



Primer acercamiento a Webots

¿Qué es Webots?

Webots es una potente plataforma profesional de simulación robótica, gratuita y de código abierto, utilizada tanto en la industria como en la investigación. Te permite crear entornos 3D detallados y replicar propiedades físicas complejas como masa, inercia y fricción para una gran variedad de robots, ya sean con ruedas, piernas o aéreos. Lo que lo hace especialmente poderoso es su capacidad para programar el comportamiento de los robots usando lenguajes populares como Python, C++ o MATLAB. Cuenta con una extensa biblioteca de sensores (cámaras, LiDAR, GPS), actuadores y modelos de robots comerciales listos para usar. Uno de sus mayores beneficios es la facilidad para transferir el código probado en la simulación a un robot físico real, lo que lo convierte en una herramienta ideal para la educación, la investigación y la participación en competiciones internacionales como la RoboCup.

¿Qué compone una simulación de Webots?

Esta compuesta por cuatro elementos: el entorno de simulación, modelos robóticos, el controlador y el motor de físicas

- **Entorno de Simulación:** Define el escenario físico: suelos, paredes, obstáculos; condiciones ambientales, posición inicial de los robots, entre otras cosas. Esto puede ser editado mediante el "Scene Tree", el cual lista todos los objetos. El escenario es un archivo de texto con la extensión .wbt
- **Modelos Róboticos:** Son los individuos del mundo creado, más allá de ser un modelo 3D, son un conjunto de nodos que definen las capacidades propias del robot. Se encuentra compuesto por las siguientes partes:
 - **Cuerpo:** son las partes físicas del robot.
 - **Actuadores:** son los componentes que le permiten moverse e interactuar.
 - **Sensores:** Son los componentes que le permiten al robot percibir su entorno.
- **Motor de Físicas:** Es la parte que se asegura de que el mundo actúe según las leyes de la física. Webots utiliza una versión modificada del Open Dynamics Engine. Se conforma de:
 - **Responsabilidades:** Se encarga de la simulación siga las físicas.
 - **Detección de Colisiones:** Identifica cuándo dos objetos se tocan.
 - **Dinámica de Cuerpos Rígidos:** Calcula cómo las fuerzas y los torques afectan el movimiento y rotación de los objetos.
 - **Fricción:** Simula la fricción estática y cinética entre las superficies.

Una vez desmenuzado y explicado como funciona y que compone el software, se puede empezar a interactuar con el sistema de simulaciones Webots. Para más información se puede visitar la página oficial de Webots: <https://cyberbotics.com/doc/guide/introduction-to-webots>

Interfaz de Webots

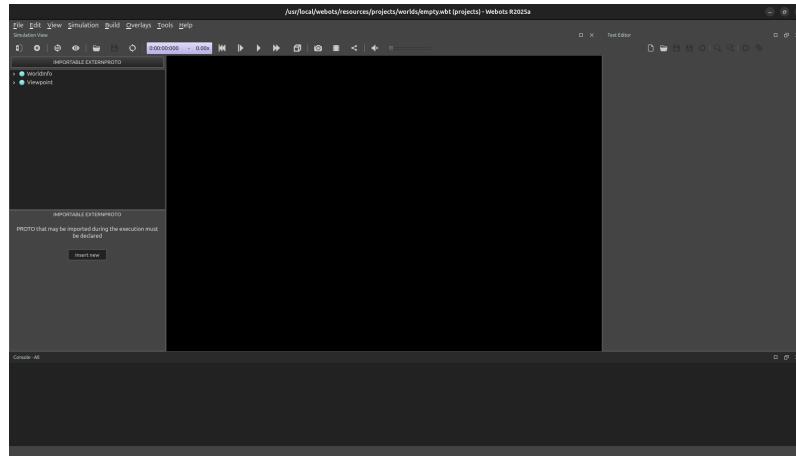


Figura 1: Interfaz

La interfaz gráfica esta compuesta de:

- **3D window:** muestra lo que sucede en la simulación.
- **Scene Tree:** representación jerarquica de la simulación.
- **Text Editor:** permite editar el código fuente.
- **Console:** Muestra la compilación y el controlador

En la barra del menú tenemos: **File, Edit, View, Simulation, Build, Overlays, Tools y Help**

Para tener una mejor comprensión del entorno de Webots, comenzemos abriendo un ejemplo de los que vienen predeterminados por el programa.

En nuestra terminal ejecutaremos la siguiente linea de comando:

```
cd /usr/local/webots/projects/samples/tutorials/worlds/
```

Esto desplegará los ejemplos disponibles del programa. En nuestra terminal ejecutaremos: `webots my_first_simulation`. Después se abrirá la siguiente ventana y veremos la siguiente simulación.

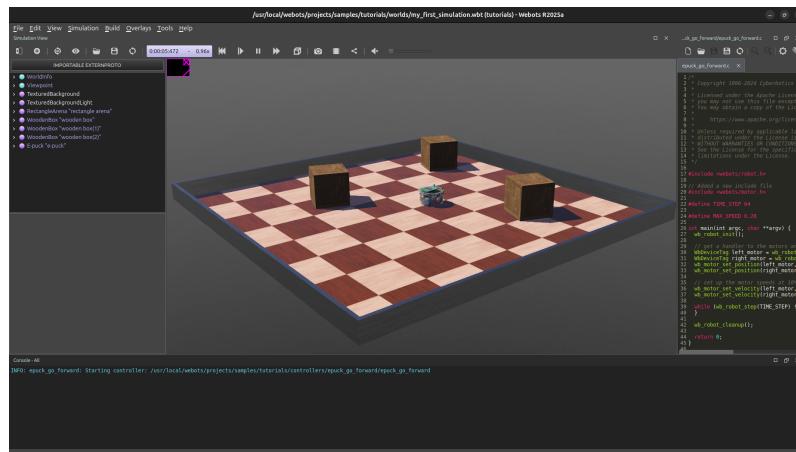


Figura 2: Ejemplo

La simulación desplegada mostrará un E-puck que solo avanzará en línea recta hasta chocar con la pared. Si hacemos click zquierdo sobre el robot, se pueden observar las líneas que delimitan su forma, vendrían siendo su contorno geométrico así como los tres ejes (x, y, z) y sus respectivas rotaciones (roll, pitch, yaw).

Figura 3: Contorno Geométrico

Si seleccionamos alguna de estas, se puede modificar la posición o alguno del objeto (solo si esta pausada la simulación) se mostrara una leyenda que indicará su posición o ángulo con respecto a sus offsets. También pueden usarse:

- **Shift + flecha hacia arriba** y click derecho para mover el objeto
- **Shift + flecha hacia arriba** y click izquierdo para rotar el objeto

Otras interacciones que pueden tenerse con la 3D window, son la aplicación de efectos físicos: como el **torque y fuerza**

- **Torque: Ctrl + Alt** y click derecho, se observará un vector amarillo el cuál mostrara un marcador del torque aplicado al obejto.
- **Fuerza: Ctrl + Alt** y click izquierdo, se observará un vector naranja el cuál mostrara un marcador de la fuerza aplicada al obejto.

Ambas varían su magnitud en función de la longitud del vector.

Hacer click derecho sobre el objeto despliega una ventana que muestra diferentes acciones disponibles para el mismo. Se recomienda probar cada una para familiarizarse con las acciones que hacen, si algo desaparece o cambia, solo refresca la simulación sin guardar. Las últimas 6 opciones, permiten acceder al código que compone el comportamiento del robot así como del mundo construido.

Figura 4: Ventana de acciones

Para poder observar la camara u otros sensores disponibles, seleccione la opción de **Overlays** y se mostrarán los que están disponibles para el robot en uso. Para el E-puck, seleccione **Show 'camera' overlay** y se desplegará una ventana negra en la parte superior de la simulación. Para poder inciarla seleccione la opción **Show Robot window** desplegándose una ventana como esta.

Seleccione el botón de **Enable All**, seleccione **Refresh** en Simultaion y Upload HEX.. ¿Qué cambios hubo en la simulación?

En nuestra terminal ejecutaremos: **Crtl+ C** y escribiremos **webots 4_wheels_robot.wbt**
La silumación desplegada es de un carro que evita chocar con las paredes.

Del lado derecho se encuentran la sección **Scene Tree y Editor Field**

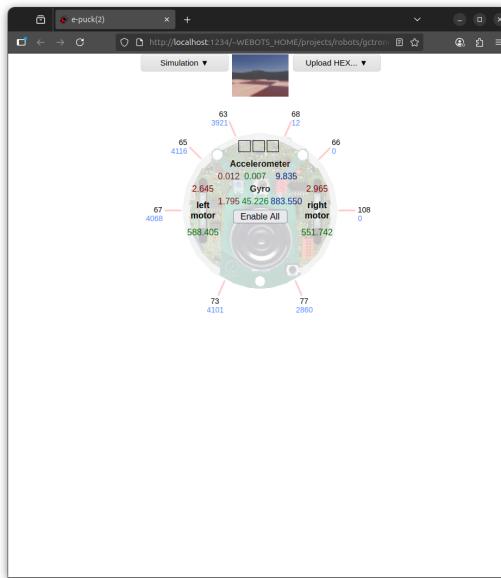


Figura 5: Robot window

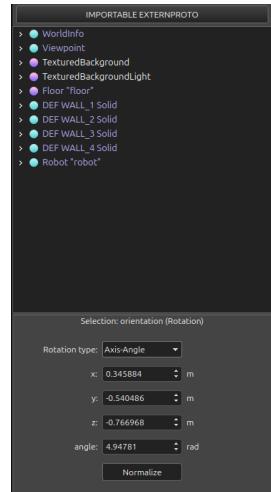


Figura 6: Barra lateral

El **Scene Tree** es una lista organizada que muestra todos los elementos que existen en la simulación (robots, luces, obstáculos, suelo, etc.) y como estos se relacionan entre si.

- **Estructura Jerárquica:** Los objetos se encuentran anidados unos dentro de otros.
- **Organización y Navegación:** Permite encontrar y seleccionar rápidamente cualquier objeto en la escena, sin importar si es visible o no en la ventana 3D
- **Contenedor Global:** Ese nodo es especial, ya que maneja las reglas globales que afectan a todo el mundo, como la gravedad o la configuración de la física.

Al seleccionar uno de los objetos que hay en el **Scene Tree**, se activa el **Field Editor** y funciona como la hoja de propiedades o el panel de control de ese objeto específico.

- **Visualización de Atributos:** Muestra todas las características y parámetros del objeto seleccionado.

- **Modificación de Propiedades:** Permite cambiar los valores de estos campos. Se puede ajustar números, seleccionar opciones de menús desplegables o escribir texto para configurar el comportamiento y la apariencia del objeto.

Seleccione el nodo Robot, se desplegará todos los nodos relacionados con el robot. Seleccione **children** y luego **DEF BODY Shape** mostrando los nodos ligados a la parte física del robot. Seleccione **geometry Box** y **size** esto activará el **Editor Field** a continuación modifique el aspecto físico del robot. Para deshacer los cambios solo reinice la simulación.

Así como se puede interactuar con el aspecto, también se puede modificar el comportamiento de los sensores y actuadores del robot.

Dentro de **DEF DS_RIGHT DistanceSensor** busqué **lookupTable** y coloque los valores a 0 y **DEF DS_LEFT DistanceSensor** eliminelo. **lookupTable** traduce la distancia que mide el sensor a un valor numérico que entiende el robot. Por ejemplo: "si mido 0.1 metros, devuelve el valor 1000". entonces al colocarlos en 0 el sensor no sabrá qué valor devolver, por lo que siempre enviará un 0. Para el controlador del robot, un 0 significa "no hay obstáculo", por lo que el sensor queda efectivamente desactivado.

Eliminar **DEF DS_LEFT DistanceSensor** quita por completo el sensor izquierdo. Lo que genera que el robot se quedé dando vueltas ya que entra en un bucle de rotación.

Reinic peace la simulación y modifique **lookupTable** para que la distancia de detección aumente y en otra disminuya.

ROVER

Un *rover* o astromóvil es un vehículo de exploración diseñado para desplazarse sobre la superficie de un planeta u objeto astronómico. Estos dispositivos son empleados en misiones espaciales para recopilar datos e imágenes, así como para realizar análisis de muestras de suelo, rocas y atmósfera. Su uso resulta fundamental para la exploración remota de cuerpos celestes como la Luna, Marte y otros planetas o asteroides de interés científico.

Los objetivos y características de un astromóvil pueden variar según la misión o el entorno en el que operará. Entre los principales se encuentran:

- Investigación científica
- Búsqueda de signos de vida
- Análisis geológico
- Búsqueda y aprovechamiento de recursos

Además, los rovers suelen incorporar sistemas avanzados que les permiten operar de forma autónoma o semiautónoma. Entre sus principales subsistemas destacan: la movilidad, el sistema de energía (paneles solares o baterías), las comunicaciones, los instrumentos científicos y los sistemas de navegación y control. Al igual que los vehículos terrestres, los rovers pueden clasificarse en distintos tipos según su aplicación y el entorno donde trabajen:

- Rovers planetarios
- Rovers lunares
- Rovers para asteroides
- Rovers de exploración submarina o terrestre



Figura 7: Rover de exploración marciana.



Figura 8: Prototipos de rover para pruebas terrestres.

Para las simulaciones y posteriormente las prácticas físicas, se empleará un modelo de rover simplificado que permite comprender los principios de locomoción, control y navegación autónoma. Este tipo de modelos facilita el diseño e implementación de algoritmos de control mediante software como **Webots** o entornos de desarrollo con **ROS 2**, donde es posible recrear sensores virtuales, cámaras, motores y terrenos de prueba realistas.

En la siguiente figura se muestra el modelo utilizado para la simulación, el cual sirve como base para experimentar con comandos de movimiento, adquisición de datos y validación de estrategias de control antes de su implementación en un sistema físico.

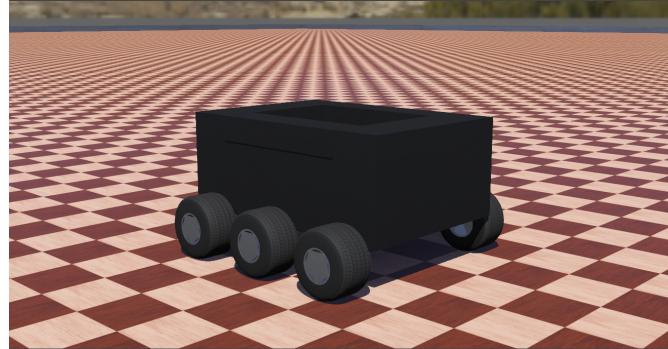


Figura 9: Modelo de rover empleado para simulación en Webots.

Para probar esta simulación faremos lo siguiente:

```
cd PAPIME_PE103825/ros2_ws/
colcon buildsource
install/setup.bash
ros2 launch rover_control rover.launch.py
```

La simulación consiste en el rover mostrado anteriormente, desplazándose en línea recta a una velocidad de 5 m/s.

Para modificar la velocidad, así como el angulo de desplazamiento, ejecute el siguiente comando en la terminal.

```
cd src/hardware/rover_control/rover_control/
```

Y ejecute el script de python: **rover_control.py**. En el modificará las siguientes líneas.

```
msg.linear.x = 0.0
msg.angular.z = 0.5
```

Figura 10: Parámetros

El parámetro **linear.x** controla la velocidad angular (en radianes por segundo) , mientras que **angular.z** controla la velocidad lineal (en metros por segundo).

Por ejemplo:

- **linear.x = 0.0, angular.z = 0.5** → movimiento recto hacia adelante.
- **linear.x = 1.0, angular.z = 0.0** → giro sobre su propio eje hacia la izquierda.
- **linear.x = -1.0, angular.z = 0.0** → giro sobre su propio eje hacia la derecha.
- **linear.x = 0.5, angular.z = 0.5** → avance con giro leve a la izquierda.
- **linear.x = -0.1, angular.z = 0.5** → avance con giro leve a la derecha.

Si únicamente modifica los valores dentro del script `rover_control.py`, basta con guardar los cambios y volver a lanzar la simulación en Webots. Sin embargo, si modifica la estructura del paquete (por ejemplo, el archivo `setup.py`, `launch` o el URDF), deberá ejecutar nuevamente:

```
colcon build  
source install/setup.bash
```

para aplicar los cambios correctamente.

Conclusión

El uso de Webots en conjunto con ROS 2 permite integrar la simulación de robots móviles dentro de un entorno físico virtual controlado mediante nodos y tópicos, reproduciendo de manera precisa el comportamiento real de sistemas robóticos. A través de esta práctica se comprendió cómo un plugin de Webots puede conectarse con ROS 2 para recibir comandos de movimiento mediante mensajes del tipo **geometry_msgs/Twist**.

El mensaje Twist representa las velocidades lineales y angulares de un robot, sirviendo como interfaz estándar entre los algoritmos de control y los actuadores simulados o reales. De esta forma, modificar los valores de `linear.x` o `angular.z` permite definir trayectorias, giros o desplazamientos en línea recta, replicando el mismo principio que se usa en robots físicos.

Esta integración demuestra la potencia de ROS 2 para el desarrollo modular y escalable de robots, y de Webots como entorno de validación antes de llevar los sistemas al hardware real. En conjunto, ambas herramientas permiten diseñar, probar y optimizar algoritmos de navegación, control de velocidad y planificación de movimiento de manera segura y eficiente.