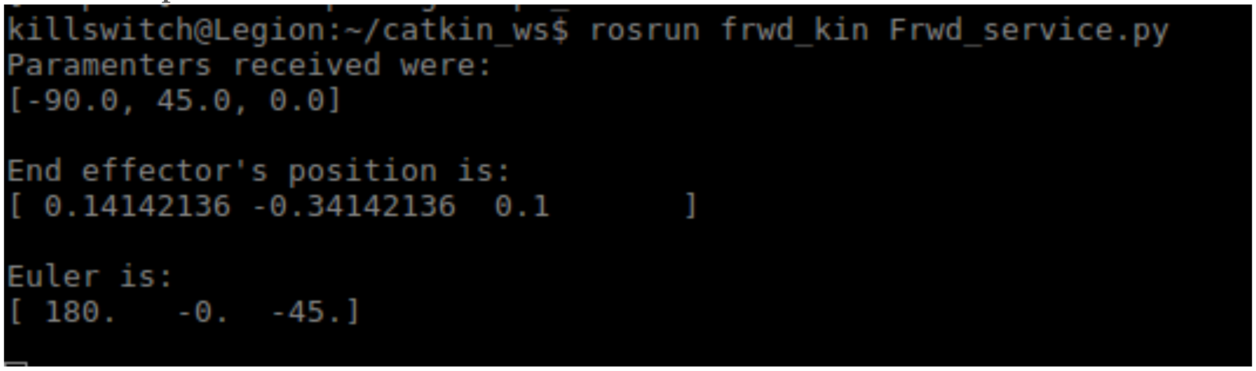

File Edit Selection Find View Goto Tools Project Preferences Help
Frwd_service.py
Line 32, Column 22
Tab Size: 4
Python

```python
#!/usr/bin/env python

from assignment.srv import frwd, frwdResponse
import rospy
from assignment.msg import q
import numpy as np
import math
np.set_printoptions(suppress=True)


def rotationMatrixToEulerAngles(R):
    # zyx euler
    sy = math.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])

    singular = sy < 1e-6

    if not singular:
        x = math.atan2(R[2, 1], R[2, 2])
        y = math.atan2(-R[2, 0], sy)
        z = math.atan2(R[1, 0], R[0, 0])
    else:
        x = math.atan2(-R[1, 2], R[1, 1])
        y = math.atan2(-R[2, 0], sy)
        z = 0

    return np.array([math.degrees(x), math.degrees(y), math.degrees(z)])


def dhparam2mat(theta, d, alpha, a):
    theta = math.radians(theta)
    alpha = math.radians(alpha)
    Rz = np.array([[math.cos(theta), -math.sin(theta), 0, 0],
                   [math.sin(theta), math.cos(theta), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

    Tz = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, d], [0, 0, 0, 1]])

    Tx = np.array([[1, 0, 0, a], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

    Rx = np.array([[1, 0, 0, 0], [0, math.cos(alpha), -math.sin(alpha), 0],
                   [0, math.sin(alpha), math.cos(alpha), 0], [0, 0, 0, 1]])

    T = np.matmul(Rz, Tz)
    T = np.matmul(T, Tx)
    T = np.matmul(T, Rx)

    return T


def callback(data):
```


File Edit Selection Find View Goto Tools Project Preferences Help
Frwd_service.py
Line 32, Column 22
Tab Size: 4
Python

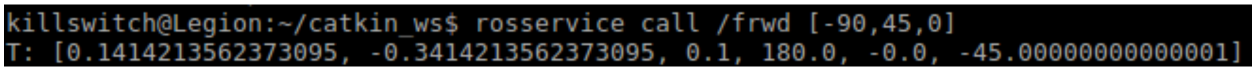
```python
    T = np.matmul(T, Tx)
    T = np.matmul(T, Rx)

    return T


def callback(data):
    l = 0.2
    m = 0.1
    pos = np.array([0, 0, 0])
    theta = np.array([data.qservice.q1, data.qservice.q2, 0])
    d = np.array([l, 0, data.qservice.q3 + m])
    alpha = np.array([0, 180, 0])
    a = np.array([l, l, 0])
    T = []
    for i in range(0, 3):
        T.append(dhparam2mat(theta[i], d[i], alpha[i], a[i]))
    K = T[0]
    for i in range(1, 3):
        K = np.matmul(K, T[i])
    E = rotationMatrixToEulerAngles(K[0:3, 0:3])
    P = K[0:3, 3]
    print "Paramenters received were: ", [data.qservice.q1, data.qservice.q2, data.qservice.q3]
    print "End effector's position is:", K[0:3, 3]
    print "End effector's direction is:", ' \n', K[0:3, 0:3]
    print "Euler is:", E
    print P, E
    K1 = np.append(P, E)
    #K1 = np.ndarray.flatten(K)
    return frwdResponse(K1)


def frwd_server():
    rospy.init_node('frwd_server')
    s = rospy.Service('frwd', frwd, callback)
    rospy.spin()


if __name__ == "__main__":
    frwd_server()
```

Desired output:



```
killswitch@Legion:~/catkin_ws$ rosrun frwd_kin Frwd_service.py
Paramenters received were:
[-90.0, 45.0, 0.0]

End effector's position is:
[ 0.14142136 -0.34142136   0.1        ]

Euler is:
[ 180.    -0.   -45.]
```

The output is in the form of a 6x1 matrix:T as follows:-



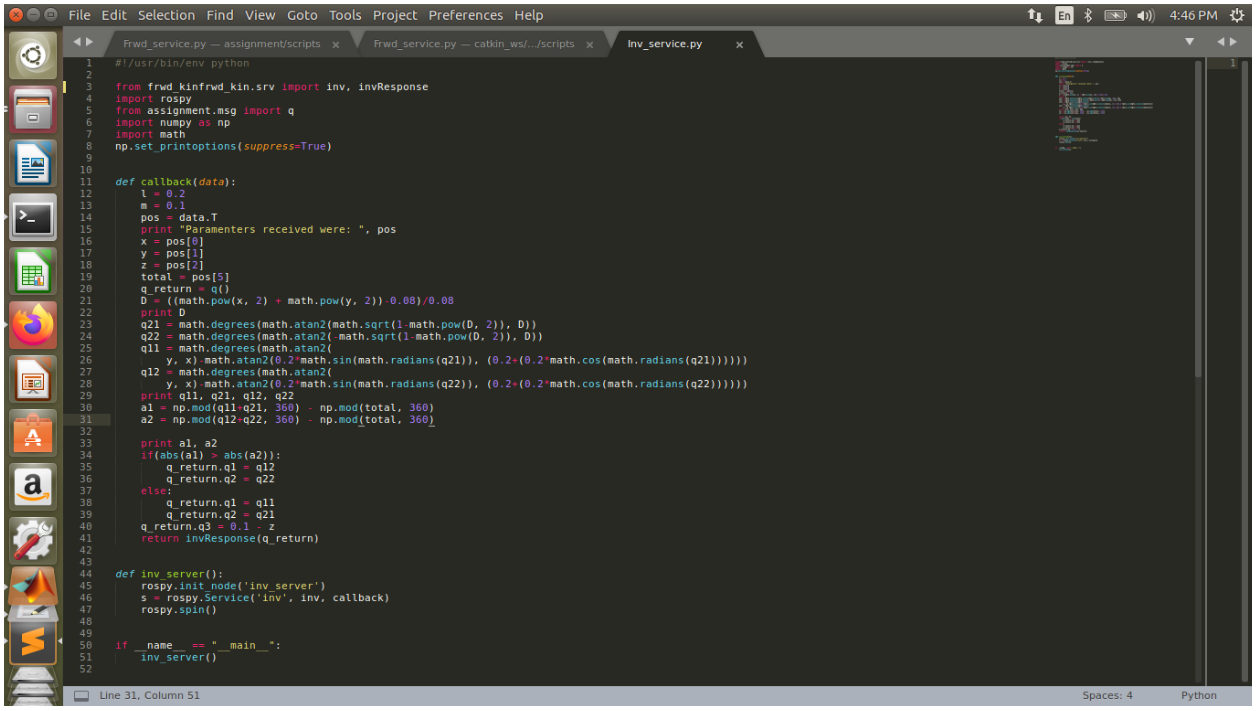
```
killswitch@Legion:~/catkin_ws$ rosservice call /frwd [-90,45,0]
T: [0.1414213562373095, -0.3414213562373095, 0.1, 180.0, -0.0, -45.00000000000001]
```

Que 2:

The objective is to write a node that takes pose of end effector and return joint positions.

- File named '**inv_service.py** is the inverse kinematics node.
- Server '**inv_server**' provides the service named '**inv**' which has '**inv**' as service type.
- Callback function takes pose of the end effector as an array T. Values of q1, q2, q3 are determined geometrically.
- Considering all the possible cases for q1, q2 and applying validations, q1, q2, q3 are determined and returned by the function.
- Joint positions(in 'radians' and 'm') as output are printed on the terminal.

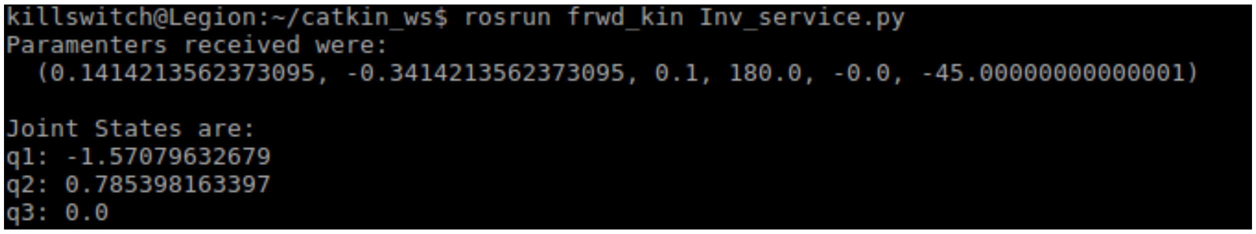
Below is the code snippet of **inv_service.py**



```python
#!/usr/bin/env python

from frwd_kinfrwd_kin.srv import inv, invResponse
import rospy
from assignment.msg import q
import numpy as np
import math
np.set_printoptions(suppress=True)


def callback(data):
    l = 0.2
    m = 0.1
    pos = data.T
    print "Paramenters received were: ", pos
    x = pos[0]
    y = pos[1]
    z = pos[2]
    total = pos[5]
    q_return = q()
    D = ((math.pow(x, 2) + math.pow(y, 2))-0.08)/0.08
    print D
    q21 = math.degrees(math.atan2(math.sqrt(1-math.pow(D, 2)), D))
    q22 = math.degrees(math.atan2(-math.sqrt(1-math.pow(D, 2)), D))
    q11 = math.degrees(math.atan2(
        y, x)-math.atan2(0.2*math.sin(math.radians(q21)), (0.2+(0.2*math.cos(math.radians(q21))))))
    q12 = math.degrees(math.atan2(
        y, x)-math.atan2(0.2*math.sin(math.radians(q22)), (0.2+(0.2*math.cos(math.radians(q22))))))
    print q11, q21, q12, q22
    a1 = np.mod(q11+q21, 360) - np.mod(total, 360)
    a2 = np.mod(q12+q22, 360) - np.mod(total, 360)

    print a1, a2
    if(abs(a1) > abs(a2)):
        q_return.q1 = q12
        q_return.q2 = q22
    else:
        q_return.q1 = q11
        q_return.q2 = q21
    q_return.q3 = 0.1 - z
    return invResponse(q_return)


def inv_server():
    rospy.init_node('inv_server')
    s = rospy.Service('inv', inv, callback)
    rospy.spin()


if __name__ == "__main__":
    inv_server()
```

File Edit Selection Find View Goto Tools Project Preferences Help
4:46 PM
Frwd_service.py — assignment/scripts
Frwd_service.py — catkin_ws/.../scripts
Inv_service.py
Line 31, Column 51
Spaces: 4
Python

Output:

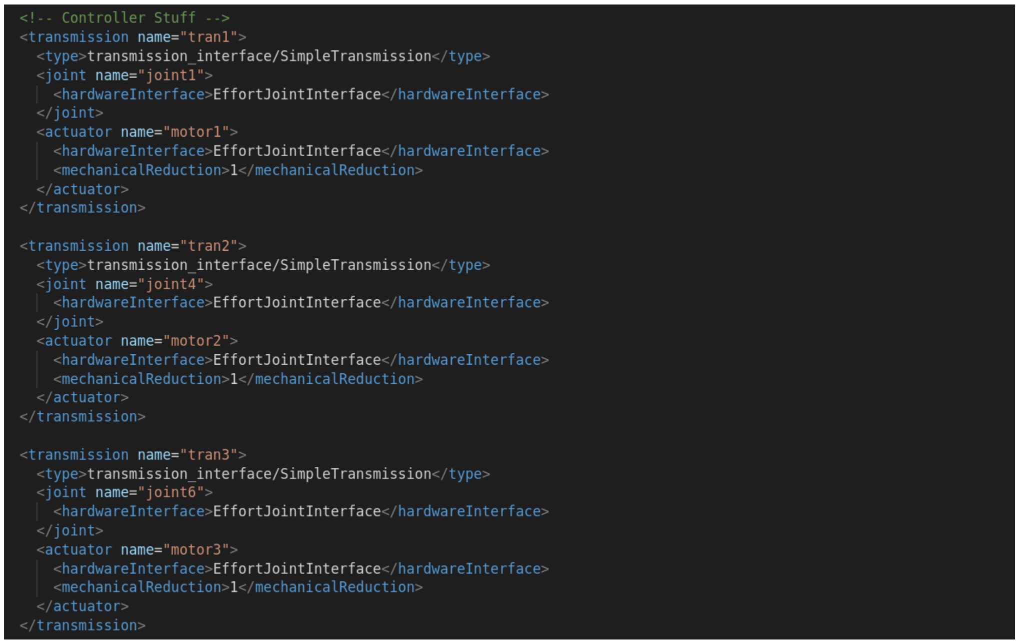


```
killswitch@Legion:~/catkin_ws$ rosrun frwd_kin Inv_service.py
Paramenters received were:
  (0.1414213562373095, -0.3414213562373095, 0.1, 180.0, -0.0, -45.00000000000001)

Joint States are:
q1: -1.57079632679
q2: 0.785398163397
q3: 0.0
```

Que 3:

The objective is to read joint positions from Gazebo and interact with the other two nodes.

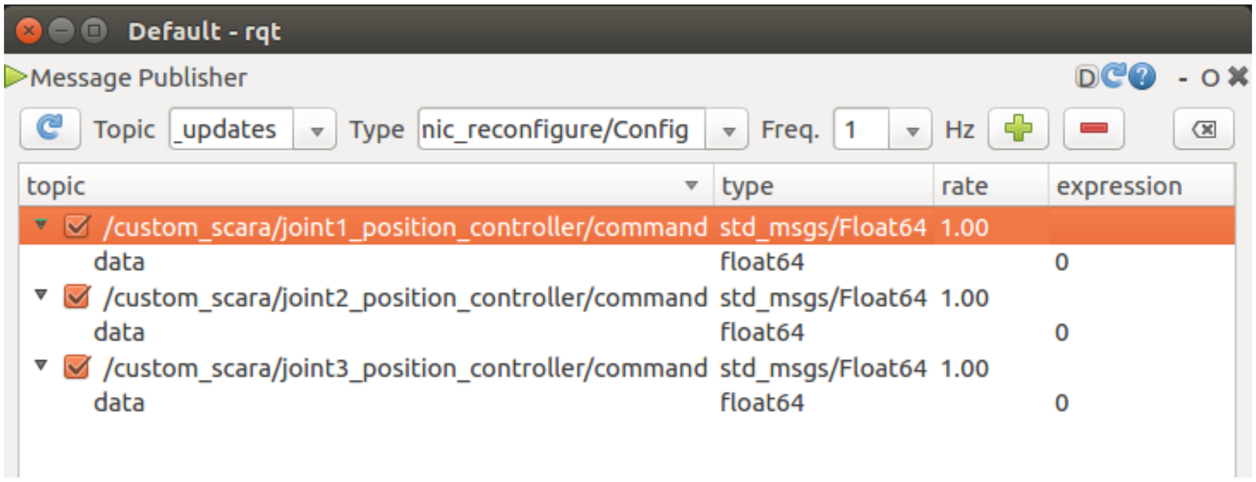
- For the same, joint controllers are defined in the custom_scara.urdf file in the form of transmission elements.
- A new package: custom_robot_control which contains config: custom_robot_control.yaml and launch: custom_robot_control.launch files is defined.
- They are used to define the controller configuration and launch the controllers respectively.
- A package named: **robot_state_publisher** is used and accessed in the launch file to publish the robot joint states.
- A node **'assignment_1'** is created to interact with the other 2 node: frwd_server and inv_server.
- The node subscribes to the topic: **"/custom_scare/joint_states"** for receiving the joint states from Gazebo. (robot_state_publisher is publishing the joint states to the same topic.)
- It calls the callback function which performs the following tasks:
  - Prints the received joint positions to the console.
  - Requests the **frwd_server** node to perform the required forward kinematics.
  - Prints the response from the forward kinematics node and inputs it to the inverse kinematics node.
  - Requests the inv_server node to perform the required following inverse kinematics.
  - Prints the response from the inverse kinematics



```
<!-- Controller Stuff -->
<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint1">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor1">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran2">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint4">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor2">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran3">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint6">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor3">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

The results obtained for 3 separate joint positions are as follows:

Case1:



```
Topic _updates   Type nic_reconfigure/Config   Freq. 1   Hz

topic                                          type            rate   expression
/custom_scara/joint1_position_controller/command  std_msgs/Float64  1.00
    data                                       float64                0
/custom_scara/joint2_position_controller/command  std_msgs/Float64  1.00
    data                                       float64                0
/custom_scara/joint3_position_controller/command  std_msgs/Float64  1.00
    data                                       float64                0
```

Fig: Custom Joint Positions: 0, 0, 0

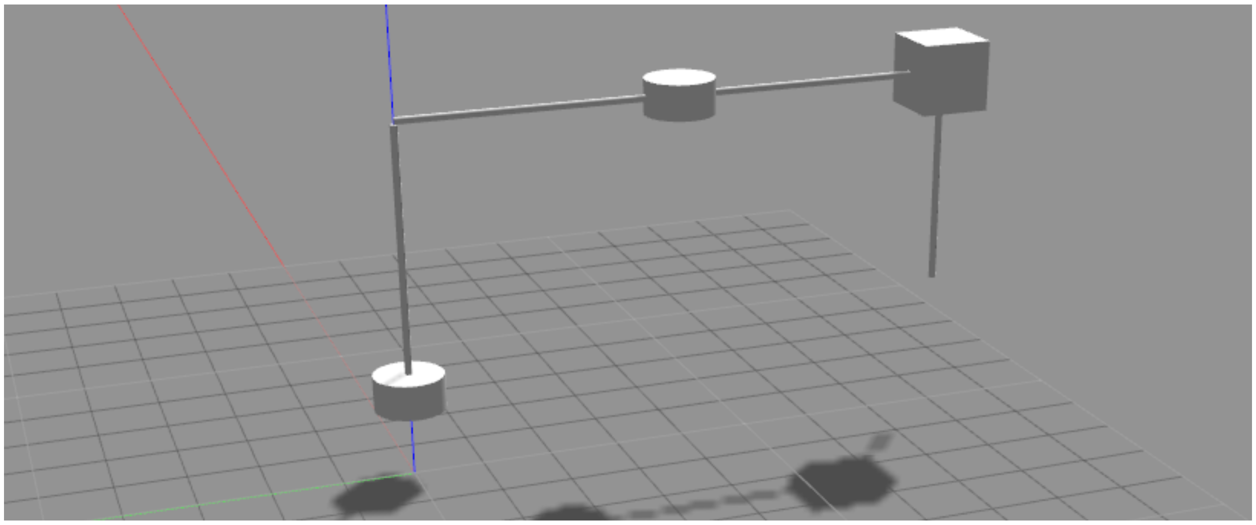
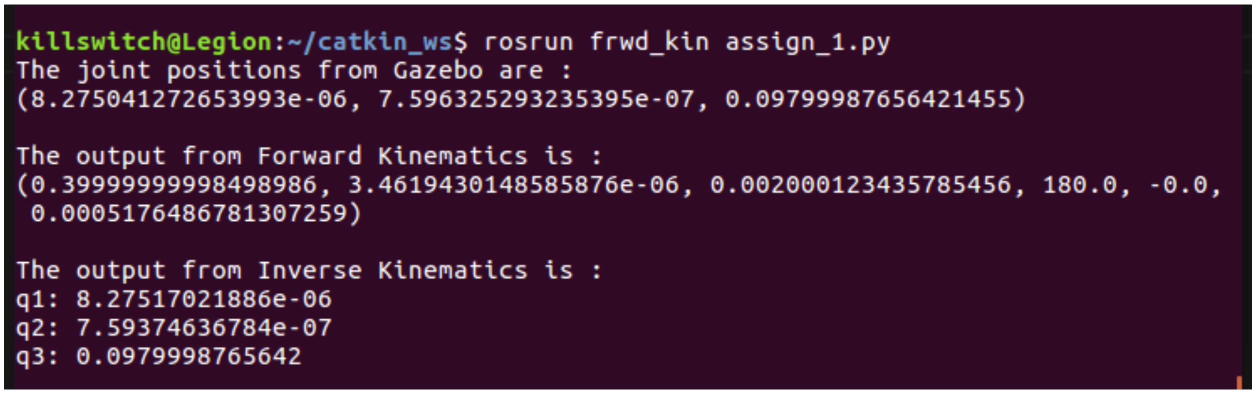


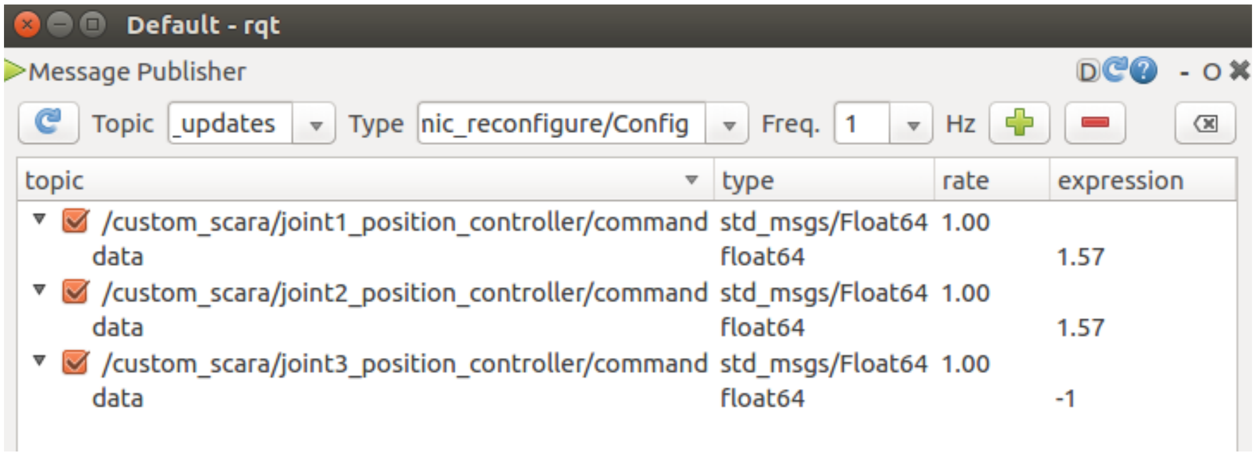
Fig: Gazebo



```
killswitch@Legion:~/catkin_ws$ rosrun frwd_kin assign_1.py
The joint positions from Gazebo are :
(8.275041272653993e-06, 7.596325293235395e-07, 0.09799987656421455)

The output from Forward Kinematics is :
(0.39999999998498986, 3.4619430148585876e-06, 0.002000123435785456, 180.0, -0.0,
 0.0005176486781307259)

The output from Inverse Kinematics is :
q1: 8.27517021886e-06
q2: 7.59374636784e-07
q3: 0.0979998765642
```

Fig: Console Output

Case2:



topic	type	rate	expression
/custom_scara/joint1_position_controller/command	std_msgs/Float64	1.00	
data	float64		1.57
/custom_scara/joint2_position_controller/command	std_msgs/Float64	1.00	
data	float64		1.57
/custom_scara/joint3_position_controller/command	std_msgs/Float64	1.00	
data	float64		-1

Fig: Custom Joint Positions: 1.57, 1.57, -1

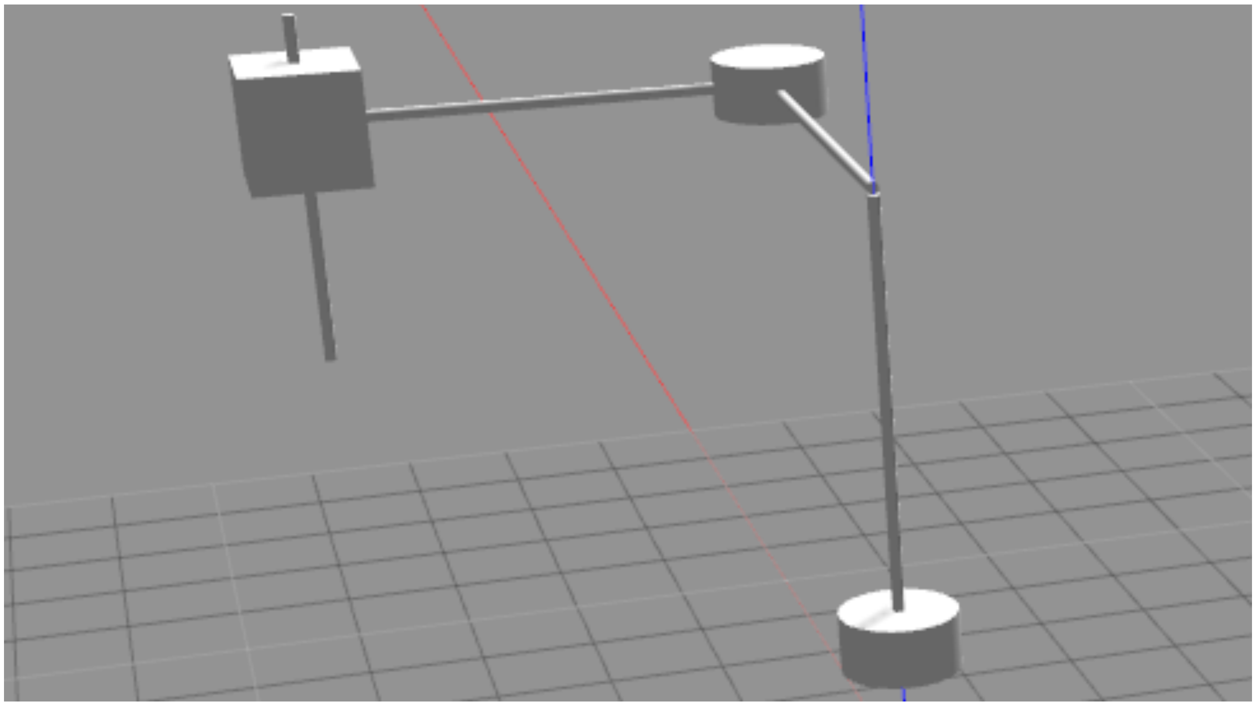
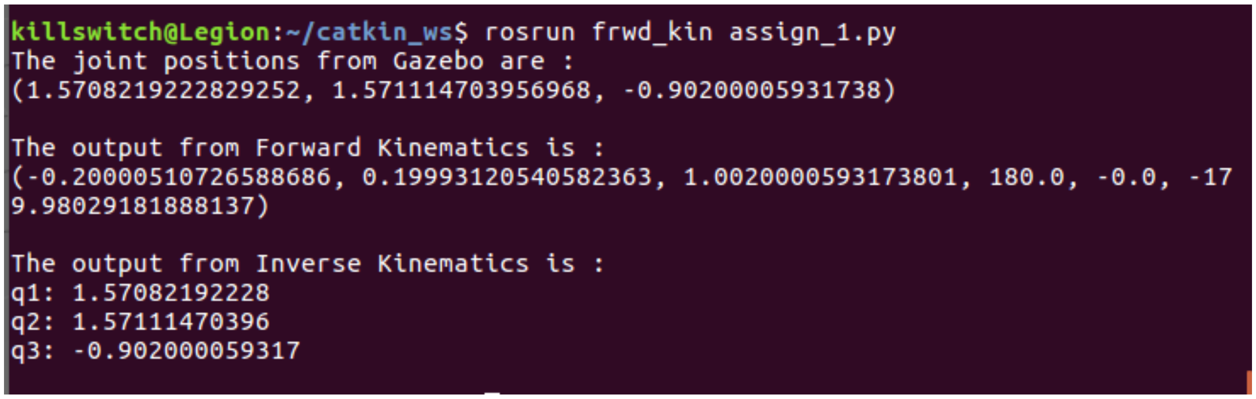


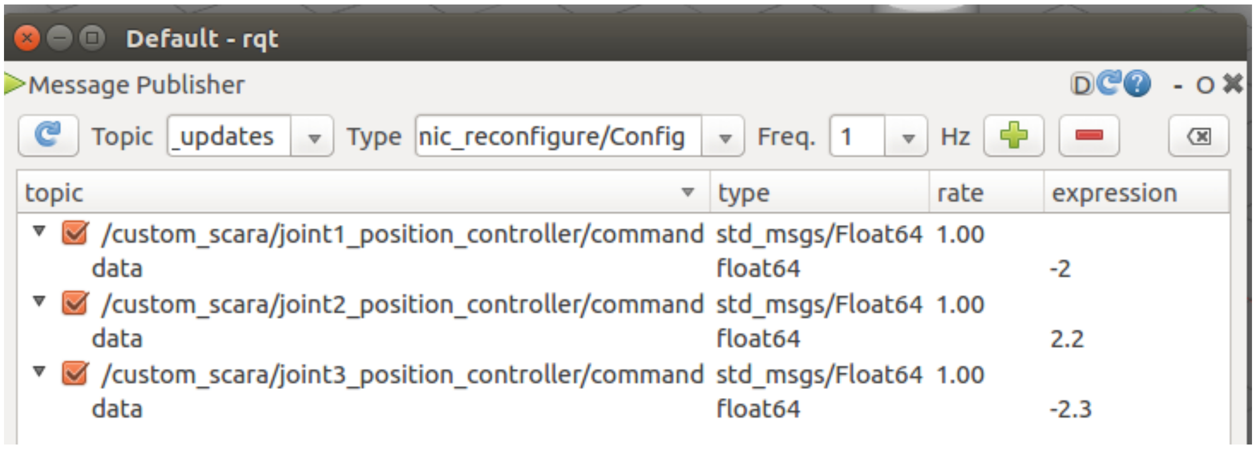
Fig: Gazebo



```
killswitch@Legion:~/catkin_ws$ rosrun frwd_kin assign_1.py
The joint positions from Gazebo are :
(1.5708219222829252, 1.571114703956968, -0.90200005931738)

The output from Forward Kinematics is :
(-0.20000510726588686, 0.19993120540582363, 1.0020000593173801, 180.0, -0.0, -17
9.98029181888137)

The output from Inverse Kinematics is :
q1: 1.57082192228
q2: 1.57111470396
q3: -0.902000059317
```

Fig: Console Output

Case 3:



topic	type	rate	expression
/custom_scara/joint1_position_controller/command	std_msgs/Float64	1.00	
data	float64		-2
/custom_scara/joint2_position_controller/command	std_msgs/Float64	1.00	
data	float64		2.2
/custom_scara/joint3_position_controller/command	std_msgs/Float64	1.00	
data	float64		-2.3

Fig: Custom Joint Positions: -2, 2.2, -2.3

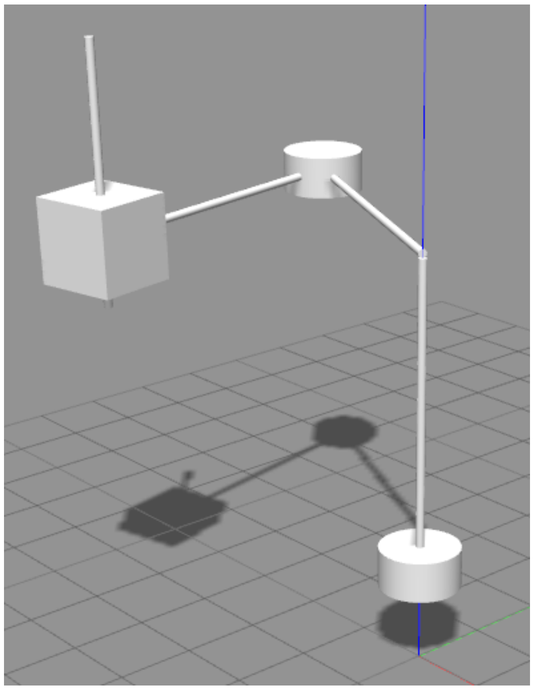
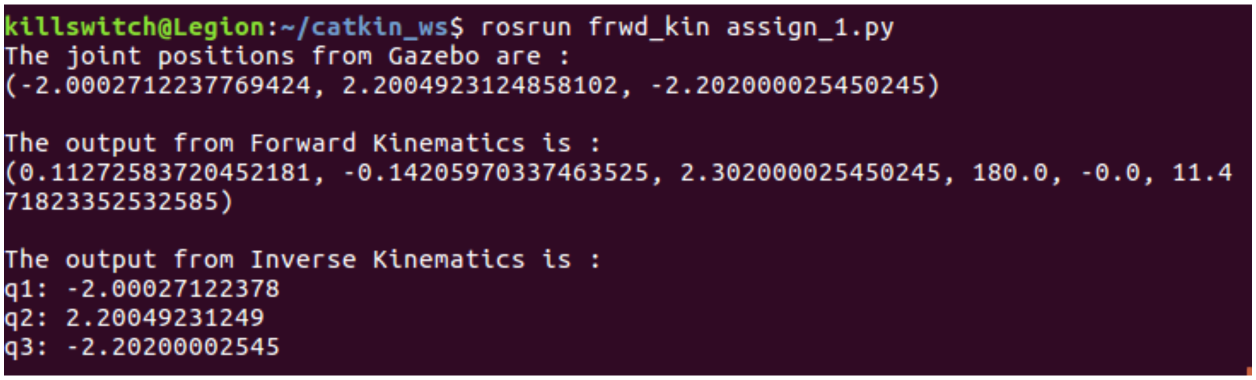


Fig: Gazebo



```
killswitch@Legion:~/catkin_ws$ rosrun frwd_kin assign_1.py
The joint positions from Gazebo are :
(-2.0002712237769424, 2.2004923124858102, -2.202000025450245)

The output from Forward Kinematics is :
(0.11272583720452181, -0.14205970337463525, 2.302000025450245, 180.0, -0.0, 11.4
71823352532585)

The output from Inverse Kinematics is :
q1: -2.00027122378
q2: 2.20049231249
q3: -2.20200002545
```

Fig: Console Output