ME5493: Robotics, HW 8
Homework due on 12-10-2017, Topics: Introduction to ROS

Instructions: You will submit this HW through `https://github.com/` (a free code hosting website). Please make an account and create the folder hw8 in your account. Within this folder make subfolders for each question. Submit all your code and other text/figure files as indicated in the questions below.

1. Use the folder *02.OKane_chapter2* to upload your work. Run the turtlesim in teleoperation mode (see Page 16 of OKane's book). Teleoperate the turtle to your initials. It does not have to be a perfect drawing. Create a screenshot of your figure window.

2. Use the folder *04.hello_world* for this exercise. Create a new file called *my_name.cpp*. This file should print out your name when run through ROS. *Hint:* You will have to make changes to *package.xml*, *CMakelists.txt*, then build, source and execute the file. **Upload your work in the folder** ***04.hello_world*** **folder and include a screenshot of the terminal window after executing the file** *my_name.cpp*

3. Use the folder *05.pub_sub* for this exercise.

   - Modify the *pubvel.cpp* file to get the turtle to do maneuver to type the letter 'L' and come to a stop. (Hint: You will have to find a way to keep a track of time, either using a timed loop (search keeping track of time in C) or initializing and incrementing a counter. **Upload a screenshot of the turtle, call it turtle_L**.
   - Modify the *example2.launch* file to call *turtle_teleop_key* (tele-operating the turtle, see Chapter 2) instead of *pubvel* (randomized motion). Now teleoperate the turtle so that it teaches the corners of the window and note the x,y position of the corners. **Upload the new launch file (call it teleop.launch) and the x,y coordinates of the four corners in the file corner.txt file**. (Hint: Besides setting the correct type, pkg in the launch file, you will also have to set the turtle to be teleoperated via the terminal. This can be done by typing the setting (launch-prefix="xterm-e"), remove the brackets (see page 90 in OKane's book).
   - Write a new file *pubvel_smart.cpp* that randomly navigates the window but never hits the edges of the window. **Upload a screenshot of the turtle after it runs for long enough time, call it** *turtle_smart.png or .jpg.*

4. Use the folder *06.OKane_chapter7* to upload your work. Create the file *pubvel_spiral* that moves the turtle in an outward spiral. To do this, keep the angular speed constant ($\omega = C$) but increase the linear speed as a linear function of time ($v = Ct$) where C is a constant. (Hint: You can use the code you developed for 1st question in *05.pub_sub*). Create a launch file, *spiral_orange.launch* that changes the background color of the turtlesim window to UTSA orange (R = 241, G = 90, B = 34) and is able to change C to a user specified value. Adjust the value of the constant C such that the turtle draws not more than 5 spiral. (Hint: Look at the launch file *fast_yellow.launch* on page 115). **Upload the files:** *pubvel_spiral*, *spiral_orange.launch*, **and a screenshot of the turtlesim, call it** *spiral.png or .jpg*..

5. Use the folder *07.OKane_chapter8* to upload your work. Create the file *pubvel_discrete_motion* that has two service calls: (1) move_linear should move the turtle 1 unit straight and (2) move_angle should move the turtle 90 degrees counterclockwise. (Hint: You can modify the code *pubvel_toggle.cpp*). The client program is still the *spawn_turtle* that you used in this chapter. Now use the server/client program to get the turtle to draw a rectangle of length of 2 units and height of 1 unit (Hint: See the terminal commands on page 130 in the book). **Upload the files:** *pubvel_discrete_motion* **and a screenshot of the turtlesim, call it** *rectangle.png or .jpg.*