MEEM 4707: Autonomous system

Spring, 2024

Lab – 3

By

Colton Kreischer

Problem 1. What will be the last printed value of "count" before the while loop gets terminated?

```
rate=rospy.Rate(5)
Stop = False
count = 0
while ~Stop:
     count = count+1
     time = count/5
     if time < 2
           print(count)
     else
           Stop = True
```

While the code would not run as is (missing import, missing colons, misuse of tilde as not operator), the loop would theoretically continue until "1.8" is printed, when count is equal to 9.

Problem 2. This file is named "controller_lab3_cw.py" and is present in the package named turtlebot3_gazebo_lab\scripts

```
#TODO : set the rate of publishing the command velocities
rate = rospy.Rate(20) #example:20hz
count=0

while not rospy.is_shutdown():

        v_cmd =0 # Command linear velocity
        omega_cmd = 0   # Command angular velocity

        #TODO : determine v_cmd and omega_cmd for the desired
        ############## WRITE YOUR CODE BELOW ######################

        v_cmd =0.2       # range:  [-0.25,0.25]
        omega_cmd = -0.5 # range : [-1.8,1.8]


        ################ YOUR CODE ENDS HERE ########################
```

In this code, "rate=rospy.Rate(20)" means the while loop will run 20 times per second. Also, the count will be added every iteration if you put "count=count+1" in the while loop as you did in problem 1.

(1) If your velocity is 0.2m/s, rate=rospy.Rate(20) and count=100; where would the turtlebot be located?

(100) / (20 Hz) = 5 s
(5s)(0.2 m/s) = 1.0 m

After 100 counts (5 s), the robot would be 1 m from where it started at count=0.

(2) Try to make the turtlebot move straight forward 1m. (*Show us your code)

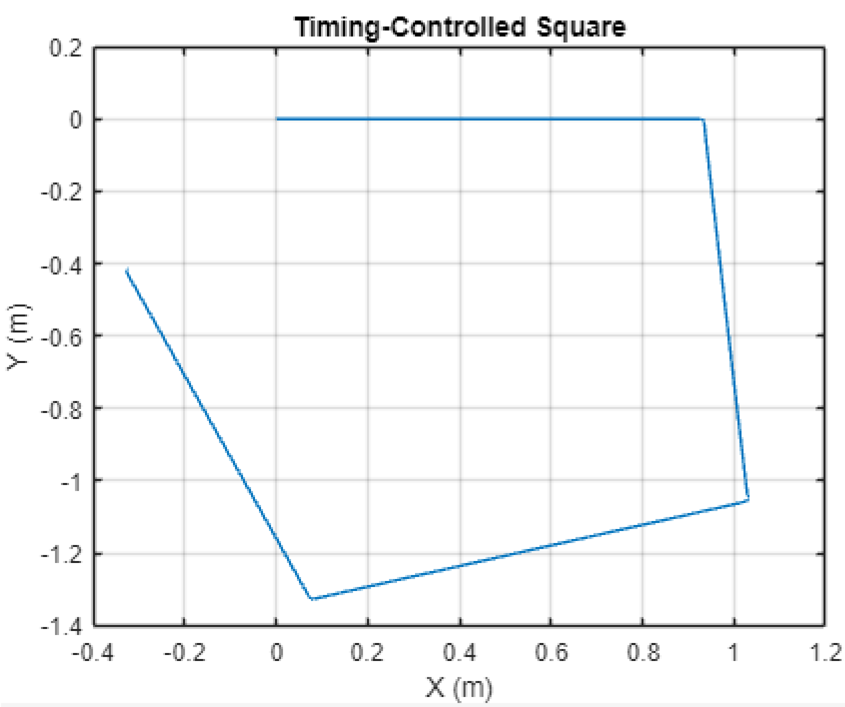
```python
#!/usr/bin/env python
import math
import rospy
from geometry_msgs.msg import Twist

if __name__ == '__main__':
    try:
        pub = rospy.Publisher('/cmd_vel', Twist, queue_size=100)
        rospy.init_node('controller_lab3_cw')

        #@TODO : set the rate of publishing the command velocities
        rate = rospy.Rate(20) # 20 Hz
        count=0

        while not rospy.is_shutdown():

            v_cmd =0 # Command linear velocity
            omega_cmd = 0   # Command angular velocity

                        #TODO : determine v_cmd and omega_cmd for the desired
                        ############## WRITE YOUR CODE BELOW ######################

            v_cmd =0    # range:  [-0.25,0.25]
            if count < 100:
                v_cmd = 0.2 # move forward at 0.2 m/s for 5 s (1 m total distance)
                count += 1
            else:
                v_cmd = 0
            omega_cmd = 0
```

(3) Spin the robot clockwise until it reaches 90 degrees on the x-axis using “if / else” conditions in Gazebo. (*Show us your code)

```python
#!/usr/bin/env python
import math
import rospy
from geometry_msgs.msg import Twist

if __name__ == '__main__':
    try:
        pub = rospy.Publisher('/cmd_vel', Twist, queue_size=100)
        rospy.init_node('controller_lab3_cw')

        #@TODO : set the rate of publishing the command velocities
        rate = rospy.Rate(20) # 20 Hz
        count=0

        while not rospy.is_shutdown():

            v_cmd =0 # Command linear velocity
            omega_cmd = 0   # Command angular velocity

                        #TODO : determine v_cmd and omega_cmd for the desired
                        ############## WRITE YOUR CODE BELOW ######################

            v_cmd =0    # range:  [-0.25,0.25]
            if count < 100:
                omega_cmd = math.radians(-90/5) # assuming units of rad/s
                count += 1
            else:
                omega_cmd = 0

            ################ YOUR CODE ENDS HERE #######################
```

(4) Repeat (2) and (3) to drive a square path. Record the trajectory of the robot in Gazebo and plot the trajectory. **Note: Your robot will not strictly follow square paths on the floor. They are for reference only. Do not try to make it run perfect square.** (*Show us your code and the resultant path)

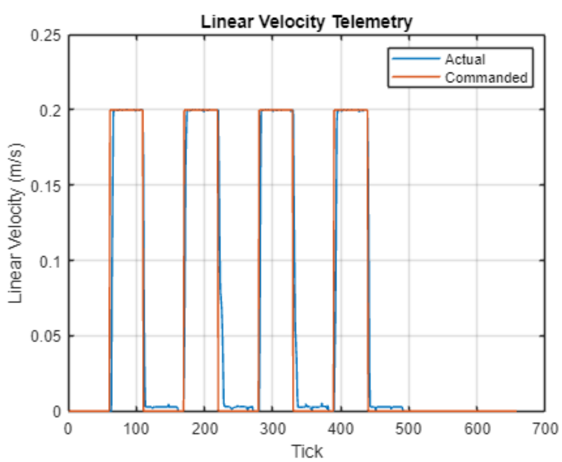
```python
#!/usr/bin/env python
import math
import rospy
from geometry_msgs.msg import Twist

if __name__ == '__main__':
    try:
        pub = rospy.Publisher('/cmd_vel', Twist, queue_size=100)
        rospy.init_node('controller_lab3_cw')

        #@TODO : set the rate of publishing the command velocities
        rate = rospy.Rate(20) # 20 Hz
        count=0

        while not rospy.is_shutdown():

            v_cmd =0 # Command linear velocity
            omega_cmd = 0   # Command angular velocity

                        #TODO : determine v_cmd and omega_cmd for the desired
                        ############## WRITE YOUR CODE BELOW ######################

            if count < (220*4):
                #  220 counts (11 s) per side, four sides

                if (count % 220) < 100:
                    # first 100 ticks (5 s) going straight at 0.2 m/s (2 m total)
                    v_cmd = 0.2
                    omega_cmd = 0

                elif (count % 220) < 200:
                    # next 100 ticks (5 s) turning cw at -90/5 deg/s (-90 deg total)
                    v_cmd = 0
                    omega_cmd = math.radians(-90/5)

                # spend 1 second buffer at rest

                count += 1

```

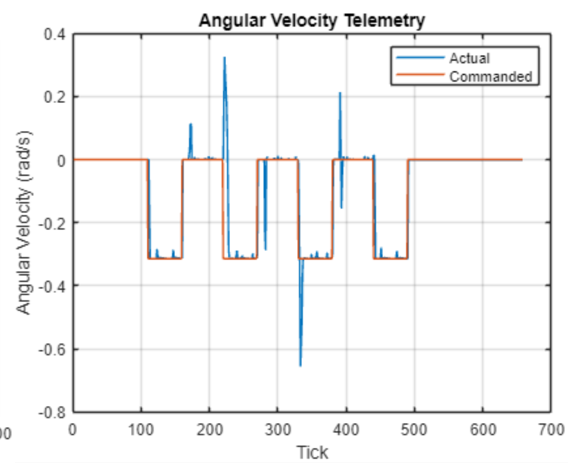


Problem 3. You can calibrate the robot's performance based on the relationship between the command and actual velocities as below. Note that b represents the distance between the wheels here.

$$V_{actual} = a_1 \times V_{command} + 0.25 \times a_2 \times b \times \omega_{command} + a_3$$
$$\omega_{actual} = \frac{a_2}{b} \times V_{command} + a_1 \times \omega_{command} + a_4$$

Calculate $a_1, a_2, a_3,$ *and* a_4 using path trajectory (the content of csv file) that you made in Problem 2. Note: Refer to pages 17&18 Simulation2. Use the values obtained in Gazebo environment (You have 4 variables and 4 equations)






```
clc, clear
close all

%% Load data
path = readmatrix("path.csv");
tele = readmatrix("Vel_Omega.csv");

%% Data manipulation
straight_v_avg = mean(tele(tele(:,3) ~= 0, 1))
straight_omega_avg = mean(tele(tele(:,3) ~= 0, 2))
turn_v_avg = mean(tele(tele(:,4) ~= 0, 1))
turn_omega_avg = mean(tele(tele(:,4) ~= 0, 2))
```

Path	V (real, avg)	Omega (real, avg)	V (commanded)	Omega (commanded)
Straight	0.1894	-4.2364e-04	0.2	0
Turn	0.0121	-0.2877	0	-0.31416

$0.1894 = (0.2)(a_1) + (a_3)$

$-4.2364e\text{-}04 = (1 / 0.287)(0.2)(a_2) + (a_4)$

$0.0121 = (0.25)(0.287)(-0.31416)(a_2) + (a_3)$

$-0.2877 = (-0.31416)(a_1) + (a_4)$

$$\begin{bmatrix} 0.2 & 0 & 1 & 0 \\ 0 & 0.69686 & 0 & 1 \\ 0 & -0.022541 & 1 & 0 \\ -0.31416 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 0.1894 \\ -0.00042364 \\ 0.0121 \\ -0.2877 \end{bmatrix}$$

$a_1 = 0.8850$

$a_2 = 0.01326$

$a_3 = 0.01240$

$a_4 = -0.009667$