# Proyecto Robótica
# Parte I

Enrique José Padilla Terrones, *Estudiante, ITAM*

**Abstract**

En el siguiente documento se reporta el procedimiento y resultados de la implementación de un programa ejecutado en ROS, diseñado para mover un robot (simulado por TurtleSim) desde un punto inicial, a un punto final en un tiempo de ejecución especificados por el usuario.

**Index Terms**

Robótica, ROS, turtleSim, LaTeX, reporte.

## I. Introducción

La primera parte del proyecto semestral de Robótica consistió en la implementación de la plataforma ROS (introducida más adelante). Por medio de este sistema operativo, que requiere de Linux para funcionar, se utilizó el programa TurtleSim (que simula trayectorias de robots por medio de una tortuga) para introducir al alumno al entorno de programación que será utilizado a lo largo del curso. Bajo estas estructuras computacionales se diseñó un programa que permitiera al usuario del mismo definir dos variables: la posición final deseada para la tortuga y el tiempo para que la tortuga llegue al punto objetivo. Este programa debe ser capaz de ejecutarse cuantas veces el usuario ingrese coordenadas destino y tiempos de operación.

## II. Marco Teórico

ROS (Robot Operating System) es una colección de frameworks dirigido al desarrollo de software robótico. ROS proporciona la funcionalidad de un sistema operativo como manejo de paquetes, comunicación entre procesos y control de dispositivos a bajo nivel. Los procesos que ROS lleva a cabo toman la forma de gráficas para el usuario. Los procesos son ejecutados como *nodos* en donde se puede recibir y publicar información de muchos tipos, llamados *tópicos*. ROS permite a los desarrolladores crear aplicaciones para robots; para maximizar este fin, ROS es un programa OpenSource. La primera práctica requiere el conocimiento de un concepto fundamental, las trayectorias de robots. Una trayectoria es la secuencia de poses del robot en el tiempo. La trayectoria de un robot cuenta con dos tasas de cambio en un espacio bidimensional; la velocidad de desplazamiento y la velocidad angular. Estas dos variables serán las que el programa deba cambiar para moverse a la posición especificada por el usuario.

## III. Desarrollo/Experimentos

Para el desarrollo de esta practica fueron necesarios numerosos experimentos de distinta naturaleza detallados más adelante. Los experimentos pueden ser clasificados dentro de tres tipos fundamentales.

### A. Experimentos de inicio

Antes de poder escribir un programa que satisficiera las necesidades de la práctica, fueron necesarios una serie de experimentos para asegurar la correcta configuración de ROS.

*1) Instalación de ROS:* Para instalar ROS fue necesario descargar un paquete raíz de la página de internet en *http://packages .ros.org/ros/ubuntu*. Adicionalmente se debió descargar un paquete de identificación para correr el programa, la ros.key. La última descarga necesaria fue TurtleSim, que ilustra gráficamente nuestro programa através de una tortuga.

*2) Creación de paquetes:* Una vez descargados los archivos de internet, fue necesario instalarlos para crear los paquetes requeridos por ROS dentro de la máquina virtual. Dado que la máquina virtual cuenta con Ubuntu Indigo, el comando para la instalación fue el siguiente: *sudo apt-get install ros-indigo-desktop-full*. Todos los archivos generados fueron colocados bajo una carpeta específica para ROS.

*3) Ejecución del ROScore y TurtleSim:* Para iniciar las pruebas, el último requisito fue correr el ROScore, comando que inicia el master de ROS y el turtlesim node que inicia la interfaz gráfica de la tortuga.

### B. Experimentos con TurtleSim

Despues de asegurarnos de una ejecución sin errores de ROS, hicimos uso de TurtleSim para simular trayectorias de un robot. Este programa nos permite conocer y modificar la trayectoria del robot por medio de protocolos de subscripción y publicación.

*1) Pruebas de ejecución y teclado.:* Primeramente, se probó la ejecución y acción (lectura del teclado) de Turtlesim con las funciones *turtlesim node* y *teleop key*.

*2) Pruebas de publicación en TurtleSim:* Seguidamente, una prueba de publicación nos permitio corroborar la comunicación del mundo exterior (usuario) con TurtleSim.

*3) Pruebas de subscripción en Turtlesim:* Por último la prueba de subscripción, especificada con una frecuencia, permitio conocer la pose del robot a lo largo de la ejecución de un programa.

### C. Experimentos de aplicación

La última etapa de los experimentos consistió en dotarle a la tortuga la funcionalidad requerida por la práctica; la generación de una ruta entre un punto y otro,al igual que hacer a la tortuga rotar a un ángulo preestablecido por el usuario.

*1) Velocidad:* La velocidad de la tortuga es determinada por la fórmula de velocidad en un plano bidimensional.(1)

$$v* = K\sqrt{(x* - x)^2 + (y* - y)^2} \tag{1}$$

Esta fórmula fue aplicada en el método *tortugaAObjetivo* donde el resultado de la ecuación es publicado en TurtleSim. En un inicio, la trayectoria de la tortuga fue contemplada sin rotacón.

*2) Ángulo de rotación:* El ángulo de rotación es también determinado por la fórmula de ángulo en un plano bidimensional. (2)

$$\theta = \arctan \frac{y* - y}{x* - x} \tag{2}$$

El ángulo es igualmente publicado en TurtleSim una vez que es calculado, dotando a la tortuga de dirección;

*3) Experimentos finales:* Los experimentos finales consistieron en la combinación de los códigos anteriores, para que el autómata se moviera en un plano 2D al punto establecido por el usuario con éxito. La última etapa de la experimentación fue hacer que el robot llegara a la pose indicada en un tiempo definido. Para esto la relación más simple de tiempo, distancia y velocidad fue aplicada a la fórmula presentada más arriba. La relación inversa entre la velocidad y el tiempo fue incluida en el cálculo de la distancia, logrando así el movimiento en un tiempo de ejecución definido.

## IV. Conclusiones

Después de mucho trabajo, tiempo y esfuerzo, se ejecutó exitosamente el programa estipulado con anterioridad. Por medio del teclado el usuario ingresa coordenadas, ángulo y tiempo de ejecución para que la tortuga cambie su pose. Todas estas variables de usuario son requisadas en la consola antes de la ejecución del programa. Una vez que la tortuga llega a la pose deseada, el programa se vuelve a ejecutar hasta que el usuario indique con una combinación especifica que las pruebas han finalizado. El proyecto probó ser todo un reto, tanto personal como colectivo. Personalmente, se aprendió manejo de la consola en una maquina virtual al igual que el uso de LaTeXpara generar un reporte con el debido formato de la IEEE. Colectivamente hubo apoyo por medio de una comunicación eficaz vía *Piazza*. Se resolvieron dudas que iban surgiendo a lo largo del proyecto con la cooperación de cualquier integrante de la clase. La comunicación entre los alumnos será una herramnienta fundamental para las siguientes etapas del proyecto.

## Agradecimientos

El autor extiende una agradecimiento a Diego Cifuentes Jimenez y a Victoria Medina García por su dedicada ayuda, sin la cual la realización de este proyecto no hubiera podido ser posible.

## References

[1] J.M. O'Kane, *A Gentle Introduction to ROS*, Abril 2016.

[2] Koubaa A., [Anis Koubaa]. (2015, Septiembre 29). *[CS460] ROS Tutorial 4.4: Go-To-Goal Location (Turtlesim Cleaner)*. URL: *https://www.youtube.com/watch?v=Qh15Nol5htM*

**Enrique Jose Padilla Terrones** Enrique nació en Knoxville, Tennessee, EEUU el 15 de agosto de 1995. En el verano del 2013 entraria al Instituto Tecnologico Autónomo de México a estudiar Ingeniería Mecatrónica e Ingeniería Industrial. Actualmente cursa el séptimo semestre de sus estudios en la Ciudad de México.