# ECS653U/ECS7004P Advanced Robotic Systems (Robotics III)
## WEEK 8 (ASSESSED LAB - 15%) [Lab Report]

The objective of this Lab Report (15% of your final exam mark) is to create a ROS package that will automatically generate Cartesian space movements of the end-effector of the Panda robot manipulator: the end-effector will have to "draw" squares of different sizes on the x-y Cartesian plane, starting from a given robot configuration.
Generic users should be able to install the package (i.e. download the .zip folder of the package, unzip it on their computer within their catkin workspace and compile it), and after having installed the moveit_tutorials and panda_moveit_config as well, they should be able to run the following on 4 different terminals:
1) roslaunch panda_moveit_config demo.launch
2) rosrun AR_week8_test square_size_generator.py
3) rosrun AR_week8_test move_panda_square.py
4) rosrun rqt_plot rqt_plot
and they should see on their screen something similar to the video available on QM+ (WEEK 8: ASSESSED LAB (15%) - VIDEO OF DESIRED OUTPUT).

Detailed instructions follow. **[Total mark is 100 points]**

### Part I --- The Package --- [5 points]

***One ROS Package*** named ***AR_week8_test*** should be created, within the source folder of your catkin workspace (which should be something like $HOME/catkin_ws/src).

The AR_week8_test package should depend on the following ROS packages:
rospy
moveit_commander
moveit_msgs
std_msgs

One additional folder named "src" must be created under the AR_week8_test folder.
You should save your .py code files in the src folder.

### Part II --- The Environment --- [5 points]

**Two repositories** should be installed on your catkin workspace:

1) git clone -b melodic-devel https://github.com/ros-planning/moveit_tutorials.git
2) git clone -b melodic-devel https://github.com/ros-planning/panda_moveit_config.git

(see details here: http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/getting_started/getting_started.html )

This will allow to start rViz with the Panda robot model by running:
roslaunch panda_moveit_config demo.launch
(see details here:
http://docs.ros.org/melodic/api/moveit_tutorials/html/doc/quickstart_in_rviz/quickstart_in_rviz_tutorial.html )

**Part III --- The Nodes --- [25 points]**

***Two ROS nodes*** will need to be created:

***1) square_size_generator.py [5 points]*** This node should generate a random value (e.g. you can use the Python random.uniform() function for that) for the size of the square (i.e. the length of the side of the square), every 20 seconds, and publish them on a ROS Topic using an appropriate message. The length of the side of the square should be a random real number (i.e. float) between 0.05 and 0.20.

***2) move_panda_square.py [20 points]*** This node should subscribe to the ROS Topic created by Node 1, wait for messages (which will include the desired length of the side of the square), and when a new message is received it should do the following:
**2.a) move the Panda robot to a starting configuration**, defined in joints space as follows: start_conf = [0, -π/4, 0, -π/2, 0, π/3, 0];
**2.b) plan a Cartesian path** that will realize the desired motion of the robot end-effector (i.e. a square of the desired size on the x-y Cartesian plane) – this step might include, or not, a visualization of the planned trajectory on rViz;
**2.c) show the planned trajectory** on rViz, without executing it;
**2.d) execute the planned trajectory** (on rViz);
**2.e) wait for the next message** (about a new desired length of the side of the square).

**Part IV --- Plotting data --- [5 points]**

The rqt_plot should continuosly plot the positions of the joints of the Panda arm (as shown in the "reference" video); you can configure this manually (i.e. after launching rqt_plot).

**Part V --- Recording a video of the output of your software --- [50 points]**

You should run the following, each on a different terminal:
1) roslaunch panda_moveit_config demo.launch
2) rosrun AR_week8_test square_size_generator.py
3) rosrun AR_week8_test move_panda_square.py
4) rosrun rqt_plot rqt_plot
and record a mp4 video of your screen while everything is running (e.g. using Kazam). Your video should be about 90 seconds long. The content of the recorded video should look similar to the content of the "reference" video available on QM+ (WEEK 8: ASSESSED LAB (15%) - VIDEO OF DESIRED OUTPUT).

**Part VI --- QM+ submission --- [10 points]**

You should submit a single .zip folder named *YourStudentID*_AR_week8_test, which includes:
- the AR_week8_test package folder;
- your recorded video (.mp4);
- a README.txt file explaining the main steps needed to compile and use the package.

Remember to add meaningful comments (and your name) to the code.