



3.3 Simulado Robot con Gazebo

César Omar Alvarado Contreras

Jonathan Fonseca Camarena

Marcos Manzo Torres

Eduardo Robles Vázquez

Víctor Gabriel Tapia Casillas

Universidad Politécnica de la Zona Metropolitana de Guadalajara

Profesor: Carlos Enrique Morán Garabito

8 de noviembre del 2019

Índice general

1	Gazebo	3
2	Desarrollo	4
3	Simulado	8
4	Control de la cámara	9
5	Mover el Robot por el mundo virtual usando comandos de velocidad	10
6	conclusión	13

Capítulo 1

Gazebo

Gazebo es un simulador de robótica en 3D de código abierto. Gazebo integró el motor físico ODE de renderizado de OpenGL y código de soporte para la simulación y control de actuadores. En 2011, Gazebo se convirtió en un proyecto independiente apoyado por Willow Garage.

En 2012, la Fundación de robótica de código abierto (OSRF por sus siglas en inglés) se convirtió en el administrador del proyecto Gazebo. Posteriormente, la OSRF cambió su nombre a Open Robotics en 2018.

Gazebo puede utilizar múltiples motores físicos de alto desempeño, como ODE, Bullet, etc (Por defecto viene siendo ODE). Este provee de renderizados realistas de ambientes, incluyendo iluminaciones, sombras y texturas, todas estas de gran calidad. Puede modelar sensores que pueden "ver" el ambiente simulado, como buscadores láser de rango, cámaras y sensores del estilo Kinect entre otros.

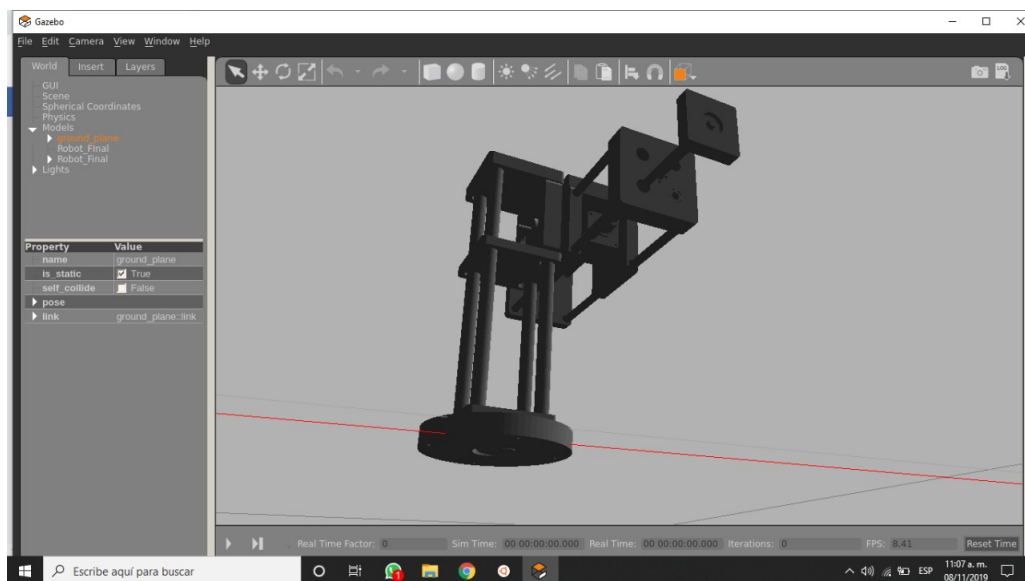


Figura 1.1: Prototipo del Robot.

:

Capítulo 2

Desarrollo

Primeramente y partiendo del ensamble realizado en SolidWorks, exportamos dicho ensamble como .STL con lo cual el archivo podrá importarse en Blender de manera sencilla por medio de mallas.

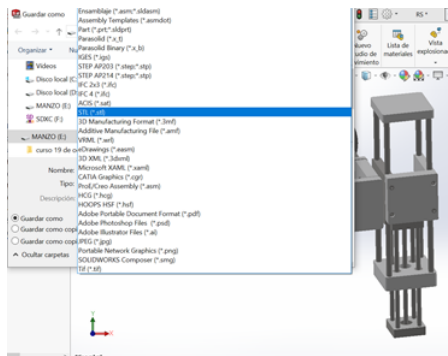


Figura 2.1: Exportación en SolidWorks

Iniciamos nuestra máquina virtual, donde trabajaremos con el archivo .STL










Archivo	Máquina	Ayuda
<div><div>Tools</div><div></div><div><div>Ubuntu Apagada</div><div>MARCOD MANZO TORRES Corriendo</div><div>ros1 Apagada</div><div>robot Apagada</div></div><div><div>General Nombre: MARCOD MANZO TORRES Sistema operativo: Ubuntu (64-bit) Settings File Location: C:\Users\Marcos\VirtualBox VMs\MARCOD MANZO TORRES</div><div>Sistema Memoria base: 4096 MB Orden de arranque: Disquete, Óptica, Disco duro Aceleración: VT-x/AMD-V, Paginación anidada, Paravirtualización KVM</div><div>Pantalla Memoria de vídeo: 16 MB Graphics Controller: VMXVGA Servidor de escritorio remoto: Inhabilitado Recording: Inhabilitado</div><div>Almacenamiento Controlador: IDE IDE primario maestro: [Unidad óptica] Vacío IDE secundario maestro: [Unidad óptica] Vacío Controlador: SATA Puerto SATA 0: MARCOD MANZO TORRES.vdi (Normal, 102.18 GB)</div></div></div>		

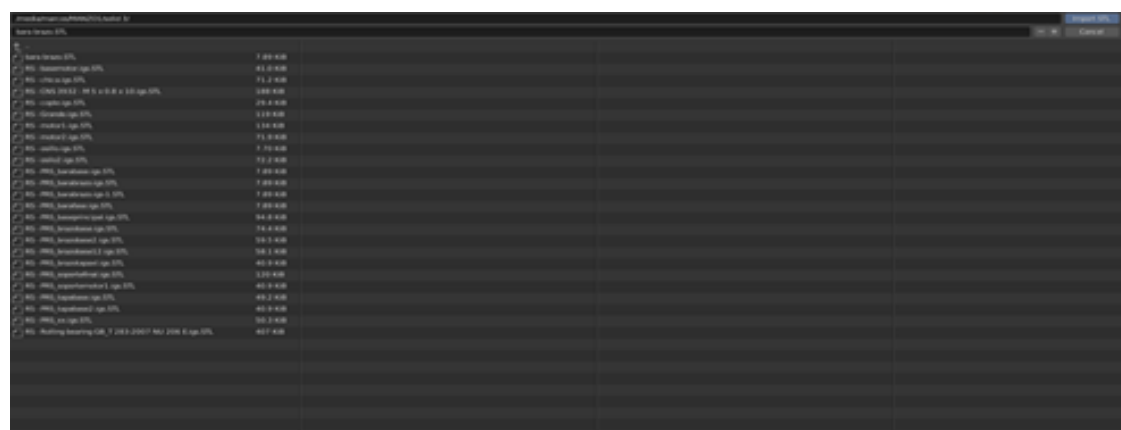
Figura 2.2: Inicio de máquina virtual

A screenshot of a terminal window titled "marcos@marcos-VirtualBox: ~". The terminal shows the execution of several commands and their outputs:

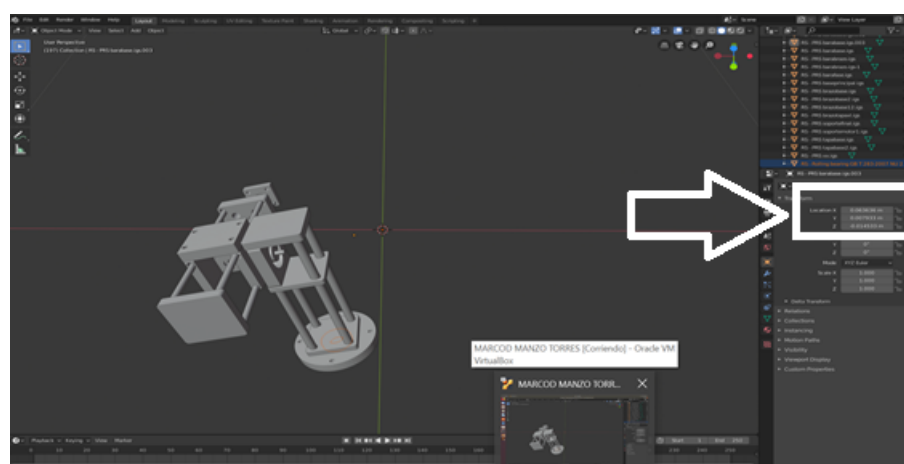
```
marcos@marcos-VirtualBox: ~  
$ Archive Editor via Secure Terminal - Apache  
$ sudo systemctl restart blender.service  
Saved session recovery to '/tmp/quilt.blend'  
$ blender quit  
marcos@marcos-VirtualBox:~$ blender  
Need more space? marcos/.config/blender/2.81/config/userpref.blend  
[Received X11 Error:  
    error code: 179  
    request code: 155  
    minor code: 34  
    error text: GLXBadPBufferConfig  
] [Received X11 Error:  
    error code: 179  
    request code: 155  
    minor code: 34  
    error text: GLXBadPBufferConfig  
] [Received X11 Error:  
    error code: 179  
    request code: 155  
    minor code: 34  
    error text: GLXBadPBufferConfig  
] [Received X11 Error:  
    error code: 179  
    request code: 155  
    minor code: 34  
    error text: GLXBadPBufferConfig  
]
```


At the bottom of the terminal, there are status indicators: a green bar on the left, and system information on the right: "CPU: 0% / Mem: 60% / GPU: 0%".

Tras haber exportado el archivo .STL de SolidWorks, lo importamos en Blender y observamos que el ensamble está subdividido por cada unas de las piezas que lo conforman



Una vez abrimos el ensamble, observamos que nos faltan algunas piezas las cuales se colocan de manera manual por medio de la opción transformar y por medio de coordenadas, se colocan en el lugar deseado.



5

Ahora que finalizamos el ensamble importado en Blender, guardamos el mismo como archivo .DAE y lo colocamos en nuestra carpeta launcher de Gazebo

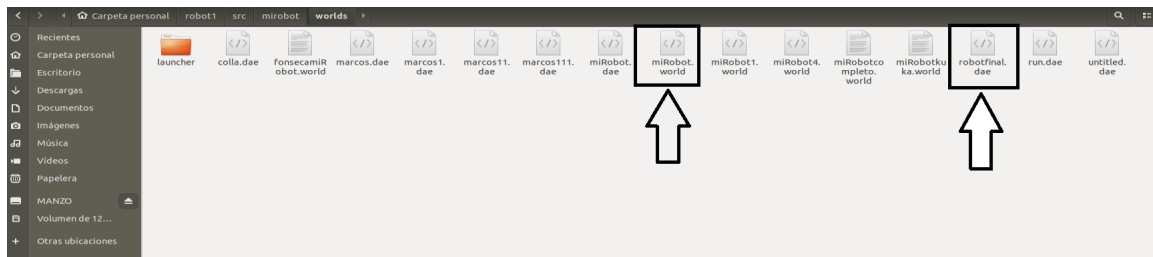


Figura 2.6: Guardar ensamble

Observamos la información almacenada en nuestro archivo .DAE, donde nuestro ensamble fue guardado por medio de mallas similar a un archivo .USDF que es el archivo base de Gazebo. Encontramos cómo se declara el entorno y las uniones para que estas aparezcan a la hora de correr el mismo.

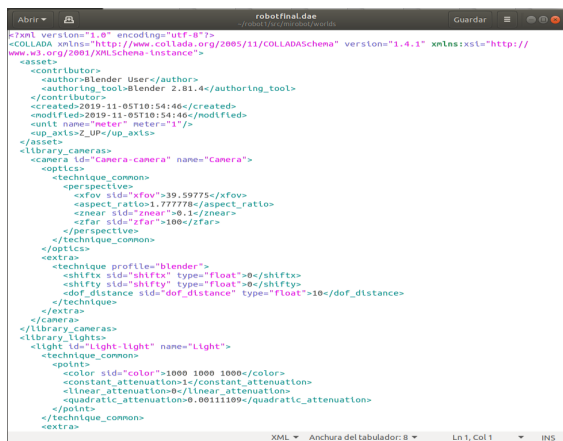


Figura 2.7: Archivo .DAE

Para iniciar el contenido en Gazebo, necesitamos declarar un archivo .World el cual funciona como launcher para iniciar el contenido o arrancar el contenido especificado. solo basta con colocar el nombre del archivo .DAE que deseamos iniciar. Ahora solo basta abrir un terminal, donde iniciaremos gazebo y

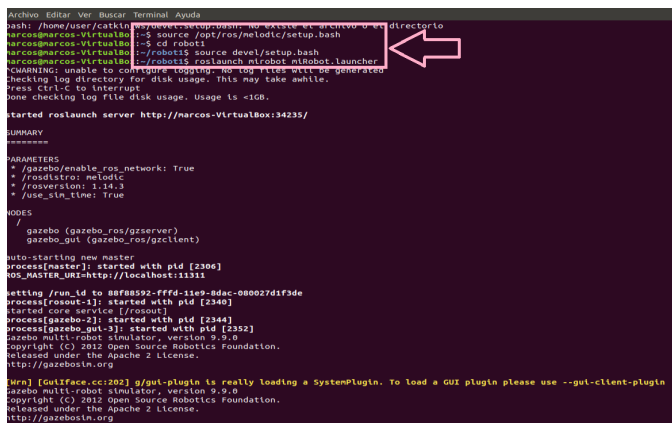
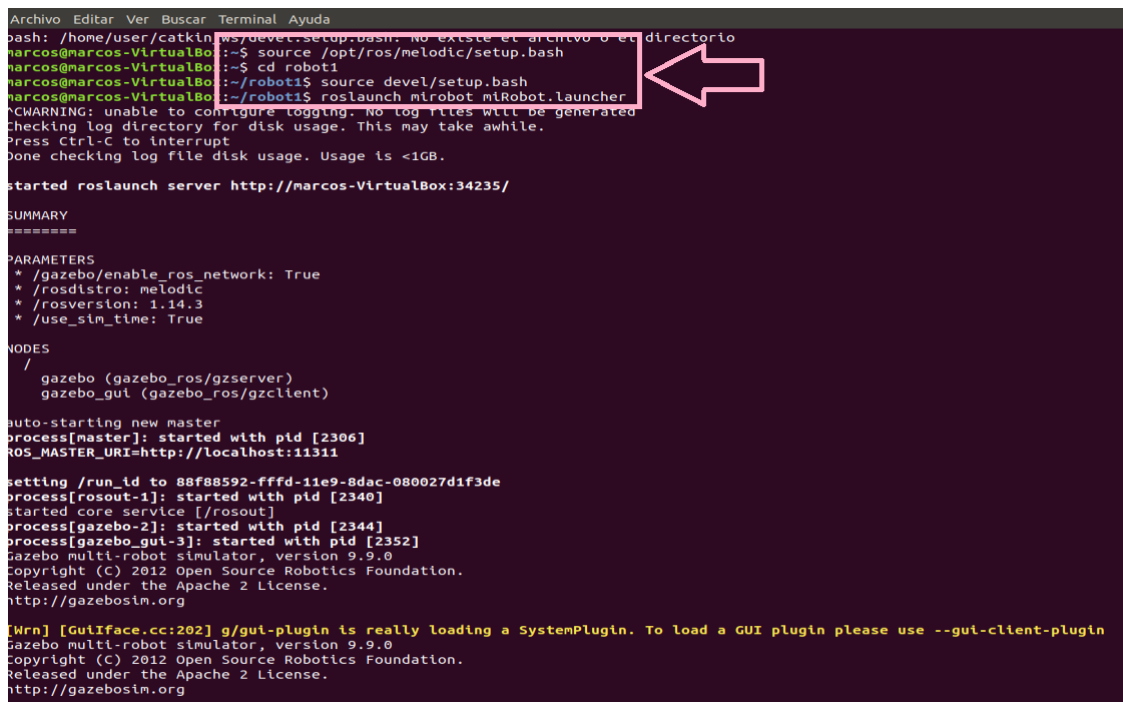


Figura 2.8: Arrancador

nos abrirá nuestro archivo colocado en el arrancador y en nuestra carpeta de trabajo.



```
Archivo Editar Ver Buscar Terminal Ayuda
bash: /home/user/catkin_ws/devel/setup.bash: No existe el archivo o el directorio
marcos@marcos-VirtualBox: ~$ source /opt/ros/melodic/setup.bash
marcos@marcos-VirtualBox: ~$ cd robot1
marcos@marcos-VirtualBox: ~/robot1$ source devel/setup.bash
marcos@marcos-VirtualBox: ~/robot1$ roslaunch mirobot miRobot.launcher
RCWARNING: unable to configure logging. No log files will be generated
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://marcos-VirtualBox:34235/

SUMMARY
=====
PARAMETERS
* /gazebo/enabe_ros_network: True
* /roscdistro: melodic
* /rosversion: 1.14.3
* /use_sim_time: True

NODES
/
  gazebo (gazebo_ros/gzserver)
  gazebo_gui (gazebo_ros/gzclient)

auto-starting new master
process[master]: started with pid [2306]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 88f88592-fffd-11e9-8dac-080027d1f3de
process[rosout-1]: started with pid [2340]
started core service [/rosout]
process[gazebo-2]: started with pid [2344]
process[gazebo_gui-3]: started with pid [2352]
Gazebo multi-robot simulator, version 9.9.0
Copyright (C) 2012 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazeboosim.org

[Warn] [Guiiface.cc:202] g/gui-plugin is really loading a SystemPlugin. To load a GUI plugin please use --gui-client-plugin
Gazebo multi-robot simulator, version 9.9.0
Copyright (C) 2012 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazeboosim.org
```

Figura 2.9: Iniciar Gazebo

Tras haber iniciado Gazebo en nuestra terminal, podemos observar una ventana nueva donde nos aparece el contenido y su entorno, además de poder observar a detalle la simulación de nuestro robot.

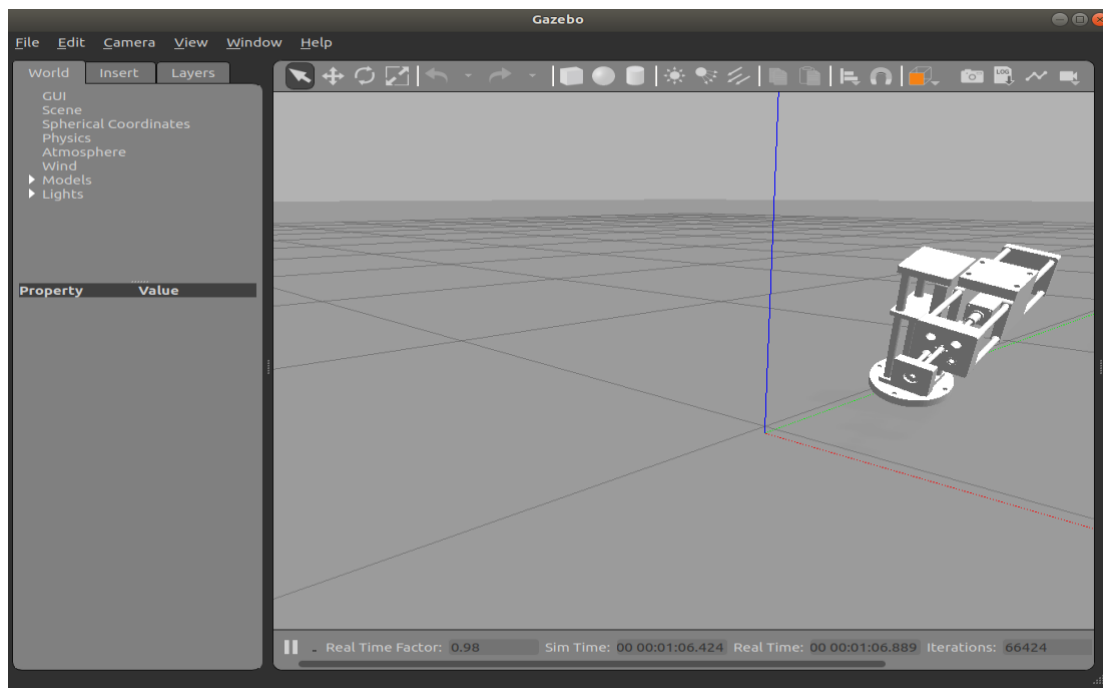


Figura 2.10: Gazebo

Capítulo 3

Simulado

Simulador 3D Gazebo Gazebo es un simulador de entornos 3D que posibilita evaluar el comportamiento de un robot en un mundo virtual. Permite, entre muchas otras opciones, diseñar robots de forma personalizada, crear mundos virtuales usando sencillas herramientas CAD e importar modelos ya creados.

Además, es posible sincronizarlo con ROS de forma que los robots emulados publiquen la información de sus sensores en nodos, así como implementar una lógica y un control que dé ordenes al robot.

Gazebo forma parte del bundle de ros ros-kinetic-desktop-full”, no obstante el robot que se usará como ejemplo no está integrado. El robot al que nos referimos es Turtlebot, un pequeño robot con una estructura montada sobre una base de Roomba y que integra sensores de odometría y una cámara RGB-D, entre otros. El proceso de instalación se detalla en la Wiki de ROS, donde además se encuentra toda la documentación y otros tutoriales de interés.

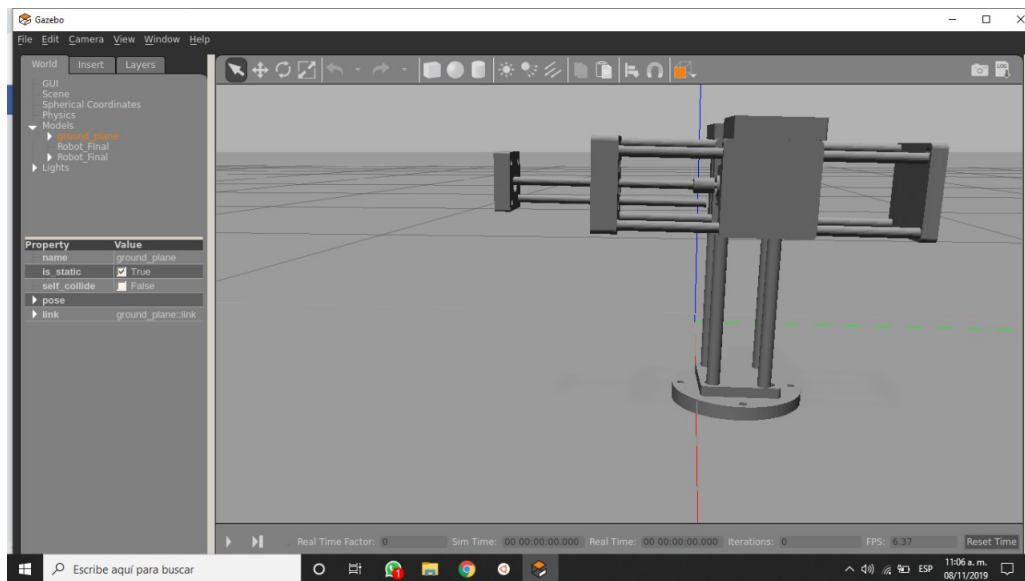


Figura 3.1: Orientación.

:

Capítulo 4

Control de la cámara

El control del punto de vista de diseño es parecido al que muestran otros programas de diseño o simulación 3D:

El botón izquierdo del ratón sirve para trasladar la cámara por el mundo.

El botón derecho o la rueda del ratón sirve para hacer zoom en la escena

El botón central del ratón sirve para rotar la cámara

Diseño de mundos

Gazebo permite diseñar elementos con herramientas CAD, dichas herramientas son muy limitadas, pero suficientes para crear mundos sencillos. Estas herramientas están situadas en la zona superior del programa y nos permite insertar cubos, esferas y cilindros, así como distintos puntos de luz.

Los elementos que contiene el mundo se van listando en la parte izquierda del programa, bajo la pestaña "Worldz" si son seleccionados permiten la edición de algunas propiedades tales como la pose o el nombre (esta característica no está implementada en la versión de que disponemos en el laboratorio).

Mover el Robot por el mundo virtual usando comandos de velocidad

La forma de mover los robots, tanto en el mundo real como en el simulador, es enviándoles comandos de velocidad. Los robots tienen un nodo que consumen mensajes de tipo geometry msgs Twist que son el tipo de mensajes para indicar precisamente eso, valores para velocidades lineales y angulares.

En este bloque vamos a aprender cómo mover el Robot mandándole comandos de velocidad tal y como hacíamos en la sección Mover a turtlebot por el mundo virtual usando comandos de velocidad pero en este caso en vez de mandarle el mensaje directamente desde la terminal lo haremos desde un nodo ROS. Las ventajas de este sistema son evidentes: podremos reunir la información de los sensores para averiguar la situación del robot y el estado del entorno, procesar estos datos y generar los comandos de movimiento necesarios para guiar el robot en función de la sensorización.

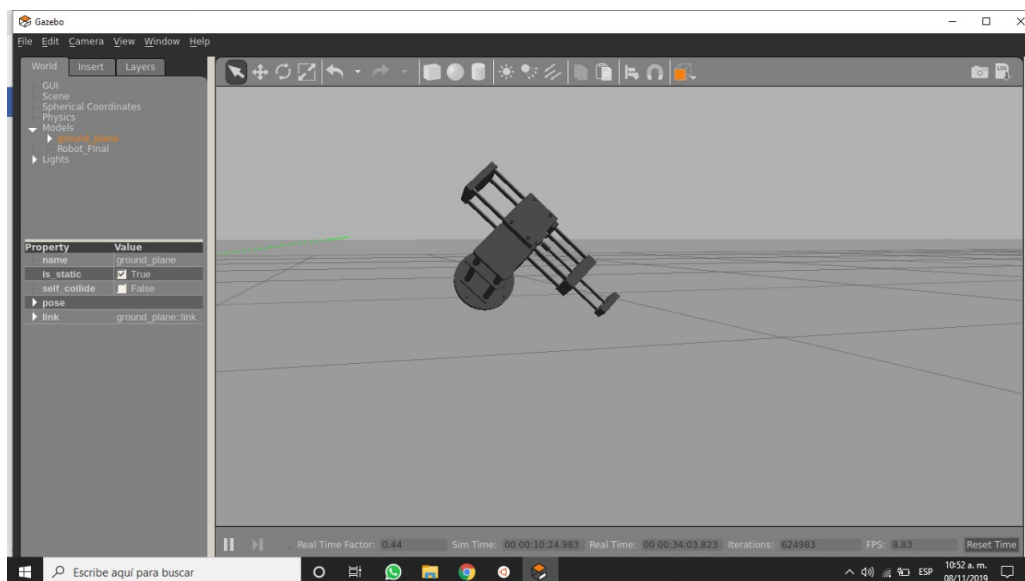


Figura 5.1: Primeros Movimientos.

:

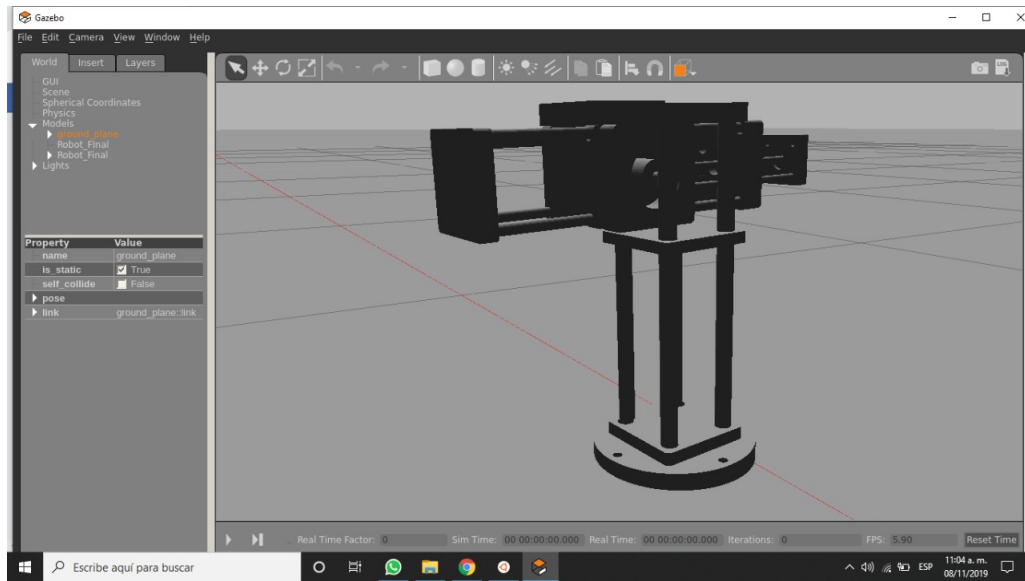


Figura 5.2: Direccionando Movimientos.

:

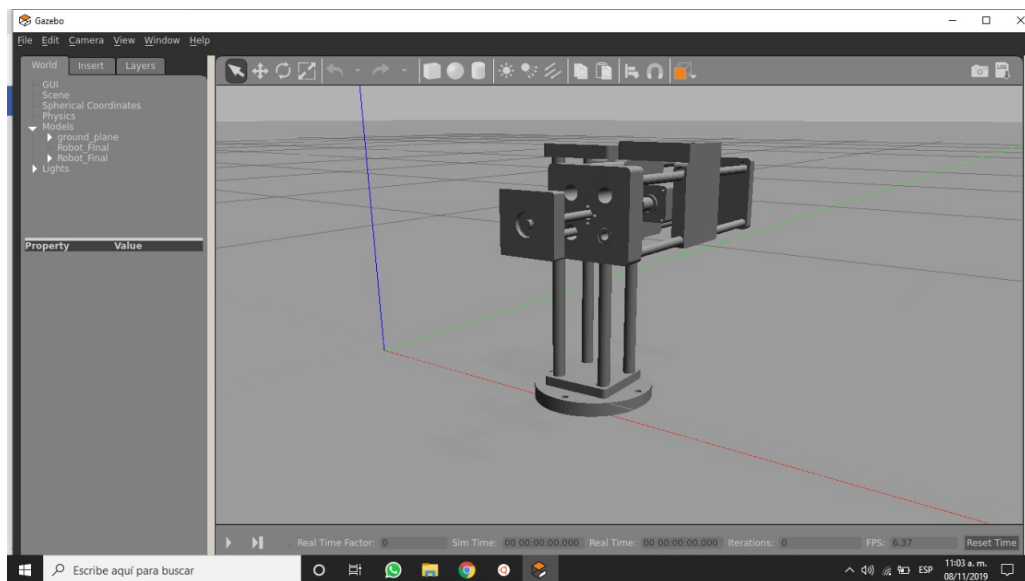


Figura 5.3: Acomodando Dirección.

:

Un mensaje de este tipo acepta como máximo 6 valores: las velocidades lineales en x, y, z y las velocidades angulares de roll, pitch y yaw. Estos valores permiten dar movimiento a robots con hasta 6 grados de libertad, pero turtlebot sólo puede moverse por el plano XY y rotar en el eje Z (yaw). Pero antes de enviar nada es necesario averiguar qué tópicos es que acepta este tipo de mensajes por lo que usamos el siguiente comando para ver qué tópicos tenemos en el sistema.

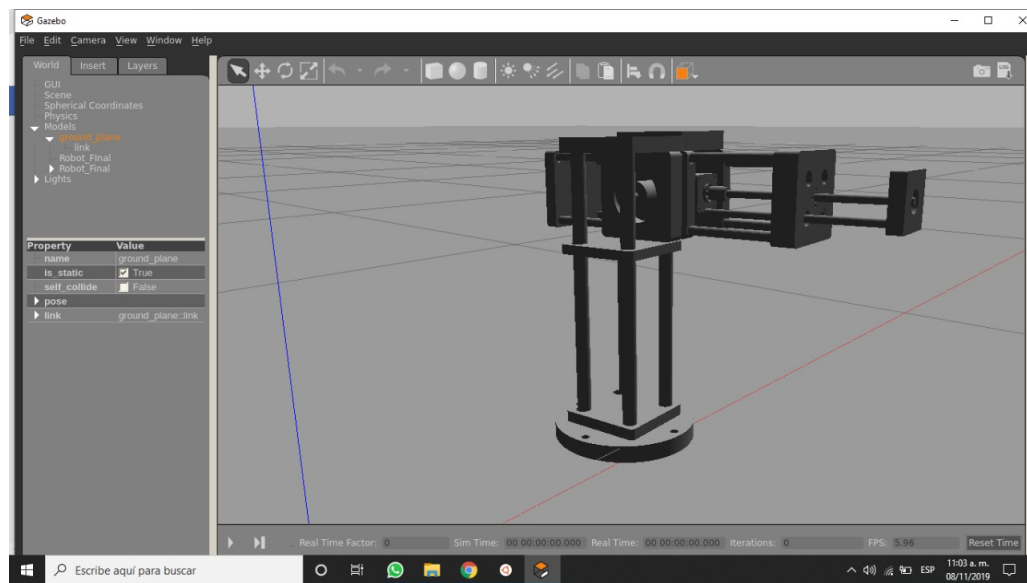


Figura 5.4: Prototipo Movindose.

:

conclusión

En la practica, consistía en familiarizarse con el simulador Gazebo, pues tampoco se conocía su funcionamiento completamente. De igual manera que con ROS, tras la labor de investigación llevada a cabo y el trabajo desarrollado, se considera superado este objetivo. Siendo capaces de simular un entorno y un robot en él correctamente, y comprendiendo el funcionamiento del programa. Por último, logramos ser capaces de integrar todas estas herramientas para que interactuasen entre ellas. Como ha quedado demostrado, este objetivo también se ha logrado, siendo capaces de mover el robot en la simulación pasando mensajes desde nodos de ROS y, de igual manera, siendo capaces de transmitir la información de la simulación a través de los codigos.

[?] [?]