**ASSIGNMENT 3 - ROS Turtlesim**

Nathan Ramos & Quinn Frady

**Description of the problem**

The goal of this assignment is to develop a ros package to integrate turtlesim (an external package) with our own user-generated package. We will use this integration to perform the following tasks.

**Task 1**

The goal of task 1 was to develop our own nodes dependent on the ros package turtlesim that subscribes to a user created command velocity publisher. Then, our node will publish commands to turtlesim to get it to drive in a square whilst publishing the turtles pose.

Task 1 wanted the following ros nodes and topics:

* Nodes
  + Command\_pub
    - Topics
      * Publishes - /cmd\_vel of type geometry\_msgs::Twist
  + Turtle\_motion
    - Topics
      * Subscribes - /cmd\_vel
      * Publishes - /tintin/cmd\_vel of type geometry\_msgs::Twist
      * Subscribes - /tintin/pose
      * Publishes - /pose\_tt of type geometry\_msgs::Pose2D

To execute this assignment, we created our package da3\_1 in the workspace da3 that references std\_msgs, geometry\_msgs, turtlesim, and roscpp for its dependencies. Afterward, we created the launch file, begin.launch which initiates plotjuggler, command\_pub, turtle\_motion, and turtlesim while killing turtle1 and spawning tintin. Next, we created our command\_pub node through the file pub\_cmd\_vel.cpp. This node simply publishes a randomly generated linear velocity and desired distance to the topic /cmd\_vel of type geometry\_msgs::Twist. Finally, we created our turtle\_motion node through the file turtle\_motion.cpp. This node simply subscribes to the aforementioned cmd\_vel topic. It uses the subscribed data to drive the turtle tintin in a square via /tintin/cmd\_vel of type geometry\_msgs::Twist. Then, it reads the turtles pose via /tintin/pose and publishes that for further use through the node /pose\_tt of type geometry\_msgs::Pose2D. The square algorithm is quite simple, based upon the given velocity and distance we created a timer that runs on the desired time the turtle should be driving. Once the timer stops, the turtle turns 90 degrees and repeats for all 4 sides of the square.

To execute this task you must run the following steps

1. Catkin\_make
2. roscore
3. roslaunch da3\_1 begin.launch
4. View the data in turtlesim and/or plotjuggler

If you’d like to re-run the task with different velocities and distances, you must ctrl+c the launch file and re-run it.

**Task 1 Source Code Directory**

DesignAssignments/da3/src/da3\_1/src/

**Task 1 Video**

<https://www.youtube.com/watch?v=in_z-T82r3g&list=PLZTXnWnnMe9eMQkYrS3KXLsXxmp48Bh74&index=4>

**Task 2**

The goal of task 2 was to develop our own nodes dependent on the ros package turtlesim that subscribes to a user created pose publisher. Then, our node will publish commands to turtlesim to get it do drive to that pose and match its theta whilst publishing the turtles pose and velocity.

Task 2 wanted the following ros nodes and topics

* Nodes
  + Turtle\_goto\_pose
    - Topics
      * Subscribes - /goto\_pose
      * Publishes - /tintin/cmd\_vel of type geometry\_msgs::Twist
      * Subscribes - /tintin/pose
      * Publishes - /pose\_tt of type geometry\_msgs::Pose2D
      * Subscribes - /tintin/cmd\_vel
      * Published - /cmd\_vel\_tt of type geometry\_msgs::Twist
  + Command\_pose
    - Topics
      * Publishes - /goto\_pose of type geometry\_msgs::Pose2D

To execute this assignment, we created our package da3\_2 in the workspace da3 that referenced std\_msgs, geometry\_msgs, turtlesim, and roscpp for its dependencies. Afterward, we created the launch file, begin.launch which initiates command\_pub, turtle\_motion, and turtlesim while killing turtle1 and spawning tintin. Note that the launch file is capable of starting plotjuggler, but we commented this out to make it easier to record the submission video. Next, we created our command\_pose node through the file pub\_goto\_pose.cpp. This node simply published a randomly generated x, y, and theta to the topic /goto\_pose of type geometry\_msgs::Pose2D. Finally, we created our turtle\_goto\_pose node through the file turtle\_goto\_pose.cpp. This node simply subscribes to the aforementioned goto\_pose topic. It uses the subscribed data to drive the turtle tintin towards the pose via /tintin/cmd\_vel of type geometry\_msgs::Twist. Then, it reads the turtles pose and cmd\_vel via /tintin/pose and /tintin/cmd\_vel and publishes that for further use through the nodes /pose\_tt and /cmd\_vel\_tt. The algorithm is quite simple, we calculate the desired distances and feed data to tintin until his current pose matches the desired pose. The algorithm runs in three phases, phase 1 is to turn towards our goal pose, phase 2 is to drive to the goal pose, and phase 3 is to turn towards the goal theta.

To execute this assignment you must run the following steps:

1. catkin\_make
2. roscore
3. Roslaunch da3\_2 begin.launch
4. Rosrun plotjuggler plotjuggler
5. view the data in plotjuggler

**Task 2 Source Code Directory**

DesignAssignments/da3/src/da3\_2/src/

**Task 2 Video**

<https://youtu.be/68UA1CVFXWs>