# RBE 500 : FINAL ASSIGNMENT - PART 1 REPORT

Team 7 : Neel Dhanaraj, Sabhari Natarajan, Sidharth Sharma

Three python scripts were developed (one for each node), for the implementation of the assignment. Following is a summary of each of the scripts:

1) Forward Kinematics : We have a node "SCARA_T7_FK_Server" which has a service "SCARA_T7_FK". This service receives the joint state (q1,q2,q3) and returns the end effector pose ($x,y,z,\theta_{z0},\theta_{y},\theta_{z1}$). It uses the DH parameters and transformation matrices for this calculation. The Euler angles used here is Z-Y-Z.

2) Inverse Kinematics : We have a node "SCARA_T7_IK_Server" which has a service "SCARA_T7_IK". This service receives the end effector pose ($x,y,z,\theta_{z0},\theta_{y},\theta_{z1}$) and returns the joint state (q1,q2,q3). By using x,y,z we get two sets of (q1,q2,q3). To choose the correct one, $\theta_{z0},\theta_{y},\theta_{z1}$ is used. We compare the angle between "end-effector frame x-axis" and "base frame x-axis" w.r.t "base frame z-axis". With this we get a condition "$180° + \theta_{z0} - \theta_{z1} = q1 + q2$". Both sides of the equation were normalized to be between 0° and 360° before comparing. The set of q1,q2,q3 for which this equation was satisfied was used as the solution.

3) Read from Gazebo and Check : Based on the SCARA related files (config.yaml, .urdf etc.) which was defined, Gazebo publishes the joint states of SCARA to a topic named "SCARA_T7/joint_states". We create a node "SCARA_T7_Check", which subscribes to this topic. Let's call this set of data as Q1. Now, a service call is made to "SCARA_T7_FK" with input as Q1, which returns the Pose P1. Further, another service call is made to "SCARA_T7_IK" with input as P1, which returns the Joint States Q2. Finally, this Q1 and Q2 are compared, and these should match exactly if our IK and FK calculations were correct.

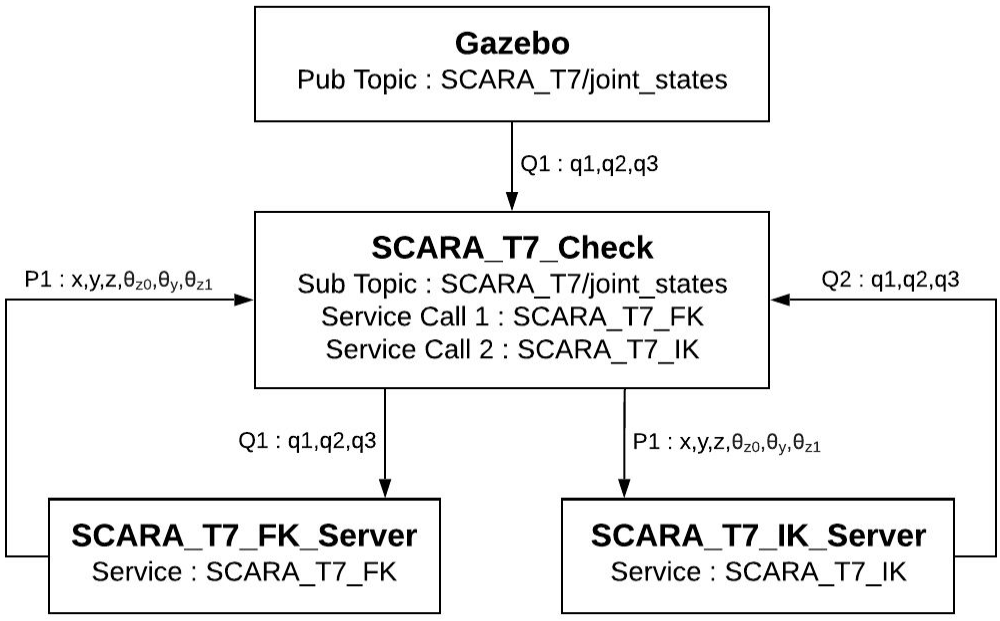
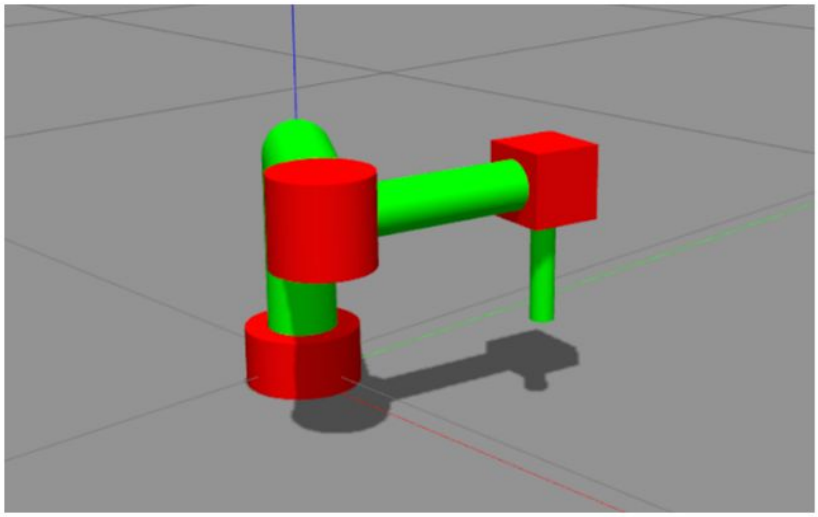



*Figure 1. Flowchart showing the various nodes and information flow.*



To run random test cases, the robot was spawned in gazebo and using rqt's "Message Publisher" plugin, the Joint values for Joint 1, Joint 2 & Joint 3 were published to follow a sinusoidal pattern. Figure 2 shows the SCARA robot at a random instant.

The python script having the node "SCARA_T7_Check" was run at different time instances to run the test cases. Figure 3 shows the results for 3 different test cases.

*Figure 2. Screenshot of SCARA robot in Gazebo*

```
sabhari@sabhari-G3-3590: ~
sabhari@sabhari-G3-3590:~$ rosrun final_project A3_Read_Check.py

Waiting for the required services...
Required services up and running...
Starting subscriber
Joint Positions Read (Q1) = [-86.83 -39.22   0.02]
Calculated Pose (P1) = [  -0.11   -0.36    0.08    0.    180.    -53.94]
Calculated Joint Position (Q2) = [-86.83 -39.22   0.02]
Test successful : Q1 and Q2 match

sabhari@sabhari-G3-3590:~$ rosrun final_project A3_Read_Check.py

Waiting for the required services...
Required services up and running...
Starting subscriber
Joint Positions Read (Q1) = [ 89.68  63.53  -0.03]
Calculated Pose (P1) = [  -0.18    0.29    0.13    0.    180.     26.79]
Calculated Joint Position (Q2) = [ 89.68  63.53  -0.03]
Test successful : Q1 and Q2 match

sabhari@sabhari-G3-3590:~$ rosrun final_project A3_Read_Check.py

Waiting for the required services...
Required services up and running...
Starting subscriber
Joint Positions Read (Q1) = [ 16.4  -80.25   0.01]
Calculated Pose (P1) = [   0.28   -0.12    0.09    0.    180.   -116.15]
Calculated Joint Position (Q2) = [ 16.4  -80.25   0.01]
Test successful : Q1 and Q2 match

sabhari@sabhari-G3-3590:~$
```

*Figure 3. Results for the 3 test cases*

As we can see in Figure 3, the test has passed for the three random test case and hence the FK & IK codes are working as required.

Attachments : final_project.zip

1) Scripts - A1_FK_Server.py, A2_IK_Server.py , A3_Read_Check.py
2) Robots - A_SCARA.urdf.xacro
3) Config - A_SCARA_config.yaml
4) Launch - A_Load_SCARA.launch
5) Srv - FK_srv.srv, IK_srv.srv