

**INTEGRANTES DEL EQUIPO:**

ARIAS RAMOS JOSÉ ANTONIO REY.

**MARCO TEORICO**

Un robot es una máquina controlada por ordenador y programada para moverse,

manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Su objetivo principal es el de sustituir al ser humano en tareas repetitivas, difíciles, desagradables e incluso peligrosas de una forma más segura, rápida y precisa.

Los robots exhiben tres elementos claves según la dentición adoptada:

- Programable, lo que sígnica disponer de capacidades computacionales y de manipulación de símbolos (el robot es un computador).

- Capacidad mecánica, que lo capacita para realizar acciones en su entorno y no ser un mero procesador de datos (el robot es una máquina).

- Flexibilidad, puesto que el robot puede operar según un amplio rango de programas y manipular material de formas distintas.

**ROS**

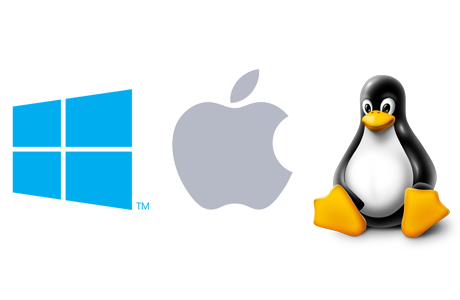
ROS (Robot Operating System) es una plataforma de desarrollo open source para

sistemas robóticos. Proporciona toda una serie de servicios y librerías que simpliﬁcan Considerablemente la creación de aplicaciones complejas para robots.

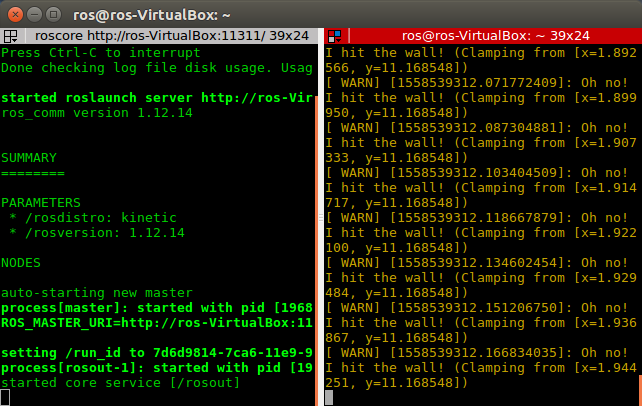
Es similar a otras plataformas de desarrollo para robots existentes en la actualidad, y su mayor virtud, es la de haber sabido aunar lo mejor de cada uno de estos sistemas, juntando todo en un solo sistema capaz de comunicarse tanto con los robots más modernos, como con los ya existentes en el mercado.



Desde su creación, ROS se ha diseñado para facilitar el intercambio de software entre los aﬁcionados y los profesionales de la robótica en todo el mundo debido a su enfoque didáctico y abierto, lo que ha permitido la construcción de una gran comunidad de colaboradores a lo largo de todo el mundo.

A la hora de desarrollar, ROS permite el uso de distintos lenguajes de programación. De forma oﬁcial soportan Python, C++ y Lisp además de muchas otras como   java (todavía en fase experimental pero apoyada por Google), Lua, etc.

ROS puede ser ejecutado sobre maquinas tipo Unix, principalmente Ubuntu y Mac OS X, aunque por parte de la comunidad puede encontrarse soporte para otras plataformas como Fedora, Gentoo, etc.

Para el desarrollo de la practica con la TurtleSim se seguirán los códigos que se muestran a continuación:

Para los paquetes:

*$ rospack find turtlesim*

Ejecutar los paquetes para obtener los archivos:

*$ rosls turtlesim*

Carpeta y commando de roscd:

*$ roscd turtlesim*

Creación del espacio de trabajo:

*$ echo $ROS\_PACKAGE\_PATH*

Creación de la carpeta:

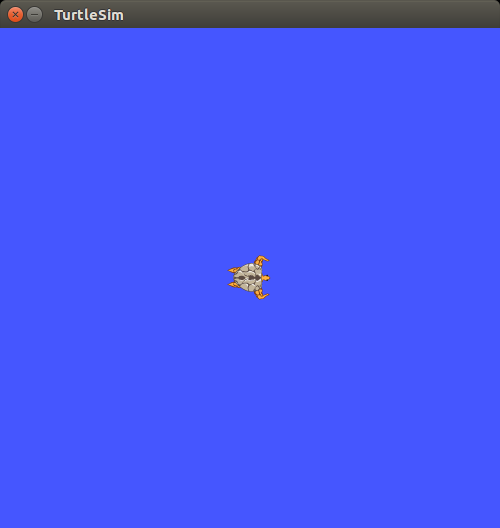
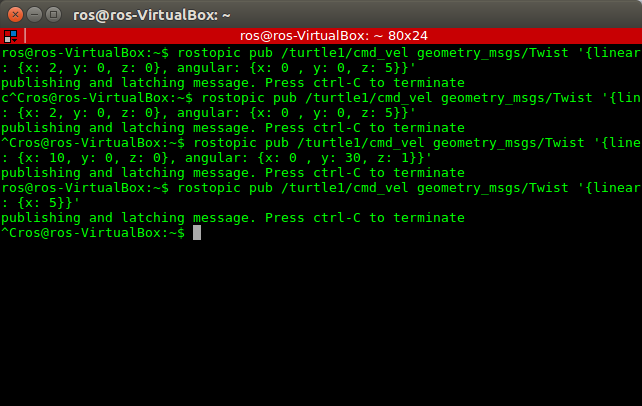
*$ cd ~*

*$ mkdir -p dev/rosbook*

*$ echo "export ROS\_PACKAGE\_PATH=~/dev/rosbook:${ROS\_PACKAGE\_PATH}" >> ~/.bashrc*

*$ . ~/.bashrc*

*Creación de paquetes de trabajo:*

*$ roscreate-pkg davebobo\_tutorials std\_msgs rospy roscpp*

Las dependencies:

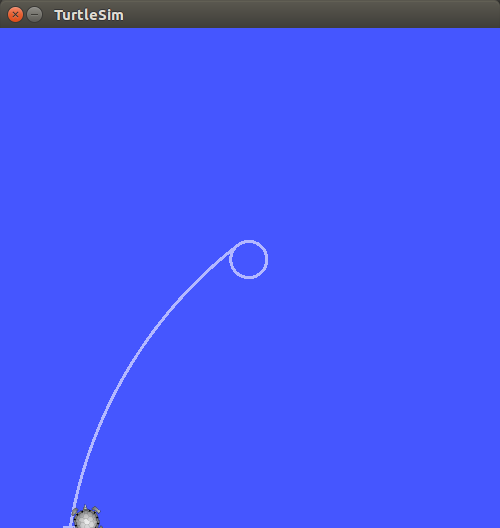
*$ roscreate-pkg [package\_name] [depend1] [depend2] [depend3]*

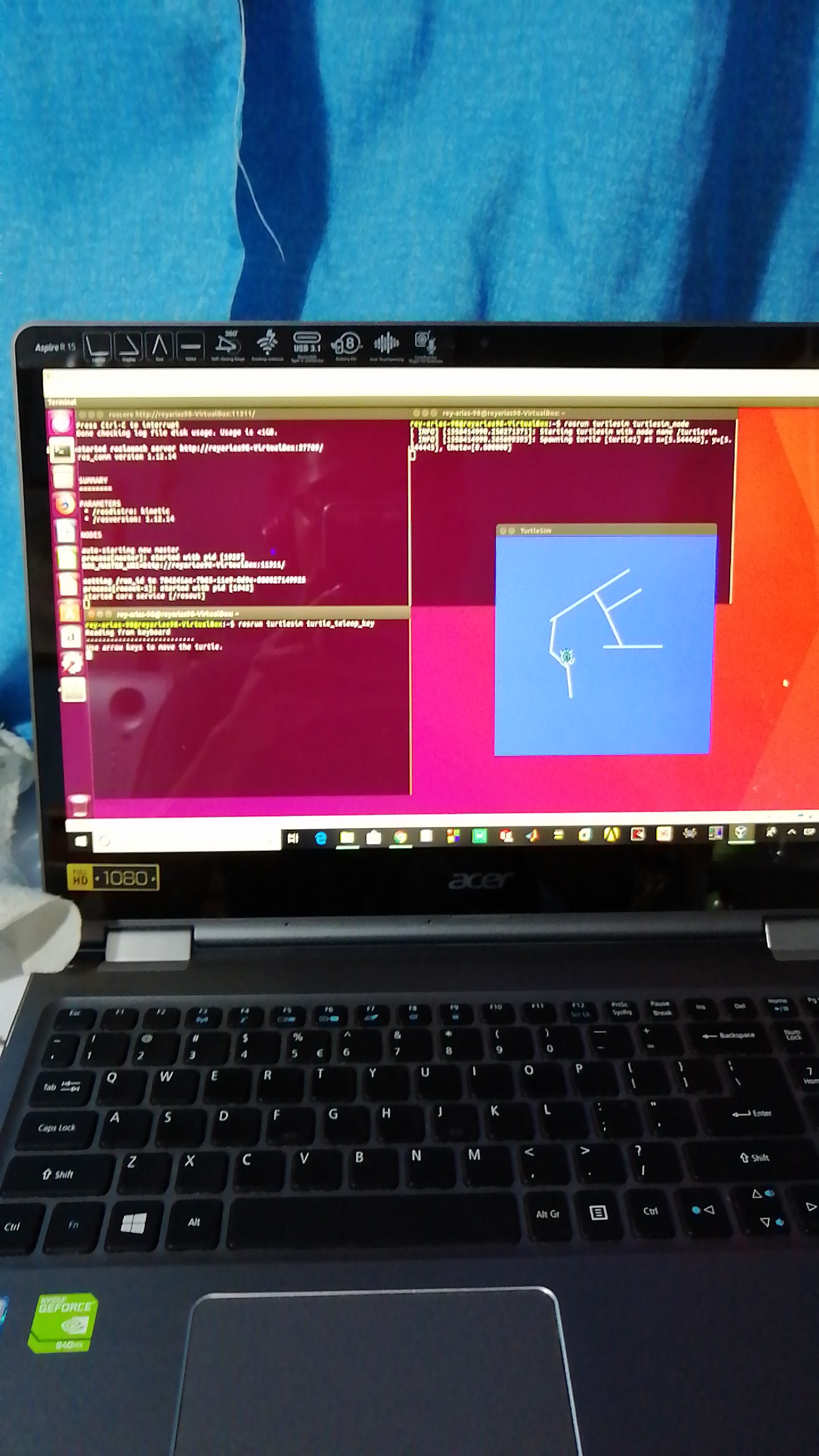
Usando ROS nodos

*$ roscore*

*$ rosrun turtlesim turtlesim\_node*

*$ rosnode info /turtlesim*





Conclusión:

REY ARIAS:

En esta práctica se implementó la simulación de los desplazamientos que tiene el sistema ros, con comandos que realizaban el desplazamiento lineal o realizaba las rotaciones que se le programaban con ángulos, la verdad fue una práctica un poco complicada por algunas líneas de comandos que estaban mal y a la vez bastante útil, ya que en futuras referencias se le aplicaran la programación a nuestro proyecto de materia, con ello podemos determinar cómo es la programación implementada en ros.