**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**





Laboratorio de Principios de Mecatrónica

**Práctica 5. ROS**

Estudiantes:

* Plauchú Rodríguez Rodrigo 182671
* Castillejos Corzo Victor Hugo 182344

Asignatura: Laboratorio de Principios de Mecatrónica

Docente: M.I. Sergio Hernández Sánchez

Grupo: \_04\_

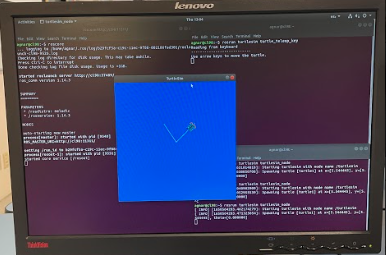
1. **Introducción**

El objetivo de la práctica es familiarizarse con la plataforma de ROS, instalada en UBUNTU para poder realizar una serie de ejercicios y acercarse un poco a lo que sería la programación de Robótica. Así mismo, aprender el uso del shell o terminal en Linux, para poder interactuar con ROS.

1. **Objetivos**

* Reconocer las características de ROS como una plataforma de comunicación, control y simulación para aplicaciones de Robótica.
* Lograr un acercamiento a la simulación de agentes móviles en un entorno especializado.
* Realizar movimiento de un robot móvil de tipo diferencial mediante la simulación Turtlesim.

1. **Material y equipo utilizado**
   1. Ubuntu 18.04 o 20.04
   2. ROS Melodic o ROS Noetic según la versión de Ubuntu con la que se cuente
2. **Experimentos**
   1. **Actividad 1 – Generación de la inicial del su nombre**

****

**Como se puede observar están las 3 terminales abiertas y se dibujó con la tortuga la letra V por el miembro del equipo Victor.**

* 1. **Actividad 2 – Creación de un publicador y un suscriptor**

**%Código listener**

**#!/usr/bin/env python %configura entorno e importamos libs**

**import rospy**

**from std\_msgs.msg import String**

**%método que recibe o escucha el mensaje de talker**

**def chatter\_callback(message):**

**rospy.loginfo(rospy.get\_caller\_id() + "M de %s", message.data)**

**%Configura el sistema de suscripción**

**def listener():**

**rospy.init\_node('listener', anonymous = True)**

**rospy.Subscriber("chatter", String, chatter\_callback)**

**rospy.spin()**

**%main**

**if \_\_name\_\_ == '\_\_main\_\_':**

**listener()**

**%Código Taller**

**%conf. de ambiente e importa libs**

**#!/usr/bin/env python**

**import rospy**

**from std\_msgs.msg import String**

**def talker():**

**#crear un publicador, donde especifiquemos el nombre del topico, despues el tipo de mensaje y el tamanio de la cola**

**pub = rospy.Publisher('chatter', String, queue\_size=10)**

**#vamos a inicializar el nodo**

**rospy.init\_node('talker', anonymous = True)**

**#Definir la frecuencia de publicacion**

**rate = rospy.Rate(1) #1 Hz**

**%look para hacerlo hasta que maten el proceso**

**i = 0**

**while not rospy.is\_shutdown():**

**hello\_str = "Motomami %s" %i**

**rospy.loginfo(hello\_str)**

**pub.publish(hello\_str)**

**i = i + 1**

**%main**

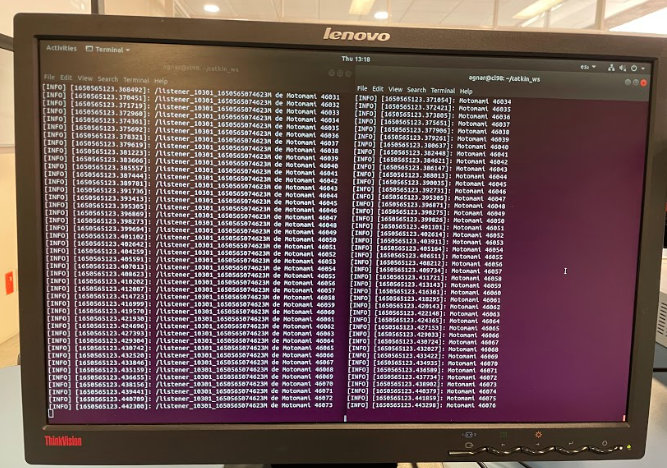
**if \_\_name\_\_ == '\_\_main\_\_':**

**try:**

**talker();**

**except rospy.ROSInterruptionException:**

**pass**

****

**Corre el listener y el talker obteniendo con éxito el mensaje**

* 1. **Actividad 3 – Creación de una trayectoria circular**

**%conf. de ambiente y libs**

**#!/usr/bin/env python**

**import rospy**

**from geometry\_msgs.msg import Twist**

**import sys**

**%en este método recibe el radio para dibujar la tort**

**def turtle\_circle(radius):**

**rospy.init\_node('turtlesim', anonymous=True)**

**pub = rospy.Publisher('/turtle1/cmd\_vel',**

**Twist, queue\_size=10)**

**rate = rospy.Rate(10)**

**vel = Twist()**

**%se define la velocidad con la que hace el trazo**

**while not rospy.is\_shutdown():**

**vel.linear.x = radius**

**vel.linear.y = 0**

**vel.linear.z = 0**

**vel.angular.x = 0**

**vel.angular.y = 0**

**vel.angular.z = 1**

**rospy.loginfo("Radius = %f",**

**radius)**

**pub.publish(vel)**

**rate.sleep()**

**%main**

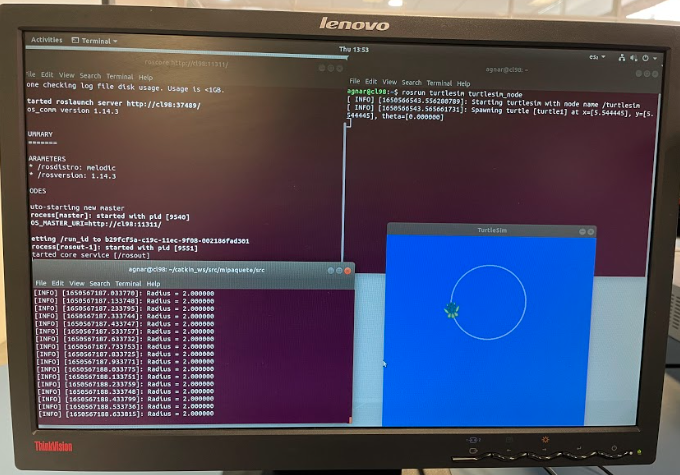
**if \_\_name\_\_ == '\_\_main\_\_':**

**try:**

**turtle\_circle(float(sys.argv[1]))**

**except rospy.ROSInterruptException:**

**pass**

****

**Se muestran las terminales y el círculo de la tortuga.**

* 1. **Actividad 4 – Creación de una trayectoria rectangular**

**%conf de ambiente y libs.**

**#!/usr/bin/env python**

**import rospy**

**import math**

**from geometry\_msgs.msg import Twist**

**import sys**

**%recibe base y altura**

**def turtle\_rec(b, a):**

**rospy.init\_node('turtlesim', anonymous=True)**

**pub = rospy.Publisher('/turtle1/cmd\_vel',**

**Twist, queue\_size=10)**

**rate = rospy.Rate(1)**

**vel = Twist()**

**%while para hacer el rectángulo rotando pi/2**

**while not rospy.is\_shutdown():**

**rate.sleep()**

**vel.linear.x = b**

**vel.linear.y = 0**

**vel.linear.z = 0**

**vel.angular.x = 0**

**vel.angular.y = 0**

**vel.angular.z = 0**

**pub.publish(vel)**

**rate.sleep()**

**vel.linear.x = 0**

**vel.linear.y = 0**

**vel.linear.z = 0**

**vel.angular.x = 0**

**vel.angular.y = 0**

**vel.angular.z = 1.56200**

**pub.publish(vel)**

**rate.sleep()**

**vel.linear.x = a**

**vel.linear.y = 0**

**vel.linear.z = 0**

**vel.angular.x = 0**

**vel.angular.y = 0**

**vel.angular.z = 0**

**pub.publish(vel)**

**rate.sleep()**

**vel.linear.x = 0**

**vel.linear.y = 0**

**vel.linear.z = 0**

**vel.angular.x = 0**

**vel.angular.y = 0**

**vel.angular.z = 1.56200**

**pub.publish(vel)**

**%main**

**if \_\_name\_\_ == '\_\_main\_\_':**

**try:**

**turtle\_rec(float(sys.argv[1]), float(sys.argv[2]))**

**except rospy.ROSInterruptException:**

**pass**

1. **Conclusiones**

Después de haber realizado el curso de introducción a ROS y completado los ejercicios podemos concluir que es una plataforma muy poderosa y que sin duda merece dedicarle tiempo para aprender. El uso de la terminal es increíblemente poderoso y nos servirá no solo para el laboratorio y la clase, sino para muchas otras cosas. Estamos satisfechos con los resultados y esperamos poder aplicarlos en la práctica final.

1. **Referencias**

***Introduction to Ros (robot operating system)*. GeeksforGeeks. (2020, January 14). Retrieved April 21, 2022, from https://www.geeksforgeeks.org/introduction-to-ros-robot-operating-system/**