timed_out_and_back_TLH.py 02/01/18

Note: Run Gazebo $ **roslaunch turtlebot_gazebo turtlebot_world.launch**
Run from directory $ **python timed_out_and_back_TLH.py**
and watch TurtleBot move out 5m in about 25 seconds, rotate and come back (almost)
Reset the World in Gazebo after the trip

#!/usr/bin/env python timed_out_and_back_TLH.py

""" timed_out_and_back.py - Version 0.1 2012-03-24 Modified by TLH 2/1/2018

A basic demo of the using odometry data to move the robot along
and out-and-back trajectory.

Created for the Pi Robot Project: http://www.pirobot.org

"""
Note: Run Gazebo $ roslaunch turtlebot_gazebo turtlebot_world.launch
Run from directory $ python timed_out_and_back_TLH.py and watch TurtleBot move out 5m and back
Reset the World in Gazebo after the trip

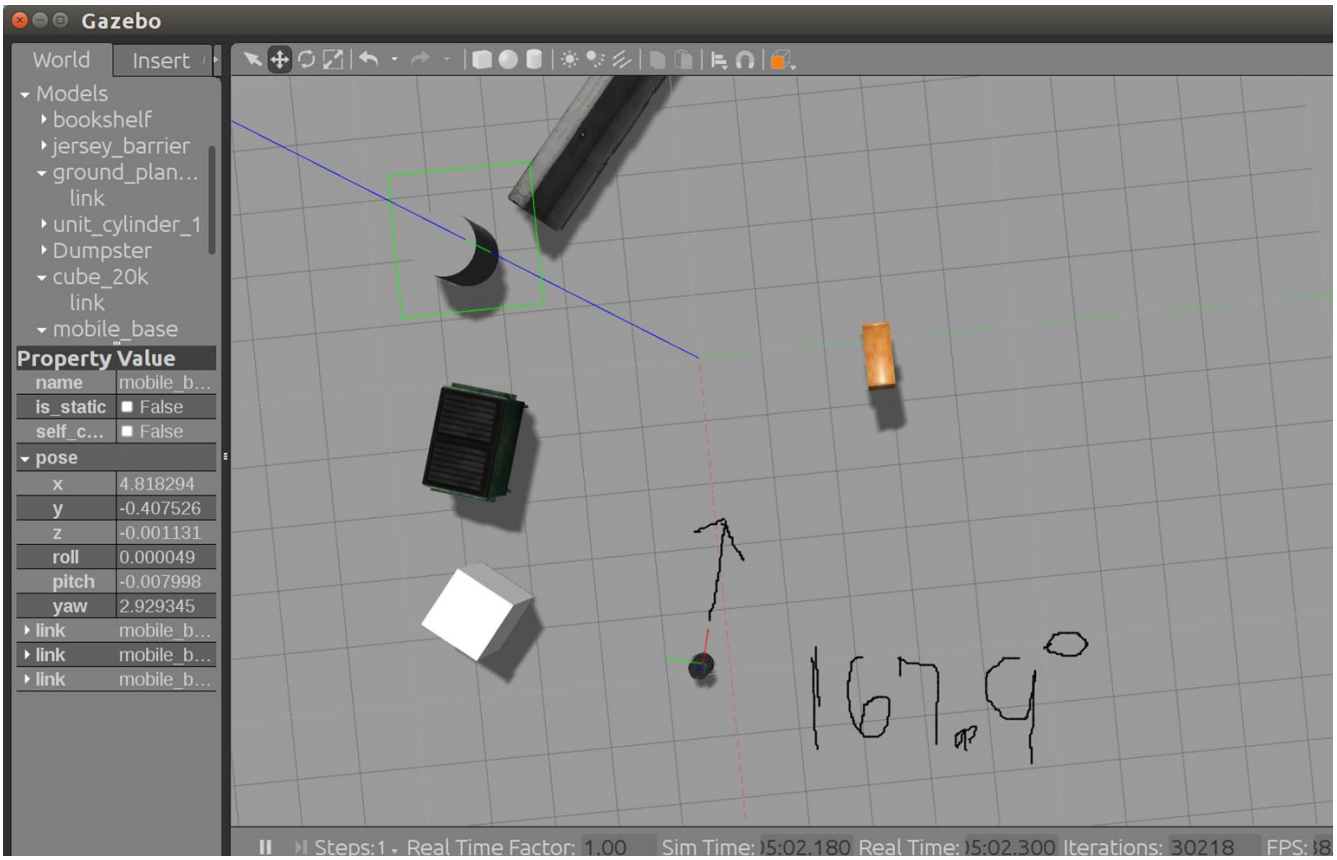
```
import rospy
from geometry_msgs.msg import Twist
from math import pi

class OutAndBack():
    def __init__(self):
        # Give the node a name
        rospy.init_node('out_and_back', anonymous=False)

        # Set rospy to exectute a shutdown function when exiting
        rospy.on_shutdown(self.shutdown)
```

```
# Publisher to control the robot's speed
# self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=10)  # Not for TurtleBot
self.cmd_vel = rospy.Publisher('/cmd_vel_mux/input/teleop', Twist, queue_size=10)
# How fast will we update the robot's movement?
rate = 50

# Set the equivalent ROS rate variable
r = rospy.Rate(rate)

# Set the forward linear speed to 0.2 meters per second
linear_speed = 0.2

# Set the travel distance to 1.0 meters
goal_distance = 5.0

# How long should it take us to get there?
linear_duration = goal_distance / linear_speed

# Set the rotation speed to 1.0 radians per second
angular_speed = 1.0

# Set the rotation angle to Pi radians (180 degrees)
goal_angle = pi

# How long should it take to rotate?
angular_duration = goal_angle / angular_speed

# Loop through the two legs of the trip
for i in range(2):
    # Initialize the movement command
    move_cmd = Twist()

    # Set the forward speed
    move_cmd.linear.x = linear_speed

    # Move forward for a time to go the desired distance
    ticks = int(linear_duration * rate)

    for t in range(ticks):
        self.cmd_vel.publish(move_cmd)
        r.sleep()

    # Stop the robot before the rotation
    move_cmd = Twist()
    self.cmd_vel.publish(move_cmd)
    rospy.sleep(1)
```

```
            # Now rotate left roughly 180 degrees

            # Set the angular speed
            move_cmd.angular.z = angular_speed

            # Rotate for a time to go 180 degrees
            ticks = int(goal_angle * rate)

            for t in range(ticks):
                self.cmd_vel.publish(move_cmd)
                r.sleep()

            # Stop the robot before the next leg
            move_cmd = Twist()
            self.cmd_vel.publish(move_cmd)
            rospy.sleep(1)

        # Stop the robot
        self.cmd_vel.publish(Twist())

    def shutdown(self):
        # Always stop the robot when shutting down the node.
        rospy.loginfo("Stopping the robot...")
        self.cmd_vel.publish(Twist())
        rospy.sleep(1)

if __name__ == '__main__':
    try:
        OutAndBack()
    except:
        rospy.loginfo("Out-and-Back node terminated.")
```

STOP PROGRAM AFTER TURTLEBOT MOVED 5 METERS

harman@D104-45931:~/Desktop$ python timed_out_and_back_TLH.py
^C[INFO] [1517533754.228104, 207.280000]: Stopping the robot...
[INFO] [1517533755.845186, 208.280000]: Out-and-Back node terminated.



Not so Hot!