Test de Manejo

ROS y más ROS

30 de agosto de 2023

Capacitación de hoy

1. Recapitulación capacitación anterior
2. Demo
3. Desafío
4. Test de Manejo

Duckietown
Engineering
Chile

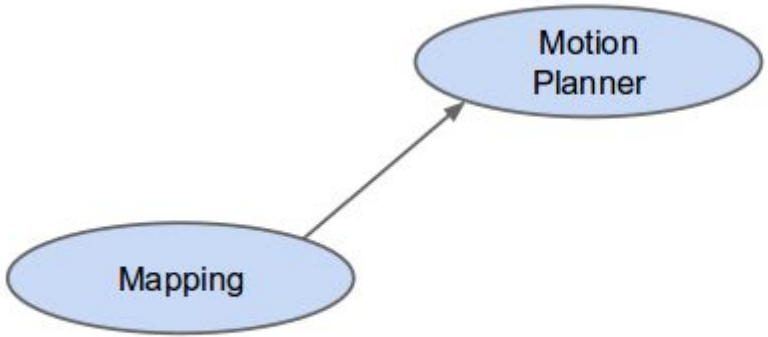
Recapitulación

ROS

Arquitectura: Mensajes

-Un mensaje es simplemente una estructura de datos, que comprende los tipos de campos.

-Lenguaje de representación de datos agnóstico. C++ puede hablar con Python.



File: pos.msg

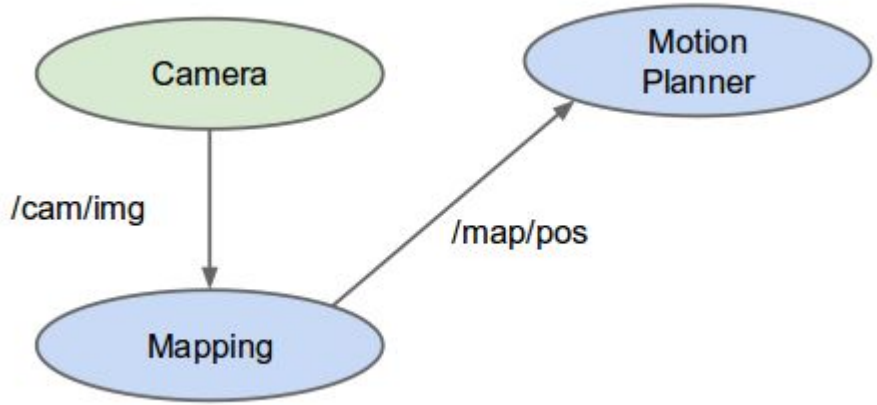
string robotName
uint32 posX
uint32 posY
uint32 goalX
uint32 goalY

Duckietown Engineering Chile

Arquitectura: Tópicos

-Flujos de datos con publicación/suscripción semántica.

-Ellos son los únicos que pueden ser identificados por su nombre.

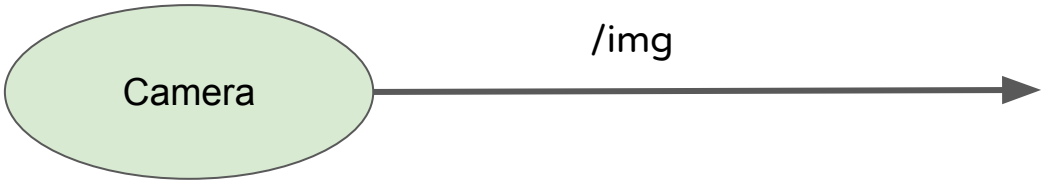


Arquitectura: Subscriber

- Un subscriber es un tipo de nodo que escucha mensajes de uno o más tópicos.

Arquitectura: Publisher

- Un publisher es un tipo de nodo que publica en uno o más tópicos.



Checklist

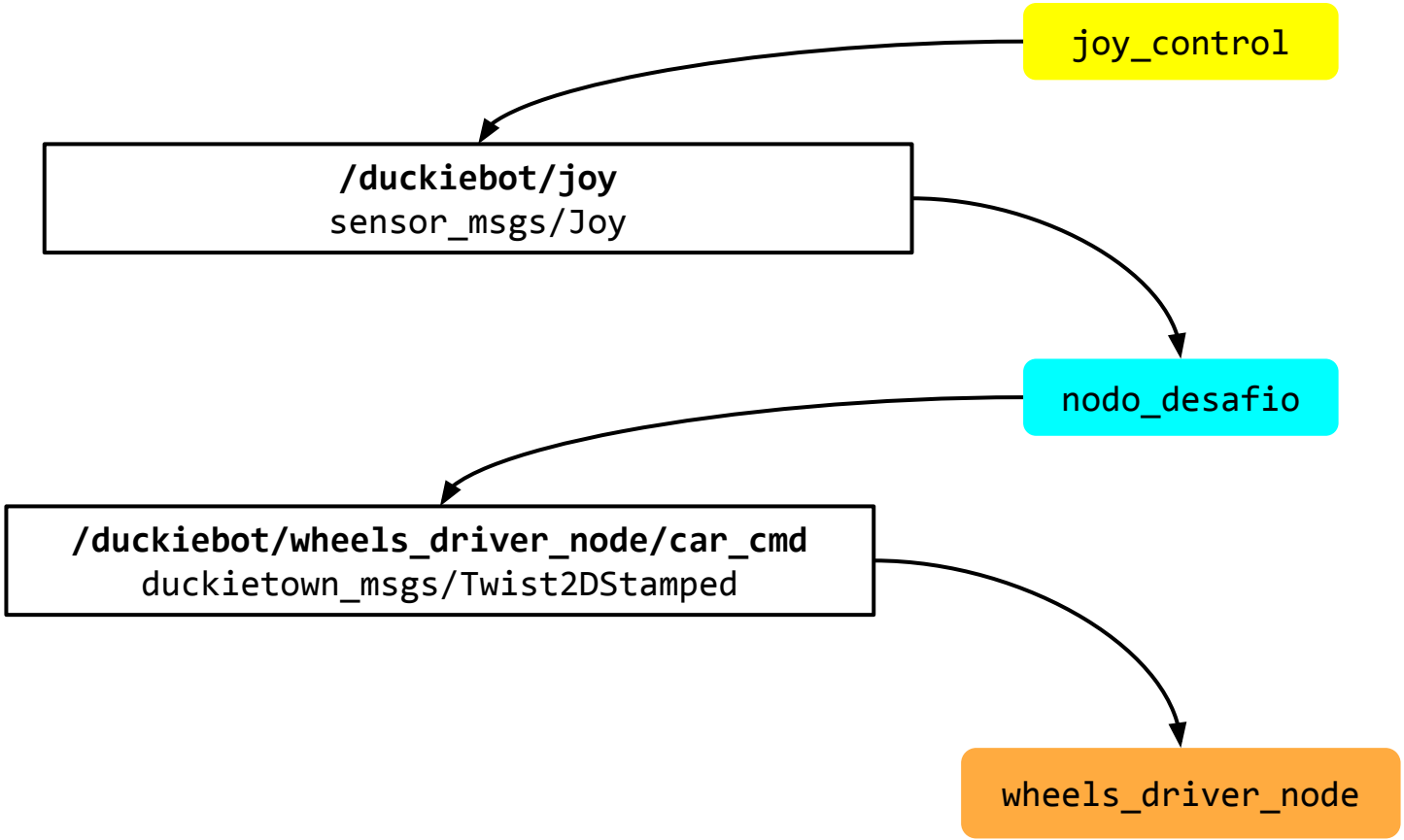
1. roscore
2. Abrir nueva terminal con byobu (F2)
3. roslaunch training_pkg duckie_core.launch veh:=duckiebot
4. Abrir ventana nueva (F2)
5. Verificar que este todo corriendo con: rostopic list
6. Verificar que los motores funcionan:
7. rostopic pub -1 /duckiebot/wheels_driver_node/car_cmd duckietown_msgs/Twist2DStamped “header”: *[tab]*

Actividad

Implementación funcionalidades duckiebot

Desafío

- ★ Crear un nodo de ROS que se suscriba al tópico donde publica el Joystick del Duckiebot y publique en el tópico de las ruedas.
- ★ En español: Mapear análogo del joystick a movimiento de ruedas, tal que el robot pueda moverse hacia adelante/atrás y girar.
- ★ Si es presionado el botón “B”, todos los movimientos en los ejes deben anularse (= 0).



Duckietown
Engineering
Chile

Test de manejo

Exámen al final de la capacitación para obtener su licencia de conducir

Su Duckiebot debe:

1. Avanzar
2. Retroceder
3. Girar
4. Tener un freno de emergencia

Duckietown Engineering Chile

Ejemplo : Publicación

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

import rospy
from std_msgs.msg import String

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():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

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

Ejemplo : Subscripción

```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():

    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()
```

Duckietown Engineering Chile

Template

File: /home/dario/duckietown/catkin...s/src/ros_cap/src/template.jpg Page 1 of 1

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

import rospy #importar ros para python
from std_msgs.msg import String, Int32 # importar mensajes de ROS tipo String y tipo
Int32
from geometry_msgs.msg import Twist # importar mensajes de ROS tipo geometry / Twist


class Template(object):
        def __init__(self, args):
                super(Template, self).__init__()
                self.args = args


        #def publicar(self):

        #def callback(self,msg):


def main():
        rospy.init_node('test') #creacion y registro del nodo!

        obj = Template('args') # Crea un objeto del tipo Template, cuya definicion se
encuentra arriba

        #objeto.publicar() #llama al metodo publicar del objeto obj de tipo Template

        #rospy.spin() #funcion de ROS que evita que el programa termine -  se debe
usar en  Subscribers


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

Duckietown Engineering Chile

sensor_msgs/Joy

std_msgs/Header header

float32[] axes

int32[] buttons

Mensajes para subscriber y publisher

duckietown_msgs/Twist2DStamped

Header header

float32 v

float32 omega

Duckietown
Engineering
Chile

Importante

Importar:

from sensor_msgs.msg import Joy

from duckietown_msgs.msg import Twist2DStamped

Probar archivo

/duckietown/catkin_ws/src/ros_cap/src

chmod +x template(copy).py

rosrun ros_cap template(copy).py

```
from sensor_msgs.msg import Joy

from duckietown_msgs.msg import Twist2DStamped

self.publisher = rospy.Publisher(“topico”,tipo de mensaje, queue_size = “x”)

self.subscriber = rospy.Subscriber(“topico”, tipo de mensaje,self.callback)

self.publisher.publish(self.twist)
```

Duckietown
Engineering
Chile

Test de Manejo

ROS y más ROS

30 de agosto de 2023