INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO
Departamento Académico de Sistemas Digitales

Temas Selectos de Robótica: SDI-31911

# Proyecto 1: Generación de trayectorias en ROS

Materia:
Robótica: SDI-11911

Profesor:
Dr. Marco Morales

Presenta:

**Sebastián Sánchez Alcalá 133447**

Otoño 2016

# Índice

**1. Introducción** **1**

**2. Marco Teórico** **2**

2.1. ¿Qué es ROS? . . . . . 2
2.2. Poses . . . . . 2
2.3. Comunicación en ROS . . . . . 2
2.4. Conceptos matemáticos . . . . . 3

**3. Experimentos** **4**

**4. Conclusiones** **6**

# 1. Introducción

Un tema de suma relevancia en esta epóca es la robótica. Dados los grandes avances que se han alcanzado en los últimos años, es una de las ramas de la ciencia que toma más fuerza con vistas al futuro. Es por ello que cada vez es más relevante saber sobre el tema. En lo personal, creo que no hay mejor forma de entender los conceptos de esta materia que experimentando por cuenta propia, ese es el por qué de este proyecto.

Este primer Proyecto tiene como objetivo mover una tortuga representada por el simulador de ROS trutlesim. Se busca que, a partir de una pose dada por el usuario, la tortuga se desplace y adopte dicha pose en un tiempo previamente otorgado por el usuario.

# 2. Marco Teórico

## 2.1. ¿Qué es ROS?

La plataforma utilizada para este proyecto lleva como nombre Robot Operating System (ROS, por sus siglas) la cual, su definición oficial es:

*ROS es un meta-sistema operativo para tu robot de código abierto. ROS te provee los servicios que esperarías de un sistema operativo, incluyendo la abstracción de hardware, control de dispositivos en bajo nivel, implementación de funcionalidades comunmente utilizadas, comunicación de mensajes entre procesos y administración de paquetes. También te provee librerías y utilidades para obtener, construir, escribir y ejecutar código en múltiples computadoras.* [1]

A su vez, ROS cuenta con un simulador, llamado turtlesim, el cual es clave en el desarrollo de este proyecto, ya que a través de él podremos comprobar si nuestro código realmente actúa conforme los requerimientos establecidos. Este simulador presenta en pantalla una tortuga, la cual representaría al robot a programar, en una pose inicial al centro de la pantalla, a través de mensajes es posible que nosotros manipulemos dicha pose, pudiendo nosotros girar y desplazar a la tortuga.

## 2.2. Poses

A todo esto surge una pregunta: ¿Qué es una pose?

Una pose es una representación de algún objeto en un espacio, en este caso me estaré enfocando al espacio de dos dimensiones, en el cual es el que el proyecto será desarrollado y el objeto será la tortuga representada en la pantalla. Dicha pose se conforma de 3 componentes: su posición conforme al eje x, su posición relativa al eje y su orientación $\theta$, la cual nos dice hacia donde apunta la cabeza. Dicha pose se puede ver de forma absoluta, es decir, en relación al sistema original, o de forma relativa, dependiendo de su pose actual.

La tortuga representada por turtlesim se puede pensar como un objeto con propiedades, en este caso su pose, y con funciones, en este caso el desplazamiento y rotación.

## 2.3. Comunicación en ROS

Para realizar dichas funciones se utilizan mensajes, es decir, nos comunicamos con la tortuga a través de ROS. Para esto, nuestro programa toma dos roles: uno de publicador, le manda instrucciones a la tortuga para cambiar sus propiedades, y el otro como suscriptor, es decir, le pide a la tortuga información sobre su estado actual.

[1] http://wiki.ros.org/ROS/Introduction

En este proyecto utilicé el mensaje geometry_msgs/Twist para publicar a qué velocidad debe de girar o desplazarse y el mensaje turtlesim/Pose para suscribirme a las actualizaciones sobre el estado de la tortuga.

El mensaje geometry_msgs/Twist recibe dos vectores: uno de velocidad lineal, el cual indica la velocidad en cada componente (x,y,z) y el otro de velocidad angular, igualmente en cada componente.

El mensaje turtlesim/Pose publica las componentes en las que se encuentra la tortuga (x,y), su orientación ($\theta$) e igualmente los dos vectores de velocidad mencionados anteriormente.

## 2.4. Conceptos matemáticos

Para poder calcular el desplazamiento y giros necesarios para que la tortuga llegue al objetivo utilicé dos conceptos básicos de la geometría.
Para calcular la velocidad lineal a la que la tortuga debe desplazarse utilicé la fórmula $v = \frac{d}{t}$ de la cual, dado que sabemos el tiempo de ejecución de la transición y podemos calcular la distancia con la fórmula $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, una vez calculada dicha distancia, solamente nos basta sustituir los valores en la fórmula de velocidad y publicar dicha velocidad linear a través del mensaje mencionado previamente, como la tortuga decidí moverla únicamente avanza hacia el frente, publico dicho mensaje reemplazando la velocidad lineal de la componente x únicamente.

Para girar a la tortuga, se hace algo similar, solamente que se publica la velocidad angular de la componente z, ya que el giro se realizará sobre ese eje. Para calcular cuánto debe girar utilicé la fórmula: $\theta' = tan^{-1}(\frac{y'-y}{x'-x})$.

# 3. Experimentos

Durante la elaboración del proyecto se realizaron muchos experimentos, casi al punto de estar programando en un método de prueba y error.

Al principio me fue muy dificil comprender los conceptos que se emplean en ROS y turtlesim, tuve que hacer varias pruebas para ver en qué unidades se describen las velocidades, después de algunas pruebas descubrí que la velocidad lineal está dada en unidades de distancia por segundo y la velocidad angular está dada en radianes por segundo.
una vez encontradas dichas unidades, me simplificó mucho los cálculos, sobre todo para el movimiento lineal, pero los problemas para la velocidad angular no pararon. La tortuga parecía no importarle las restricciones que le daba en cuanto a cuándo debá de parar, después de horas y horas de nuevamente estar a prueba y error, descubrí que el problema se debía a que hay un retraso en cuanto a la obtención de la pose actual de la tortuga, debido a que la velocidad que le estaba asignando era muy grande. Para solucionar este problema decidí que el giro final debía de hacerlo en un tiempo arbitrario definido por mi, decidí que un tiempo considerable era un segundo. Dicha restricción me modificó algunas variables en cuanto a mi movimiento lineal.

Para el desplazamiento de un punto en el espacio a otro, la estrategia fue difernte: aquí decidí no darle importancia a la velocidad angular que debía darle, solamente me enfoqué en la velocidad lineal. Como consecuencia, la velocidad angular que le mando a la tortuga es un tanto arbitraria, la definí en 4 veces la distancia entre el ángulo actual y el ángulo al cual debía de orientarse para llegar al punto final. Para la velocidad actual adapté la fórmula mencionada en el marco teórico para la final usar la siguiente: $velocidad = \frac{distancia\ actual\ al\ punto\ destino}{tiempo\ dado\ -\ 1\ segundo\ -\ (tiempo\ actual\ -\ tiempoinicio)}$. Aún con esos ajustes, la tortuga no llegaba al punto de destino, por lo que le otorgué una tolerancia arbitraria de 0.1 unidades de distancia.

Desgraciadamente, mientras luchaba contra ROS y rutlesim, nunca me acordé de documentar los resultados de los experimentos fallidos, algo que no pasará en proyectos posteriores.

Por otro lado, sí documenté los resultados finales, como lo muestra la imagen siguiente:

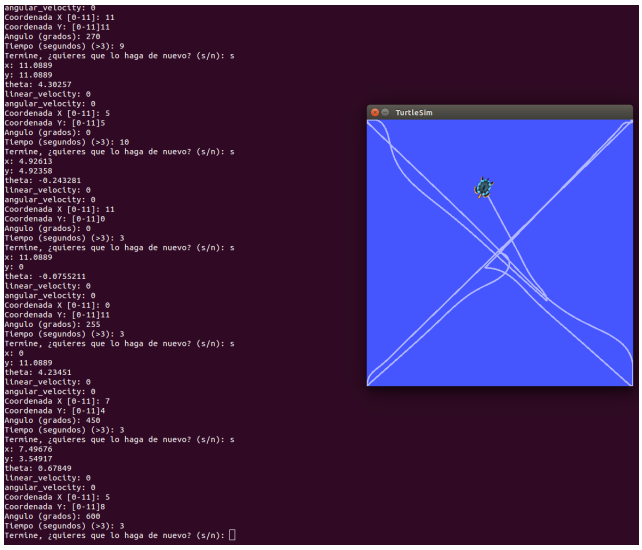


Figura 1: Resultados de instrucciones mandadas a la tortuga

También debo mencionar que dado los problemas de velocidad arriba mencionados, la decisión de otrogarle un segundo solamente al giro, me restringió también el tiempo mínimo de ejecución, el cual tuve que fijar en 3 segundos, ya que al relaizar pruebas, en caso de ingresar un valor menor a este, perdía todo control sobre la tortuga.

# 4. Conclusiones

A mi parecer el proyecto se concluyó satisfactoriamente ya que ROS era una plataforma completamente desconocida para mí y nunca es fácil comenzar un proyecto así desde cero.
También debo mencionar que en algún punto subestimé la dificultad del proyecto, ya que al leer los requerimientos y objetivos me pareció muy fÃ¡cil, pero son temas mucho más complejos de lo que aparentan.

A pesar de no obtener resultados 100 % exactos y de tener algunos casos de falla, considero que es satisfactorio, ya que es completamente normal que un proyecto elaborado con esta premura y, repito, sin tener todos los conceptos tan claros o frescos, se obtengan este tipo de resultados.
Finalmente, creo que el proyecto me ayudó mucho a refrescar algunos temas que no tenía tan claros y, por supuesto, a aprender un poco del mundo de la robótica, además de haber conocido ROS y la forma en la que trabaja. Sin mencionar que me ayudó a mejorar mis habilidades en control de errores y toma de desiciones para establecer restricciones a proyectos de esta índole.

# Referencias

[1] Jason M. O'Kane. *A Gentle Introduction to ROS.* Independently published, 2013. Available at http://www.cse.sc.edu/~jokane/agitr/..