Guía de usuario

1 Bienvenida

Bienvenido al *modelo computacional de simulación del sistema robótico Lego Mindstorms EV3*. Este modelo computacional tiene todos los elementos necesarios para poder simular de manera fiable cualquier comportamiento que se desee implementar en la plataforma robótica real. Con dicho desarrollo, es ahora fácil desarrollar las prácticas de la asignatura de robótica sin la necesidad de utilizar el Lego Mindstorms EV3 en físico. Y se abren muchos caminos de alcance educativo e investigativo.

En la presente guía se pueden encontrar tres herramientas para el usuario:

- Un manual de instalación de las herramientas software necesarias para la utilización y aprovechamiento del modelo computacional.
- Un manual de uso básico de los diferentes elementos del modelo computacional que provea el conocimiento necesario para el manejo del mismo en breve tiempo.
- Ejemplos sencillos sobre el comportamiento del sistema robótico en donde se detallan los pormenores de lectura de sensores y envío de órdenes a los actuadores de la configuración del kit.

Además, el usuario encontrará en los códigos de programación disponibles, explicaciones y comentarios a más bajo nivel con el fin de proveer una comprensión de los ejemplos y uso de la simulación más profunda.

2 Manual de instalación

2.1 Requerimientos software y hardware

A continuación, en la Tabla 1 se muestran las especificaciones computacionales recomendadas para utilizar el modelo computacional de simulación del Lego Mindstorms EV3.

Especificación	Recomendación para el modelo computacional
OS	Ubuntu 16.04 LTS
CPU	Intel Core i5 x4 (o semejante)
GPU	Tarjeta gráfica dedicada de 1 GB
RAM	6 GB
Disco Duro	1 GB de espacio libre

Tabla 1. Requerimientos computacionales para el modelo computacional de simulación para el sistema robótico Lego Mindstorms EV3. Fuente: Elaboración propia.

2.2 Instrucciones de instalación

Para el aprovechamiento del modelo computacional es necesario instalar las siguientes herramientas:

- ROS Kinetic Kame
- Gazebo 7.x
- lmev package

El proceso de instalación es muy sencillo. Para instalar ROS Kinetic Kame, se deben seguir las instrucciones del siguiente enlace.

http://wiki.ros.org/kinetic/Installation/Ubuntu

Al seguir dichas instrucciones, se instalará ROS y a la vez la versión más reciente de Gazebo 7. Adicionalmente, se debe instalar el paquete “gazebo_ros” siguiente las instrucciones “Pre-Built Debians” del siguiente enlace.

http://gazebosim.org/tutorials?tut=ros_installing

Finalmente, cabe nada más descargar la carpeta “lmev” puesta a disposición por el profesor y colocarla en el directorio “/home”.

3 Manual de uso

Antes de iniciar con lo que respecta propiamente con el modelo computacional en cuestión, es recomendable que el usuario cuente con conocimientos previos de las herramientas que se utilizan en la simulación. Para esto, se deben seguir los siguientes tutoriales. Al seguir dichos tutoriales, se garantiza que el usuario este en la capacidad de entender y manejar el modelo computacional de simulación a un nivel básico pero suficiente.

- Contar con un conocimiento básico del sistema operativo Linux. Con lo que respecta a comandos y manejo desde terminales. Se puede encontrar un tutorial básico en el siguiente enlace. No es necesario realizarlo todo, pero es una buena guía de futura referencia.
  http://www.ee.surrey.ac.uk/Teaching/Unix/

- Contar con un conocimiento básico del funcionamiento del software Gazebo. Con lo que respecta a manejo de interfaz y las plataformas robóticas. Se puede encontrar un tutorial básico en el siguiente enlace. Es necesario realizar los primeros tres tutoriales de la categoría de principiantes y recomendable realizar hasta el sexto tutorial para intermedios.
  http://gazebosim.org/tutorials

- Contar con un conocimiento básico del funcionamiento del software ROS. Con lo que respecta a sus conceptos básicos y comandos. Se puede encontrar un tutorial básico en el siguiente enlace. Es necesario realizar el tutorial 4, 5, 6 y 12 de la categoría de principiantes y recomendable realizar del 1 al 13 de la misma categoría.
  http://wiki.ros.org/ROS/Tutorials

3.1 Inicialización de la simulación

Una vez la carpeta “lmev” contenida en el directorio “/home”, se debe abrir una terminal (Ctrl+Alt+T). En dicha terminal, se debe correr una única vez el siguiente comando:

```
echo "source ~/lmev/devel/setup.bash" >> ~/.bashrc
```

Dicho comando sirve para habilitar el paquete ROS en cualquier futura terminal que se abra. Posteriormente, para inicializar la simulación se deben ejecutar los siguientes comandos:

```
cd lmev
./run_lmev.sh
```

Se debe recordar que si se presiona TAB se autocompletan los comandos en la terminal. Al correr este último comando, se abrirá una nueva terminal con tres pestañas, se inicializará Gazebo en pausa, se cargara la plataforma robótica en el mundo de simulación y se abrirá una pequeña ventana que muestra la visualización de la cámara.
La primer pestaña de la nueva terminal es la responsable de ejecutar roscore y por ende de inicializar todos los elementos ROS necesarios para la simulación, incluyendo el ROS Master. Además, da una pequeña bienvenida en la que se encuentra información sobre el paquete desarrollado. La segunda pestaña de la nueva terminal es la responsable de iniciar la visualización de la cámara que posee la plataforma robótica. El sensor de color es simulado mediante una cámara, por lo que es imperativo mantener tanto esta pestaña como la ventana abierta en todo momento que se utilice el modelo computacional. La tercera pestaña de la nueva terminal muestra en pantalla los valores actuales de todos los sensores presentes en la plataforma robótica simulada. Solo se actualizarán los valores mientras la simulación en Gazebo no esté en pausa y es solo con el fin de visualización. Más adelante se retomará esta pestaña.

Con lo que respecta a la ventana de Gazebo inicializada, se muestra una interfaz gráfica de usuario común para este software y un entorno de simulación básico diseñado. Al igual que la plataforma robótica simulada en el centro del mundo de simulación, lista para su uso.

3.2 Control de la simulación

Para el control del comportamiento de la plataforma robótica, existen dos acercamientos. El primer acercamiento es mediante un control manual por terminal. Este consiste en enviar y visualizar datos de la plataforma robótica en una terminal para mayormente familiarizarse con las herramientas de la simulación o para pequeñas pruebas de debug. El segundo

acercamiento es mediante un control programático por un nodo ejecutable de Python. Este último es especialmente útil para cuando se deseen diseñar comportamientos complejos y que posteriormente se deseen trasladar a la plataforma robótica real.

3.2.1 Control por terminal

Inicialice la simulación

```
cd lmev
./run_lmev.sh
```

En la interfaz gráfica de Gazebo, presione el botón de “Play”, esto habilitará el movimiento de la plataforma robótica, y el funcionamiento de los sensores – los cuales puede comprobar en la tercer pestaña de la nueva terminal. A partir de este momento, Gazebo está utilizando sus motores de cálculo y gráficos para calcular y mostrar dinámicamente un comportamiento de la plataforma robótica fiel a su semejante real. Presione ahora el botón de “Pausa” para tratar primero con algunos otros aspectos de la simulación.

En general, a nivel de usuario el control que se desea es poder realizar una lectura de las mediciones de los sensores y un poder enviar órdenes a los actuadores de la plataforma robótica en simulación. Este proceso se lleva a cabo mediante la comunicación y uso de nodos y topics en el entorno ROS. El usuario, al haber realizado los tutoriales propuestos, debe estar familiarizado con estos conceptos y comandos de ROS, al igual que sus principios de funcionamiento.

En la terminal ejecute el siguiente comando.

```
rostopic list
```

Se enlistará, una serie de topics activos que ROS Master está manejando. De los cuales importan los siguientes:

- /Arm_Motor/vel_cmd
- /Color_Sensor/Ambient
- /Encoder_Arm_Sensor

- /Encoder_Left_Wheel_Sensor
- /Encoder_Right_Wheel_Sensor
- /Gyroscope_Sensor
- /Left_Wheel_Motor/vel_cmd
- /Right_Wheel_Motor/vel_cmd
- /Touch_Sensor
- /Ultrasonic_Sensor
- /cmd_vel

Cada topic de estos maneja información que recibe o envía de un nodo de procesamiento ROS. Aquellos topics que contengan en su nombre la palabra “Sensor” son resultado de un nodo publicador, el cual constantemente envía mensajes por medio de ellos. Mientras que aquellos topics que contengan en su nombre la palabra “cmd” son topics que al recibir un mensaje, envían dicha información a un nodo subscritor para realizar algún tipo de procesamiento.

Para visualizar desde la terminal, la medición de un sensor de la plataforma robótica en simulación, se utiliza el siguiente comando. Considerando que la simulación se encuentre corriendo, es decir en “Play”.

```
//rostopic echo “Nombre del topic”
rostopic echo /Touch_Sensor
```

De esta manera, el usuario estará visualizando el valor actual de la medición del sensor táctil simulada. Puede detener su impresión en pantalla con Ctrl+C y luego consultar otro sensor como por ejemplo:

```
rostopic echo /Gyroscope_Sensor
```

Así, el usuario puede consultar manualmente la medición de cualquier sensor en cualquier momento. En la Tabla 2, se muestra la documentación programática necesaria para cada topic tipo Sensor.

Nombre del topic	Funcionalidad	Tipo de mensaje	Unidades físicas
/Color_Sensor/Ambient	Indica la medición del sensor de color en modo ambiente	std_msgs/int16	%
/Encoder_Arm_Sensor	Indica la medición del encoder del motor del brazo actuador	std_msgs/int16	°
/Encoder_Left_Wheel_Sensor	Indica la medición del encoder del motor del motor de la rueda izquierda	std_msgs/int16	°
/Encoder_Right_Wheel_Sensor	Indica la medición del encoder del motor del motor de la rueda derecha	std_msgs/int16	°
/Gyroscope_Sensor	Indica la medición del sensor giroscopio	std_msgs/int16	°
/Touch_Sensor	Indica la medición del sensor táctil	std_msgs/int16	-
/Ultrasonic_Sensor	Indica la medición del sensor ultrasónico	std_msgs/float64	cm

Tabla 2. Especificación programática de los topics para los sensores de la plataforma robótica en simulación. Fuente: Elaboración propia.

Para enviar órdenes desde la terminal a los actuadores de la plataforma robótica simulada, se utiliza el siguiente comando. Se pueden enviar aun cuando la simulación se encuentre en pausa y su efecto se verá cuando se corra la simulación.

```
//rostopic pub “Nombre del topic” *Presionar dos veces TAB*
rostopic pub /Left_Wheel_Motor/vel_cmd std_msgs/Float64 "data: 0.0"
```

Luego el usuario puede manipular el argumento “data: 0.0” estableciendo cualquier número que desee, siempre y cuando cumpla con su debido formato. Al darle enter, la plataforma robótica en la simulación de Gazebo empezará a moverse acorde. Se puede detener el envío de datos con Ctrl+C pero usualmente es necesario especificar un valor igual a cero si desea que el movimiento se detenga. Así, el usuario puede manipular el estado de cada actuador

en cualquier momento. En la Tabla 3, se muestra la documentación programática necesaria para cada topic tipo cmd.

Nombre del topic	Funcionalidad	Tipo de mensaje	Unidades físicas	Máximo valor
/Arm_Motor/vel_cmd	Define una velocidad angular al motor del brazo actuador	std_msgs/Float64	rad/s	27 rad/s
/Left_Wheel_Motor/vel_cmd	Define una velocidad angular al motor de la rueda izquierda	std_msgs/Float64	rad/s	18 rad/s
/Right_Wheel_Motor/vel_cmd	Define una velocidad angular al motor de la rueda derecha	std_msgs/Float64	rad/s	18 rad/s
/cmd_vel	Define una velocidad lineal y angular al movimiento simultáneo de ambas ruedas	geometry_msgs/Twist	m/s para la velocidad lineal y rad/s para la velocidad angular	Velocidad lineal: 0.513 m/s Velocidad angular: 0 rad/s --- Velocidad lineal: 0 m/s Velocidad angular: 7.71 rad/s

Tabla 3. Especificación programática de los topics para los motores de la plataforma robótica en simulación. Fuente: Elaboración propia.

Se podrá notar, que el topic /cmd_vel funciona ligeramente diferente a los demás. Dicho topic es encargado de movilizar la plataforma robótica mediante el movimiento simultáneo de ambas ruedas, semejante a la función “Move Tank”. Para esto, el usuario debe especificar un

mensaje tipo Twist. A modo de ejemplo, se puede ejecutar (emplee la funcionalidad de TAB para llegar a dicho comando):

```
rostopic pub /cmd_vel geometry_msgs/Twist "linear:
  x: -0.1281
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.925"
```

Al ejecutar dicho comando, la plataforma empezará a describir una trayectoria circular – Para detener la plataforma, puede ejecutar el mismo comando pero con todos los valores en 0.0. El usuario puede entonces especificar un valor en la componente x del mensaje y en la componente z del mensaje. La primera corresponde a la velocidad lineal a la que se moverá la plataforma robótica y la segunda corresponde a la velocidad angular en el eje z a la que girara la plataforma. Note que estos dos son los únicos parámetros configurables, pues corresponden a los grados de libertad de la plataforma robótica. También, es conveniente que cuando se utilice este topic, no se utilicen los otros dos topics responsables de mover las ruedas de manera individual. Pero esto no quiere decir que no se pueda.

3.2.2 Control por script

Para controlar la plataforma robótica mediante un script el usuario debe crear un archivo ejecutable de extensión .py. Dicho script es propiamente un código de programación escrito en Python, capaz de utilizar todas las herramientas que dicho lenguaje ofrece. Este script en realidad funciona como un nodo en el entorno ROS, el cual se subscribe a los topics de los sensores y publica en los topics de los actuadores. Lo único relevante para el modelo computacional es las bibliotecas que se deben agregar y las líneas de código encargadas de lectura y envío de datos. Cabe rescatar que por cuestiones de rendimiento y eficiencia es conveniente realizar el diseño programático del comportamiento deseado mediante una clase en Pyhton.

Bibliotecas necesarias:

```
# ROS libraries
import roslib
import rospy

# ROS messages
from std_msgs.msg import Int16
from std_msgs.msg import Float64
from std_msgs.msg import Float32
from geometry_msgs.msg import Twist
```

La forma más común y versátil para la lectura de la medición de un sensor de la plataforma robótica simulada se realiza mediante la siguiente línea de código:

```
//variable_sensor=rospy.wait_for_message('Nombre del topic sensor', Tipo de mensaje)
Color_Sensor=rospy.wait_for_message('Color_Sensor/Ambient', Int16)
```

Esto ocasionara que la variable “Color_Sensor” almacene el siguiente mensaje que se publique en topic “Color_Sensor/Ambient”, el cual corresponde a la medición más reciente del sensor simulado. Cabe rescatar que lo que se almacena en la variable es el mensaje, por lo que si se quiere utilizar propiamente el valor de la medición, ya sea para realizar comparaciones u operaciones, se debe accesar de la siguiente manera:

```
Color_Sensor.data
```

La forma más común y versátil para el envío de órdenes a los actuadores de la plataforma robótica simulada se realiza mediante las siguientes instrucciones.

```
//Publicador= rospy.Publisher('Topic de actuador', ‘Tipo de mensaje’, queue_size=0)
Mover_Brazo= rospy.Publisher('/Arm_Motor/vel_cmd', Float64, queue_size=0)
```

Esta instrucción se encarga de inicializar el topic, por lo que es importante ejecutar dicha línea en las funciones de inicialización del algoritmo. Luego, a lo largo del algoritmo, para

enviar una orden en concreto, se sigue el siguiente comando. Esto ocasionará un movimiento de la articulación correspondiente acorde, siempre y cuando “Data” cumpla con el formato debido.

```
//Publicador.publish(“Data”)
Mover_Brazo.publish(-0.5)
```

Prácticamente, estos métodos son la versión programática del rostopic echo y pub. Es altamente recomendable que el usuario estudie los scripts llamados “Main Controller”, “Target Motor”, “Target Tank” y “Validation” presentes en la carpeta “lmev”. Y que tome estos como fuente de ejemplificación de las herramientas aquí explicadas y como base para sus futuros diseños programáticos.

Para ejecutar un script, se debe abrir una nueva terminal y ejecutar el siguiente código:

```
//rosrun lmev_gazebo “Nombre del script”.py
rosrun lmev_gazebo Validation.py
```

3.3 Otros elementos de interés

El usuario además de solo saber cómo controlar la plataforma robótica, es de utilidad saber manejar y entender algunos otros componentes presentes en el paquete “lmev”. En la dirección “/home/user/lmev/src/lmev_gazebo”, el usuario encontrará 7 carpetas de las cuales interesa el estudio de 4. Para ejemplificar el uso de estas carpetas la mejor forma es que el usuario estudie los pequeños códigos que contienen y entienda los comentarios adjuntos para rápidamente aprender a realizar las modificaciones que necesite.

3.3.1 models

La carpeta llamada models es la carpeta donde se contienen todos los elementos tipo modelos descritos en el formato SDF en Gazebo y todos los archivos adicionales para la descripción de la misma. Por ejemplo, en esta carpeta se encuentra el modelo lmev, que corresponde a la descripción de la plataforma robótica Lego Mindstorm EV3 y todas las piezas tridimensionales construidas (archivos .dae). Note que en esta carpeta se puede

agregar cualquier otro modelo que el usuario desee incluir en la simulación y sus respectivos meshes. Estos modelos pueden ser guardados directamente desde la construcción gráfica de Gazebo sin necesidad de diseñar un código SDF. Es decir, aquí el usuario puede almacenar la descripción de cualquier otro modelo que considere necesario para su simulación.

3.3.2 worlds

La carpeta llamada worlds es la carpeta donde se van a contener todos los elementos tipo mundo descritos en el formato SDF en Gazebo. Por ejemplo, en esta carpeta se encuentra el mundo lmev_world, que corresponde a la descripción del mundo de simulación básico. Note que en esta carpeta se podría agregar cualquier otro mundo que el usuario desee simular. Ahora, si se quiere simular el comportamiento de la plataforma robótica pero con una pista de carreras distinta o en el aula de trabajo, basta solo con cambiar el archivo .world utilizado. De esta manera, se puede realizar simulaciones con distintas condiciones de manera eficiente.

3.3.3 launch

La carpeta llamada launch es la carpeta donde se van a contener instrucciones para la ejecución de distintos nodos de ROS. Por ejemplo en esta carpeta se encuentra el archivo lmev.launch, que se encarga de ejecutar un nodo cuya funcionalidad es inicializar Gazebo y cargar el mundo de simulación especificado, un otro nodo cuya funcionalidad es cargar en el mundo de simulación el modelo lmev en una posición especificada. Note que en esta carpeta se podría agregar cualquier otro archivo launch que ejecute diversos nodos en diverso orden, según como el usuario desee en su simulación.

3.3.4 scripts

La carpeta llamada scripts es la carpeta donde se van a contener los algoritmos programáticos que describen un comportamiento de movimiento y sistema sensorial de la plataforma robótica ensamblada. Por ejemplo en esta carpeta el estudiante puede agregar archivos de extensión .py o .cpp (provenientes de Python o C++, respectivamente) que dentro de sus líneas de código hayan instrucciones de lectura de sensores, una lógica programática y luego instrucciones de control de actuadores de la plataforma robótica en función al algoritmo. Es decir, cada archivo de estos, funciona como un nodo individual en ROS. Prácticamente los archivos de esta carpeta se pueden comparar con los archivos que

se generan al diseñar un comportamiento del Lego Mindstorm EV3 en su software de programación oficial o de RobotC.

4 Ejemplos de uso

Para ejemplificar el uso y funcionalidades del modelo computacional de simulación para el Lego Mindstorms EV3, se pusieron en disposición cuatro comportamientos programáticamente diseñados. El estudiante, al estudiar y probar estos cuatro comportamientos, puede comprender cómo utilizar la simulación de una manera eficiente y así adaptarse a sus herramientas en el menor tiempo posible. La creación de estos ejemplos, se ve también como una oportunidad de desarrollar funcionalidades en la plataforma robótica, pues como se verá a continuación, son comportamientos comunes de la misma. En seguida, una guía de uso para los cuatro ejemplos.

Antes de iniciar, cabe rescatar que si desea terminar la simulación, la mejor forma de hacerlo es primero cerrando la ventana de Gazebo y luego cerrar la terminal en donde se encuentra corriendo roscore.

4.1 Main_Controller

Este algoritmo es el ejemplo más simple, escrito en el lenguaje Python. Ejemplifica la lectura de las mediciones de los sensores simulados y da una primera idea de cómo se diseña un comportamiento básico.

1. Entre al directorio “/home/user/lmev/src/lmev_gazebo/scripts” y abra el archivo Main_Controller.
2. Observe la estructura de la clase y de la función main().
3. Observe en la función Show_Measures() cómo se almacenan las mediciones de cada sensor.
4. Observe el otro método para realizar la lectura de los sensores.
5. Abra una terminal e inicialice la simulación y presione “Play”.

   ```
   cd lmev
   ./run_lmev.sh
   ```

6. En la tercera pestaña de la nueva terminal podrá ver las mediciones de cada sensor cada 2 segundos.

4.2 Target_Motor

Este algoritmo, escrito en el lenguaje Python, ejemplifica una forma alternativa de realizar la lectura de las mediciones de los sensores. Además, se ejemplifica como enviar órdenes al motor del brazo actuador. Note, que eventualmente este algoritmo habilita la funcionalidad de mover el brazo actuador a un ángulo específico. A partir de aquí, además de servir como ejemplo, el usuario dispone de dicho comportamiento como nueva herramienta en la simulación.

1. Entre al directorio "/home/user/lmev/src/lmev_gazebo/scripts" y abra el archivo Target_Motor.
2. Observe el flujo lógico de la programación. Ayúdese con los comentarios.
3. Note la línea 23 y 53. Estas son las encargadas de enviar las ordenes de movimiento al motor.
4. Note la línea 26. Aquí se crea un topic el cual el usuario puede acceder desde la terminal y especificar un ángulo deseado.
5. Abra una terminal, inicialice la simulación y presione "Play".
   ```
   cd lmev
   ./run_lmev.sh
   ```
6. En la misma terminal, ejecute el siguiente comando:
   ```
   rosrun lmev_gazebo Target_Motor.py
   ```
7. Abra una nueva terminal y ejecute el comando:
   ```
   rostopic list
   ```
   El usuario podrá notar la incorporación de un nuevo topic "/Arm_Motor/target_cmd".
8. Ejecute el siguiente comando y note lo que pasa con la plataforma robótica en simulación.
   ```
   rostopic pub /Arm_Motor/target_cmd std_msgs/Int16 "data: -40"
   ```
9. Cuando termine el movimiento, fíjese en la pestaña de mediciones de sensores y note el valor del sensor encoder del brazo.
10. Puede tratar con diferentes valores de ángulo o puede tratar de modificar la velocidad a la que el brazo actuador se mueve modificando la variable en el script.

4.3 Target_Tank

Al igual que el ejemplo anterior, el algoritmo Target_Tank está escrito en el lenguaje Python y no solo encuentra funcionalidad como ejemplo, si no que también permanece como futura herramienta disponible para el usuario. Este comportamiento permite especificar un número de rotaciones por realizar por las ruedas de la plataforma robótica. Se fortalece el control por terminal, la lectura y envío de datos, el uso del movimiento sincronizado de las ruedas y la creación de mensajes ROS.

1. Entre al directorio “/home/user/lmev/src/lmev_gazebo/scripts” y abra el archivo Target_Tank.
2. Observe el flujo lógico de la programación. Ayúdese con los comentarios.
3. En la línea 51 y 63 puede repasar como realizar la lectura de los sensores de manera simple.
4. La línea 24 inicializa el topic del actuador por usar. Y en la línea 60 y 67 se envía la orden de movimiento.
5. Note la línea 34. Aquí se crea de manera programática un mensaje tipo “Twist” y se inicializa su valor.
6. Abra una terminal, inicialice la simulación y presione “Play”.
   ```
   cd lmev
   ./run_lmev.sh
   ```
7. En la misma terminal, ejecute el siguiente comando:
   ```
   rosrun lmev_gazebo Target_Tank.py
   ```
8. Abra una nueva terminal y ejecute el comando:
   ```
   rostopic list
   ```
   El usuario podrá notar la incorporación de un nuevo topic “/Move_Tank_Rev/target_cmd”.
9. Ejecute el siguiente comando y note lo que pasa con la plataforma robótica en simulación.
   ```
   rostopic pub /Move_Tank_Rev/target_cmd std_msgs/Int16 "data: 10"
   ```
10. La plataforma robótica se moverá 10 revoluciones de las ruedas hacia adelante. Cuando termine el movimiento, fíjese en la pestaña de mediciones de sensores y note el valor del sensor encoder de las ruedas.
11. Puede tratar con diferente número de revoluciones, a una diferente velocidad de avance. ¿Cuánto debería la plataforma desplazarse hacia adelante? ¿Cuánto tiempo

debería tomarle completar esas revoluciones a esa velocidad? ¿Qué sucede si se modifica el mensaje Twist para ahora tener también velocidad angular?

4.4 Validation

Este último ejemplo pretende utilizar todas las funcionalidades básicas del modelo computacional desarrollado. Consiste en que la plataforma, al haber un nivel de luz ambiente alto, se mueva diez revoluciones hacia adelante. Luego, tomando en cuenta la medición del sensor giroscopio, girar noventa grados. Una vez ahí, lee la medición del sensor ultrasónico y si existe un objeto cerca de la plataforma entonces baja el brazo actuador. Este ejemplo pretende que el usuario se familiarice con lo que se podría considerar un diseño de algoritmo semejante al que tiene que implementar en las prácticas de clase, en donde utilice gran número de sensores, envíe diferentes órdenes a los motores y en dónde programe un comportamiento lógico deseado. Además, motiva al usuario a utilizar las herramientas de la interfaz gráfica de Gazebo.

1. Entre al directorio “/home/user/lmev/src/lmev_gazebo/scripts” y abra el archivo Target_Tank.
2. Observe el flujo lógico de la programación. Ayúdese con los comentarios.
3. Abra una terminal, inicialice la simulación y presione “Play”.
   ```
   cd lmev
   ./run_lmev.sh
   ```
4. En la misma terminal, ejecute el siguiente comando:
   ```
   rosrun lmev_gazebo Validation.py
   ```
5. Observe con atención que sucede en la plataforma robótica en simulación. Mire la medición del sensor de color en modo ambiente en la tercera pestaña.
6. En la ventana de Gazebo, inserte un cubo al mundo. Eso se logra presionando el símbolo de cubo en el panel superior de la interfaz gráfica y luego haga click en el mundo. Puede cambiar la ubicación del cubo con la herramienta de traslación en el mismo panel. Coloque el cubo aproximadamente a 2m en el eje x y 2m en el eje y. Se puede ayudar observando la “pose” del cubo en el panel izquierdo.
7. Ahora en la ventana de Gazebo, inserte una luz tipo “Point Light”. Esta se encuentra en el panel superior, es un dibujo semejante a un sol. Presiónelo e haga click en algún sitio cercano a la plataforma robótica y frente a ella. Observe con atención. (Este tipo de iluminación es conveniente para un comportamiento de buscador de luz).

8. Cuando el movimiento haya terminado. Puede repetir los pasos del 1 al 7, pero ahora inserte el cubo a una distancia más cercana, por ejemplo 1m en el eje x y 2m en el eje y. ¿Ahora qué sucede cuando la plataforma robótica gira?
9. El usuario puede ahora experimentar a gusto, modificando las variables en el algoritmo y modificando dinámicamente los modelos en la interfaz gráfica de Gazebo.