Práctica 3: Robot Operating System

Automatización y Robótica

Francisco Javier Pérez Martínez

12 de mayo de 2022

Índice

1. Descripción

Para el presente documento, correspondiente a la práctica 3 de la asignatura, se trabajará con una plataforma de software dedicada a la robótica de código abierta denominada ROS. Para el desarrollo de la práctica, se ha descargado la máquina virtual con Linux+ROS proporcionada en el enunciado.

2. Ejercicio 1

Desarrolla un programa en Python que se llamará minibot.py. Este simulará un pequeño robot móvil que se encuentra en un punto de una habitación de paredes paralelas dos a dos, saluda diciendo que es Minibot y que si queremos escanear la habitación. En caso de que contestásemos que sí, nos pedirá cuatro medidas del láser. Una vez hemos introducido las cuatro medidas mostrará como resultado cuál es el largo y ancho de la habitación, la superficie total y la posición del robot en coordenadas X, Y. En caso de que le contestemos que no, Minibot se despide y termina el programa.

2.1. Implementación y resultados

A partir de la especificación dada, se ha implementado el siguiente código en Python:

```
1  print("¡Hola, soy Minibot!")
2  print("¿Quieres escanear la habitación?")
3
4  while True:
5      res = input()
6
7      if (res == "y" or res == "Y"):
8          print("Introduce la primera medida del láser (arriba): ")
9          medida1 = input()
10         print("Introduce la segunda medida del láser (abajo): ")
11         medida2 = input()
12         print("Introduce la tercera medida del láser (derecha): ")
13         medida3 = input()
14         print("Introduce la cuarta medida del láser (izquierda): ")
15         medida4 = input()
16
17         largo = int(medida1) + int(medida2)
18         print("El largo del robot es de", largo)
19
20         ancho = int(medida3) + int(medida4)
21         print("El ancho del robot es de", ancho)
22
23         superficie = largo * ancho
24         print("La superficie total del robot es de", superficie)
25         print("Posición del robot (X,Y) = (", medida4, ",", medida2, ")")
26         break
27
```

```
28      elif (res == "n" or res == "N"):
29          print("¡Minibot se despide! - Fin del programa")
30          break
31
32      else:
33          print("No te he entendido, vuelve a introducir una respuesta válida (y/Y o n/N): ")
```

Al iniciar el programa, el robot saludará y preguntará si se desea escanear la habitación, según la respuesta el programa éste continuará de la siguiente forma:

- Si respondemos que "y/Y", el programa pedirá las 4 medidas y calculará el largo, el ancho, la superficie y la posición del robot finalizando así el programa.
- Si respondemos que "n/N", el robot se despedirá y finalizará el programa.
- Si respondemos otra cosa que no sea las 2 opciones anteriores, el programa estará continuamente esperando una respuesta válida (y/Y o n/N).

A continuación, se muestra un ejemplo de ejecución del código anterior:

```
python .\minibot.py
¡Hola, soy Minibot!
¿Quieres escanear la habitación?
mañana
No te he entendido, vuelve a introducir una respuesta válida (y/Y o n/N):
y
Introduce la primera medida del láser (arriba):
2
Introduce la segunda medida del láser (abajo):
2
Introduce la tercera medida del láser (derecha):
2
Introduce la cuarta medida del láser (izquierda):
2
El largo del robot es de 4
El ancho del robot es de 4
La superficie total del robot es de 16
Posición del robot (X,Y) = ( 2 , 2 )
```

Figura 1: Salida programa minibot.py

3. Ejercicio 2

Para este ejercicio, se nos pide acceder a la siguiente dirección http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29 y realizar el tutorial de "Simple Publisher and Subscriber" con la diferencia de que se debe personalizar el mensaje añadiéndole un toque de originalidad al ejemplo.

En mi caso, el mensaje enviado se trata de un String con el mensaje fijo de "Fecha y hora actuales" seguidamente de la fecha y hora actual del sistema, la cual por cada mensaje enviado, se va actualizando respectivamente, segundo a segundo.

3.1. Implementación y resultados

Del ejemplo dado, tenemos 2 códigos: talker.py y listener.py. El primer corresponde al nodo publicador y el segundo al nodo suscriptor.

3.1.1. Talker.py

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
from datetime import datetime
import time

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz

    while not rospy.is_shutdown():
        # Enviamos la fecha y hora actuales utilizando la funcion datetime.now()
        # Mediante la funcion strftime le indicamos el formato que queremos
        str_date = 'Fecha y hora actuales: %s' % str(datetime.now().strftime("%d/%m/%Y %H:%M:%S"))

        # sleep de 1 segundo para poder apreciar el mensaje adecuadamente
        time.sleep(1)
        rospy.loginfo(str_date)
        pub.publish(str_date)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

A continuación, se muestra un ejemplo de ejecución del código en cuestión:

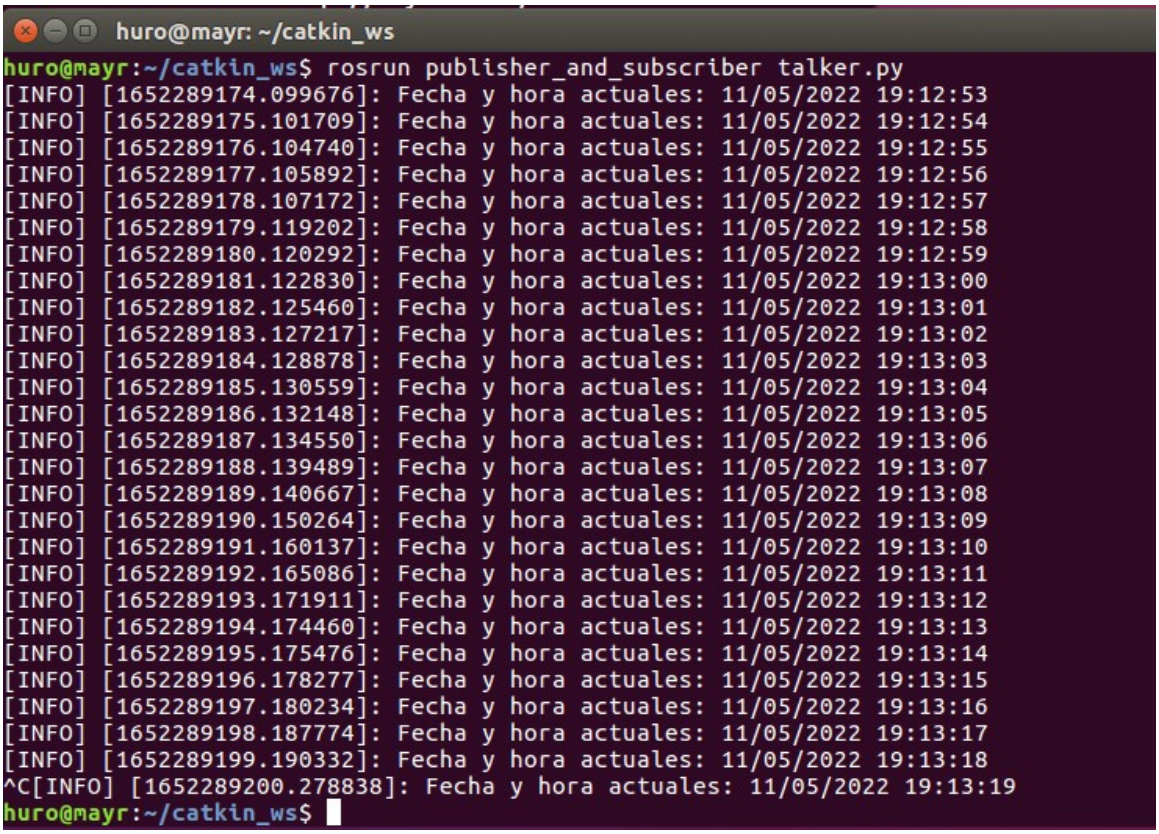

Figura 2: Salida programa talker.py

3.1.2. Listener.py

```python
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + 'I heard %s', data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber('chatter', String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

A continuación, se muestra un ejemplo de ejecución del código en cuestión:

```
huro@mayr: ~/catkin_ws
[INFO] [1652289176.105418]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:12:55
[INFO] [1652289177.106725]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:12:56
[INFO] [1652289178.108111]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:12:57
[INFO] [1652289179.120067]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:12:58
[INFO] [1652289180.120975]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:12:59
[INFO] [1652289181.123675]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:00
[INFO] [1652289182.126191]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:01
[INFO] [1652289183.128135]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:02
[INFO] [1652289184.129702]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:03
[INFO] [1652289185.131232]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:04
[INFO] [1652289186.132996]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:05
[INFO] [1652289187.135321]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:06
[INFO] [1652289188.140240]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:07
[INFO] [1652289189.141565]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:08
[INFO] [1652289190.151087]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:09
[INFO] [1652289191.160875]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:10
[INFO] [1652289192.165793]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:11
[INFO] [1652289193.172821]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:12
[INFO] [1652289194.175369]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:13
[INFO] [1652289195.176074]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:14
[INFO] [1652289196.179146]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:15
[INFO] [1652289197.181056]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:16
[INFO] [1652289198.188719]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:17
[INFO] [1652289199.191109]: /listener_8086_16522891749511 heard Fecha y hora actuales: 11/05/2022 19:13:18
^Churo@mayr:~/catkin_ws$
```

Figura 3: Salida programa listener.py

3.1.3. Resultado conjunto

Finalmente, se muestra una captura con los 2 nodos (publicador y suscriptor) y el nodo master en ejecución:

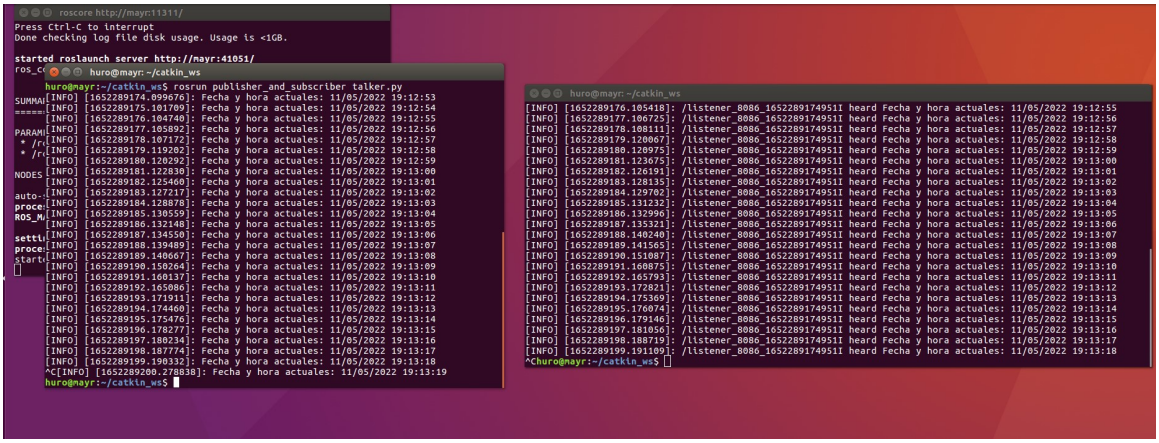

Figura 4: Muestra del nodo master + los 2 nodos creados

NOTA: si queremos ejecutar cada nodo en un terminal distinto, debemos ejecutar previamente el siguiente comando al abrir el terminal situándonos en el directorio /home/user/catkin_ws:

```
source ./devel/setup.bash
```