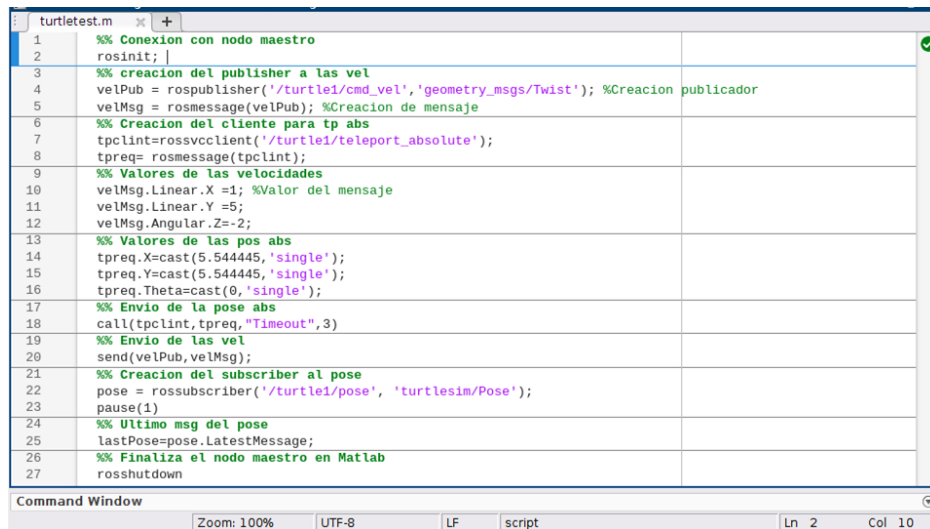


Metodología

El objetivo principal es desarrollar un script en Matlab y un código Python para controlar una simulación de tortuga en el entorno de ROS. Para lograrlo, se utilizó ubuntu 20,04 lts, ROS Noetic, MATLAB con el Robotics toolbox de matworks, visual studio para la programación en Python y el módulo rospy para hacer la conexión.

1. Desarrollo en Matlab



```
1 %% Conexion con nodo maestro
2 rosininit;
3 %% creacion del publisher a las vel
4 velPub = rospublisher('/turtle1/cmd_vel','geometry_msgs/Twist'); %Creacion publicador
5 velMsg = rosmesssage(velPub); %Creacion de mensaje
6 %% Creacion del cliente para tp abs
7 tpclint=rossvcclient('/turtle1/teleport_absolute');
8 tpreq= rosmesssage(tpclint);
9 %% Valores de las velocidades
10 velMsg.Linear.X =1; %Valor del mensaje
11 velMsg.Linear.Y =5;
12 velMsg.Angular.Z=-2;
13 %% Valores de las pos abs
14 tpreq.X=cast(5.544445,'single');
15 tpreq.Y=cast(5.544445,'single');
16 tpreq.Theta=cast(0,'single');
17 %% Envio de la pose abs
18 call(tpclint,tpreq,'Timeout',3)
19 %% Envio de las vel
20 send(velPub,velMsg);
21 %% Creacion del subscriber al pose
22 pose = rossubscriber('/turtle1/pose', 'turtlesim/Pose');
23 pause(1)
24 %% Ultimo msg del pose
25 lastPose=pose.LatestMessage;
26 %% Finaliza el nodo maestro en Matlab
27 rosshutdown
```

Figura 1. Script de Matlab. Anexo en

https://github.com/Ggio0/Ros_Lab3/blob/main/Entregables/matlab/turtletest.m

Para la primera parte se busca realizar el control de la tortuga a partir de un script en MATLAB, donde primero se establece la conexión con el nodo maestro de ROS utilizando rosininit, lo que permite la interacción con otros nodos y servicios en el entorno ROS, a continuación, se crea un publicador (velPub) que envía comandos de velocidad a la tortuga en la simulación. Estos comandos se publican en el tópico /turtle1/cmd_vel, y se utiliza el mensaje geometry_msgs/Twist para describir la velocidad de la tortuga. Además, se configura un cliente (tpclint) para el servicio de teletransportación absoluta en ROS, permitiendo teletransportar la tortuga a una ubicación específica. También establece los valores de velocidad y posición absoluta en los mensajes correspondientes (velMsg y tpreq). Luego, se envían comandos de teletransportación y velocidad, lo que controla el movimiento de la tortuga en la simulación y la teletransporta a la ubicación especificada.

Se crea un suscriptor (pose) para recibir datos de posición de la tortuga desde el tópico /turtle1/pose. Esto permite obtener información sobre la posición actual de la tortuga en la simulación. El código pausa durante un segundo y luego obtiene el último mensaje de posición de la tortuga. Por último, se cierra la conexión con el nodo maestro de ROS utilizando rosshutdown.

2. Desarrollo en Python

Para la segunda parte además de realizar el control de la tortuga, se busca que responda de acuerdo con la tecla presionada por el usuario, por lo que se realizó un script en Python y la biblioteca de ROS y Python para llevar a cabo sus funciones, las cuales se listan a continuación.

- `pubVel(vel_x, vel_y, ang_z, t, nt)`: Publica comandos de velocidad que controlan el movimiento de la tortuga en la simulación. Con parámetros como las velocidades lineales y angulares, así como la duración y subdivisión del tiempo.
- `getkey()`: Captura las pulsaciones de teclas del usuario, lo que posibilita el control en tiempo real de la tortuga mediante el teclado.
- `teleport(x, y, ang)` utiliza un servicio de teletransportación para cambiar la posición de la tortuga en la simulación a una ubicación absoluta específica. Con parámetros como las posiciones en coordenadas absolutas y el ángulo respecto a la horizontal.
- `run()`: La función principal del programa, la cual inicia un bucle que escucha las pulsaciones de teclas del usuario. Dependiendo de la tecla presionada, realiza diversas acciones como mover la tortuga hacia adelante o atrás, con `w` y `s` respectivamente, girarla en sentido horario o antihorario, con `a` y `d`, teletransportarla a una posición fija, con `r`, o girarla 90 grados, con la barra espaciadora. El bucle continúa hasta que se presiona la tecla "Esc".

Dado que este código es más extenso, solo se muestra como anexo en

https://github.com/Ggio0/Ros_Lab3/blob/main/Entregables/Python/myTeleopKey.py.

Resultados

1. Resultados en Matlab

Al compilar el script desarrollado en Matlab, primero la tortuga comienza en una posición inicial, la cual es ajustable según las variables correspondientes. La posición inicial de la tortuga se muestra en la figura 3.

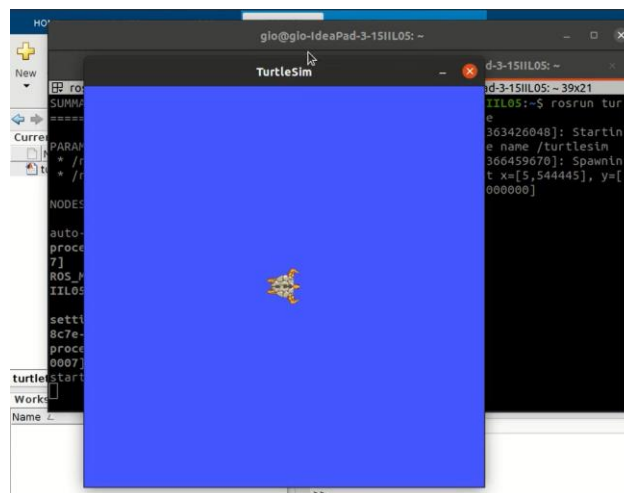


Figura 3. Interfaz de la tortuga en su posición inicial.

Al ejecutar el script con unas velocidades $V_x = 2$, $V_y = 6$, $w_{ang} = 3$, la tortuga se desplaza describiendo la trayectoria mostrada en la figura 4.

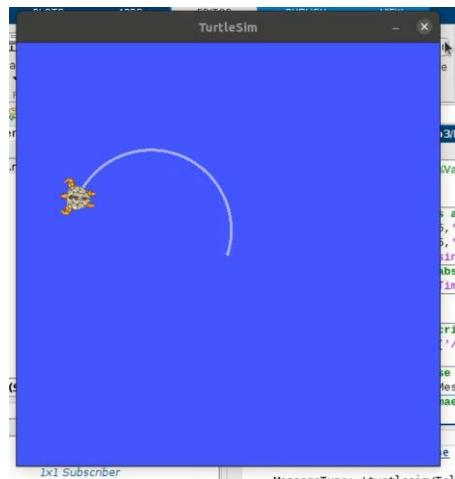


Figura 4. Trayectoria descrita con los parámetros seleccionados.

Si se cambian las velocidades a $V_x =$, la tortuga describe la trayectoria mostrada en la figura 5.



Figura 5. Nueva trayectoria descrita con los parámetros seleccionados.

Adicionalmente, la variable "lastPose" guarda la pose de la tortuga después de terminar el script.

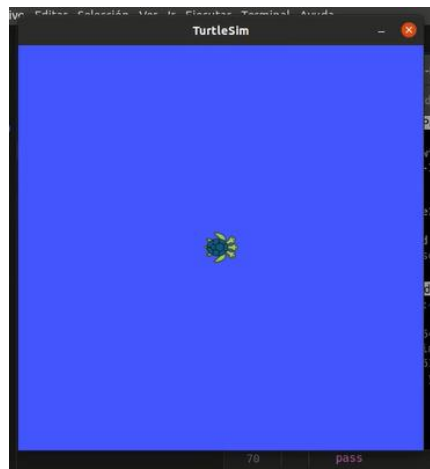
Este procedimiento se ve más a detalle en el video anexo en

https://github.com/Ggio0/Ros_Lab3/blob/main/Entregables/Videos/Matlab_Ros.mp4

.

2. Resultados en Python

Al compilar el código creado, se inicializa la interfaz con la tortuga en su pose inicial como se puede observar en la figura 6.



Ahora, se presionan las teclas para mover la tortuga, mientras que en la terminal se muestra la tecla presionada por el usuario, en la interfaz la tortuga se mueve describiendo una trayectoria. Esto se puede observar en la figura 7.

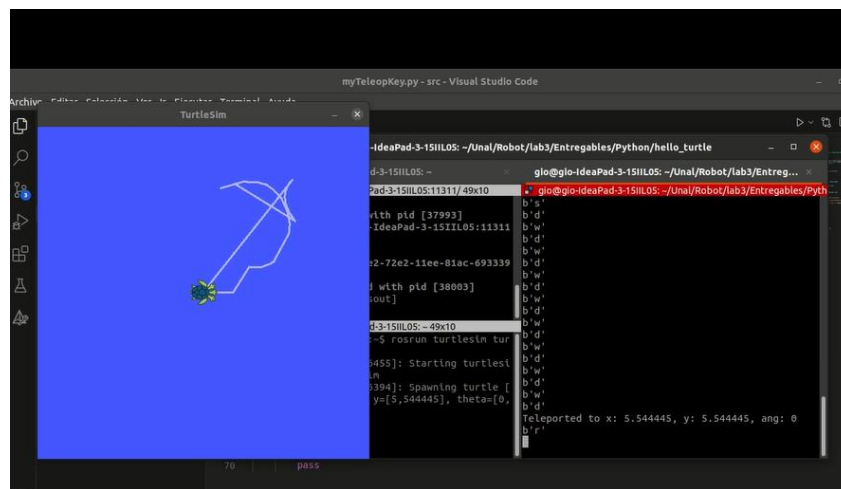


Figura 7. Trayectoria de la tortuga descrita según la secuencia de teclas presionadas mostradas en el terminal.

Se observa que la tortuga se mueve como es deseado para todas las teclas y combinaciones. **Este procedimiento se ve más a detalle en el video anexo en** https://github.com/Ggio0/Ros_Lab3/blob/main/Entregables/Videos/Python_Ros.mp4

Análisis

Esta práctica puede ser vista como una analogía al manejo del efector final de un robot en un plano, puesto que en el caso de Matlab, se le da una velocidad en unas direcciones durante un tiempo y la tortuga describe una trayectoria hasta llegar a una pose final (como lo haría el efector final de un robot), en un caso de la realidad, se le podrían ingresar perfiles de velocidad diseñados según una trayectoria deseada y mediante un paso intermedio como el Jacobiano se podría hallar las velocidades de la articulación. Sin embargo, esta primera práctica es solo un inicio, por lo que no se toman en cuenta

estas consideraciones, en este caso es como si se estuviera moviendo sin pensar en las articulaciones. Por otra parte, para el código de Python, se observa una analogía con el movimiento manual de las articulaciones de un robot, las teclas ASDWR serían el movimiento de las articulaciones como se hacía con el stick del FlexPendant en el laboratorio, así mismo, la tecla ESPACIO ejecutaría una rutina de movimiento predefinida que en este caso es el giro de 180°. En general, se observa que esta práctica es una introducción al manejo de robots desde su sistema operativo, mientras que en las anteriores prácticas solo se enfocaba en la generación de trayectorias, puntos y todo lo que está “por fuera” del robot, esta práctica está más centrada hacia la programación y lógica interna que realiza un robot bajo ciertas acciones solicitadas con el fin de entender e interpretar el funcionamiento de la conexión entre software y hardware del robot.

Conclusiones

De esta práctica se concluye:

1. ROS permite controlar y crear funciones predefinidas para el movimiento bajo ciertos parámetros seleccionados. Se observa en esta práctica el tratamiento de datos que le hace a los parámetros de entrada como velocidad para producir una salida deseada.
2. Se observa que esta lógica se puede escribir tanto en Python como en Matlab, por lo que se concluye que la programación de un robot se puede realizar en estos dos entornos, sin embargo, una programación en Python puede llegar a ser más versátil porque puede editarse en más entornos.
3. La decisión de utilizar software de código abierto como Ubuntu 20.04 LTS y ROS Noetic en el proyecto de robótica ofrece una ventaja debido a la facilidad de acceso a estas herramientas. Sin embargo, debemos ser conscientes del riesgo de incompatibilidades con futuras versiones de software, ya que las tecnologías avanzan y las versiones anteriores pueden volverse obsoletas.