LR_2 Создание простого регулятора

Подготовка

Рассмотрим содержимое файла my_best_node.py с содержажимся в ней следующим программным кодом, реализующий управление для наше 2D робота (черепашки) вдоль оси x:

```
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose

class SimpleController():
    def __init__(self):
        rospy.init_node('simple_controller', anonymous=True)
        rospy.on_shutdown(self.shutdown)
        self.cmd_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=1)
        subscriber = rospy.Subscriber("/turtle1/pose", Pose, self.pose_callback)
        subscriber_target = rospy.Subscriber("/targets_topic", Pose, self.target_callback)
        self.rate = rospy.Rate(30)

        self.target_x = 0
        self.target_y = 0

        self.x = 0
        self.y = 0

    def target_callback(self, msg:Pose):
        rospy.loginfo("target updared with: {0} {1}".format(msg.x, msg.y))
        self.target_x, self.target_y =  msg.x, msg.y

    def pose_callback(self, msg:Pose):
        # rospy.loginfo("cur pose: {0} {1}".format(msg.x, msg.y))
        self.update_control(msg.x, msg.y)

    def update_control(self, x, y):
        self.x, self.y = x, y

    def spin(self):
        while not rospy.is_shutdown():
            twist_msg = Twist()
            diff = self.x - self.target_x
            if diff > 0.1:
                twist_msg.linear.x  = -1.0
            elif self.x - self.target_x < -0.1:
                twist_msg.linear.x  = 1.0
            else:
                twist_msg.linear.x  = 0.0
            self.cmd_vel_pub.publish(twist_msg)
            self.rate.sleep()

    def shutdown(self):
        self.cmd_vel_pub.publish(Twist())
        rospy.sleep(1)
```

```
simple_mover = SimpleController()
simple_mover.spin()
```

Описание лабораторной работы № 2

1. Склонировать актуальные изменения из репозитория **ros_course_2023 (likerobotics)** в локальную копию (в папке локальной копии выполнить)

   `git pull`

2. Скопировать содержимое папки practice_2 из локальной копии в рабочее пространство catkin (т.е. скопировать папку **my_best_controller** в папку **catkin_ws/src/**).

3. Запуск нескольких экземляров при помощи пространства имен

   a. Откройте терминал и запустите ROS Master

      `roscore`

   b. Создайте новый файл запуска с названием **lab2_setup.launch** и создайте два пространства имен с названиями ns1_ISUID и ns2_ISUID, где ISUID это ваш номер в ИСУ. Пример:

      ```
      <launch>
          <group ns="sim1">

          </group>
          <group ns="sim2">

          </group>
      </launch>
      ```

   c. Внутри каждого пространства имен необходимо запустить turtlesim.

   d. Необходимо в папке scripts создать файл с названием lab2_controller.py и внем написать программу движения для первой черепашки согласно разделу Движение по траектории (пункт 4).

      > 💡 Обратите внимание на названия ресурсов, запускаемых внутри пространства имен.

   e. Вторая черепашка должна подписываться на топик …/pose первой черпашки и повторять его действия (запрещено использовать remap). Управляющая программа для второй черепашки также должна быть прописана в файле lab2_controller.py

   f. Обратите внимание, что внутри контроллера необходимо использовать имена ресурсов относительные или частные(private), иначе будет конфиликт имен при запуске дувх

экземпляров одного и того же контроллера.

4. Программа движения:

   Черепашка №1 должна пройти последовательность точек согласно варианту, указанному в таблице. В таблице указаны 5 разных точек, ваша программа должна проходить только по тем точкам, у которых указаны координаты напротив вашего номера варианта. Координаты указаны в формате (x,y). Поряядок прохождения точек не важен.

5. Готовый пакет должен запускаться при помощи команды

   `roslaunch lab2_setup.launch`

6. Время выполнения поставленной задачи вашим пакетом не должно превышать 5 минут. Т.е. хаотичное движение черепашки не сработает!

Номера вариантов для студентов соответствуют последней цифре ISU ID:

Последняя цифра ису ID	Точка 1 ()	Точка2	Точка 3	Точка 4	Точка 5
1	(2,3)	(4,2)	(7,1)		(8,4)
2	(2,3)	(4,2)		(5,4)	
3	(2,3)		(7,1)	(5,4)	(8,4)
4		(4,2)		(5,4)	(8,4)
5	(2,3)		(7,1)		(8,4)
6		(4,2)	(7,1)	(5,4)	
7	(2,3)	(4,2)		(5,4)	
8			(7,1)	(5,4)	(8,4)
9	(2,3)	(4,2)	(7,1)	(5,4)	
0	(2,3)	(4,2)	(7,1)		(8,4)

Как отправить готовое решение на проверку?

Нобходимо в вашей копии репозитория **ros_course_2023** (которая хранится на вашем гитлабе) создать папку **practice_2** и в нее скопировать папку **my_best_controller** с вашим решением. После это сделать

```
git add .
git commit -m "my best lab 2"
git push
```