Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

Fundamentación de Robótica

TE3001B gpo 101

Semestre ago-dic 2022

Challenge 01

Integrantes:

Jonathan Josafat Vázquez Suárez	A01734225
Jhonatan Yael Martinez Vargas	A01734193
Diego García Rueda	A01735217

Profesor

Rigoberto Cerino Jiménez

Fecha de Entrega: 20 de Febrero del 2023

En este proyecto se han creado dos nodos de ROS, el primero de ellos como publisher a 2 topics, mientras que el otro nodo como subscriber de 2 tópicos y publisher de 1. El resultado final de este proyecto es una gráfica que muestra tanto la función sinusoidal original como la procesada, permitiendo una comparación visual entre ambas. Este proyecto es un buen ejemplo de cómo se pueden utilizar los nodos de ROS para realizar tareas de procesamiento y visualización de datos.

Objetivos

Comprender el uso general de ROS, creación y uso de nodos, así como visualización y procesamiento de mensajes.

Introducción

ROS en significado por sus siglas (Robot Operating System) es un framework de código abierto que es utilizado para el control y desarrollo de sistemas autónomos, este framework está principalmente enfocado a el aprendizaje automático e inteligencia artificial en robótica. La estructura básica de ROS se basa en Nodos, tópicos y servicios.

Un nodo es un punto de interconexión dentro de la unidad de procesamiento básica de ROS. Los tópicos y servicios son los métodos de intercomunicación entre nodos, un tópico funciona como respuesta unidireccional con un nombre único, los nodos pueden suscribirse o publicar estos tópicos, un servicio, es la manera en que puedes solicitar información dentro de un nodo.

La comunicación dentro de ROS es controlada mediante el master quien asigna diferentes IP y puertos a cada nodo del sistema y se puede realizar comunicación de manera local o a través de internet.

Solución del problema

Para la solución de este challenge se siguió la metodología presentada por el socio formador. En esta se realizaron 2 códigos siendo estos los nodos del proyecto. las funciones de los nodos se describirán a continuación:

- **Signal_generator:**

Este nodo se encarga de generar una señal sinusoidal, este nodo publicará 2 tópicos llamados “Time” y “Signal". El primer tópico “Time” se genera publicando la diferencia de tiempo cuando el nodo inicia y cuando publica este tiempo. En el tópico “Signal” se utiliza la función sin de *numpy* para generar esta señal usando como parámetro el tiempo anteriormente establecido. Finalmente estos 2 valores son imprimidos en la terminal y publicados.

El código de este nodo es el siguiente:

```python
# !/usr/bin/env/ python
import rospy
import numpy as np
from std_msgs.msg import Float32

def talker():

    # Se declaran las dos variables para los publisher
    time_pub = rospy.Publisher("Time", Float32, queue_size = 10)
    signal_pub = rospy.Publisher("Signal", Float32, queue_size = 10)

    # Se inicializa nodo
    rospy.init_node("signal_generator", anonymous = True)

    # Se establece 10 HZ como la frecuencia con la que se van a publicar los datos
    rate = rospy.Rate(10)

    # Variable para cálculo del tiempo
    main_time = rospy.get_time()

    while not rospy.is_shutdown():

        # Variable para cálculo del tiempo
        tm = rospy.get_time() - main_time

        # Se muestra en la terminal el tiempo de ejecución
        rospy.loginfo("Time: %f" % tm)

        # Se publica el tiempo
        time_pub.publish(tm)

        # Se calcula la función sin respecto al tiempo
        data = np.sin(tm)

        # Se muestra en pantalla el valor calculado por la función sin
        rospy.loginfo("Value: %f" % data)

        # Se publica el valor calculado por la función sin
        signal_pub.publish(data)

        rate.sleep()

if __name__ == '__main__':

    try:
        talker()

    except rospy.ROSInterruptExeption:
        pass
```

- **Process:**

Este es el nodo diseñado de modificar y procesar la señal sinusoidal publicada por el nodo “signal_generator”, para esto se suscribe a los tópicos anteriormente mencionados en el nodo anterior, en cada una de estas suscripciones se utiliza una función *callback* diferente, esto debido a que este nodo es suscriptor de más de 1 tópico. Los valores recibidos de los tópicos son guardados en variables globales para su uso. Este nodo a su vez publica un tópico llamado “Proc_signal” que es la señal ya procesada. Para el procesamiento de la señal se utiliza la siguiente función trigonométrica:

$$y = A*\sin(\omega t) = A*\sin(\omega t + 0)$$
$$\sin(\omega t + \varphi) = \sin(\omega t)*\cos(\varphi) + \cos(\omega t)*\sin(\varphi)$$
$$y = A*\sin(\omega t + \varphi) = A*\sin(\omega t)*\cos(\varphi) + A*\cos(\omega t)*\sin(\varphi)$$

Esta función nos permite hacer un desfase de la función sinusoidal como es requerido en el reto, en este caso el valor de A es de 0.5 debido a que esto genera que la amplitud de la señal se divida a la mitad y al final de la función agregamos un offset que tiene un valor de 0.5 qu e genera que todos los valores de la nueva función sinusoidal sean positivos. Finalmente esta nueva señal es publicada por el nodo.

El código del nodo es el siguiente:

```
# !/usr/bin/env/ python
import rospy
import numpy as np
from std_msgs.msg import Float32

tmp = 0.0
value = 0.0

def callback_time(msg):

   global tmp
   tmp = msg.data

   rospy.loginfo("Tiempo Recibido: %f" % tmp)

def callback_signal(msg):

   global value
   value = msg.data

   rospy.loginfo("f(x) Recibido %f" % value)

def init():
```

```
    #  Se declara la variable para el publisher
    rqt_pub = rospy.Publisher("Proc_signal", Float32, queue_size = 10)

    # Se inicializa el nodo
    rospy.init_node("process", anonymous = True)

    # Se definen los topicos
    rospy.Subscriber("Time", Float32, callback_time)
    rospy.Subscriber("Signal", Float32, callback_signal)

    # Se estable 10 HZ como la frecuencia con la cual se van a publicar los datos
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():

        # Se mandan a llamar las variables globales
        global tmp
        global value

        # Se genera la señal desplazadda y escalada
             new_value  =  0.5  *  ((np.sin(tmp)  *  np.cos(np.pi))  +  (np.cos(tmp)  *
np.sin(np.pi))) + 0.5

        # Se mueestra en terminal y se publican los valores de la nueva señal
        rospy.loginfo("Dato modificado: %f" % new_value)
        rqt_pub.publish(new_value)

        rate.sleep()

if __name__ == '__main__':


    try:
        init()

    except rospy.ROSInterruptExeption:
        pass
```

- **rqt_plot:**

Se utiliza esta herramienta para graficar las 2 señales sinusoidales, la original generada por el primer nodo y la señal ya procesada por el segundo nodo, esto se realiza ya que *rqt_plot* se suscribe a los 2 tópicos que contienen los valores de ambas señales. Finalmente, los 2 nodos y la ejecución de *rqt_plot* se realizan mediante un archivo **launch** que ejecuta todo lo anteriormente mencionado.

El código es el siguiente:

```
<?xml version="1.0" ?>

<launch>

    <node name = "signal_generator" pkg = "courseworks" type = "signal_generator.py" output = "screen" launch-prefix = "gnome-terminal --command" />
    <node name = "process" pkg = "courseworks" type = "process.py" output = "screen" launch-prefix = "gnome-terminal --command" />

    <!-- rqt launch -->
        <node   name="rqt_plot"   pkg="rqt_plot"   type="rqt_plot"   output="screen" args="/Signal /Proc_signal"/>

</launch>
```

Resultados

De acuerdo a lo realizado, se obtuvo la siguiente gráfica usando la herramienta *rqt plot*:

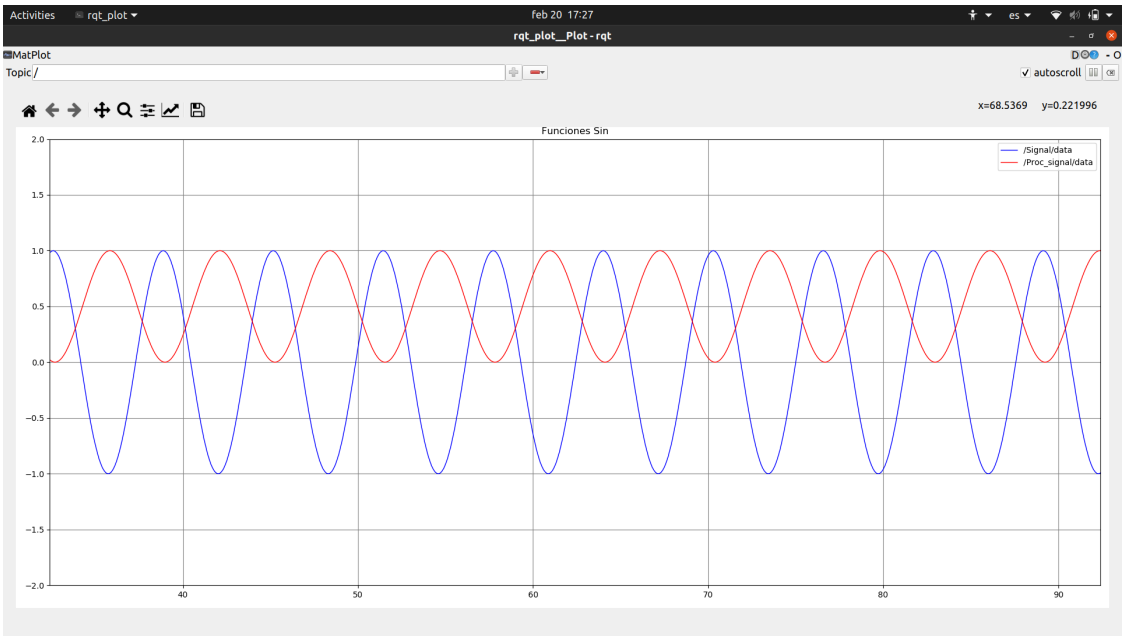

Figura 01. Resultados

La señal sinusoidal de color azul representa la señal creada por el nodo *signal_generator* mientras que la señal sinuidal dde color rojo representa la señal procesada por el nodo *process*. En esta segunda señal se puede apreciar los cambios solicitados en el challenge, se puede observar que la amplitud de la onda procesada es de la mitad del valor original, de la misma manera, la señal se encuentra desfasada respecto a su señal original. Finalmente, todos los valores de la onda procesada son valores positivos cumpliendo con todos los requisitos solicitados por el challenge.

Conclusiones

Consideramos que para aprender a usar ROS este pequeño challenge fue bastante divertido e interesante de implementar, pues al momento de comenzar a programar aunque ya teníamos nociones básicas de lo que es ROS, surgieron las confusiones sobre cómo funciona el sistema de ficheros que maneja esta ambiente de trabajo y además tuvimos que dar un pequeño repaso de lo que fue señales para poder solucionar el cambio que se nos pedía, sin embargo el resultados fue satisfactorio y de gran aprendizaje, pues ahora la funcionalidad de nodos y tópicos sentimos que ya la tenemos presente al momento de implementar código y nos es mucho más fácil manipular estos mismos.