

# Preparación entorno 'Autonomous Stretcher Project'

El primer paso es instalar el software que utilizaremos para emular nuestro robot, ROS Kinetic. Instalaremos la versión Full (Desktop-Full Install), que incluye paquetes que utilizaremos como rviz (para gestionar eventos y sensores): <http://wiki.ros.org/kinetic/Installation/Ubuntu>

\*\*\* Seguimos el tutorial hasta el punto 1.5, donde podemos obviar la comanda 'source ~/.zshrc'.

El segundo paso es instalar gazebo (para crear el entorno de emulación y los modelos del robot) paso a paso a partir de la guía de la página oficial: [http://gazebo.org/tutorials?tut=install\\_ubuntu](http://gazebo.org/tutorials?tut=install_ubuntu)

El tercer paso instalar turtlebot (obtenido del manual oficial → <https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/#pc-setup>) :


1. Ejecutamos la comanda 'sudo apt-get update'.
2. Ejecutamos la comanda 'sudo apt-get upgrade'.
3. Ejecutamos la siguiente comanda para instalar dependencias de ROS:  

```
sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy \
ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc \
ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan \
ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python \
ros-kinetic-rosserial-server ros-kinetic-rosserial-client \
ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server \
ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro \
ros-kinetic-compressed-image-transport ros-kinetic-rqt* \
ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers
```
4. Instalamos turtlebot3 con las siguientes comandas:
  1. sudo apt-get install ros-kinetic-dynamixel-sdk
  2. sudo apt-get install ros-kinetic-turtlebot3-msgs
  3. sudo apt-get install ros-kinetic-turtlebot3
5. Por último indicamos el modelo de turtlebot3 que queremos utilizar:  

```
echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
```

Una vez instalado ROS Kinetic, procedemos a generar la base de nuestro proyecto:

1. Ejecutamos la consola en modo administrador:



```
root@sergio-desktop: /home/sergio
sergio@sergio-desktop:~$ sudo su
[sudo] password for sergio:
root@sergio-desktop: /home/sergio#
```

2. Posicionar en la raíz:

```
root@sergio-desktop: ~  
root@sergio-desktop:/home/sergio# cd ~/  
root@sergio-desktop:~#
```

3. Crear carpeta catkin, entrar en as y crear src:

```
root@sergio-desktop: ~/catkin_ws  
root@sergio-desktop:~# mkdir catkin_ws  
root@sergio-desktop:~# cd catkin_ws/  
root@sergio-desktop:~/catkin_ws# mkdir src  
root@sergio-desktop:~/catkin_ws#
```

4. Nos posicionamos en as y ejecutamos la comanda catkin\_make

```
root@sergio-desktop: ~/catkin_ws  
  
root@sergio-desktop:~/catkin_ws# catkin_make  
Base path: /root/catkin_ws  
Source space: /root/catkin_ws/src  
Build space: /root/catkin_ws/build  
Devel space: /root/catkin_ws/devel  
Install space: /root/catkin_ws/install  
Creating symlink "/root/catkin_ws/src/CMakeLists.txt" pointing to "/opt/ros/kinetic/share/catkin/cmake/toplevel.cmake"  
####  
#### Running command: "cmake /root/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/root/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/root/catkin_ws/install -G Unix Makefiles" in "/root/catkin_ws/build"  
####  
-- The C compiler identification is GNU 5.4.0  
-- The CXX compiler identification is GNU 5.4.0  
-- Check for working C compiler: /usr/bin/cc  
-- Check for working C compiler: /usr/bin/cc -- works  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- Detecting C compile features  
-- Detecting C compile features - done  
-- Check for working CXX compiler: /usr/bin/c++
```

5. Nos posicionamos en la carpeta src y ejecutamos las siguientes comandas:

1. catkin\_create\_pkg as\_bringup

```
root@sergio-desktop: ~/catkin_ws/src  
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_bringup  
Created file as_bringup/CMakeLists.txt  
Created file as_bringup/package.xml  
Successfully created files in /root/catkin_ws/src/as_bringup. Please adjust the values in package.xml.  
root@sergio-desktop:~/catkin_ws/src#
```

## 2. catkin\_create\_pkg as\_description

```
root@sergio-desktop: ~/catkin_ws/src
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_description
Created file as_description/package.xml
Created file as_description/CMakeLists.txt
Successfully created files in /root/catkin_ws/src/as_description. Please adjust the values in package.xml.
root@sergio-desktop:~/catkin_ws/src#
```

## 3. catkin\_create\_pkg as\_gazebo

```
root@sergio-desktop: ~/catkin_ws/src
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_gazebo
Created file as_gazebo/package.xml
Created file as_gazebo/CMakeLists.txt
Successfully created files in /root/catkin_ws/src/as_gazebo. Please adjust the values in package.xml.
root@sergio-desktop:~/catkin_ws/src#
```

## 4. catkin\_create\_pkg as\_navigation

```
root@sergio-desktop: ~/catkin_ws/src
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_navigation
Created file as_navigation/package.xml
Created file as_navigation/CMakeLists.txt
Successfully created files in /root/catkin_ws/src/as_navigation. Please adjust the values in package.xml.
root@sergio-desktop:~/catkin_ws/src#
```

## 5. catkin\_create\_pkg as\_slam

```
root@sergio-desktop: ~/catkin_ws/src
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_slam
Created file as_slam/CMakeLists.txt
Created file as_slam/package.xml
Successfully created files in /root/catkin_ws/src/as_slam. Please adjust the values in package.xml.
root@sergio-desktop:~/catkin_ws/src#
```

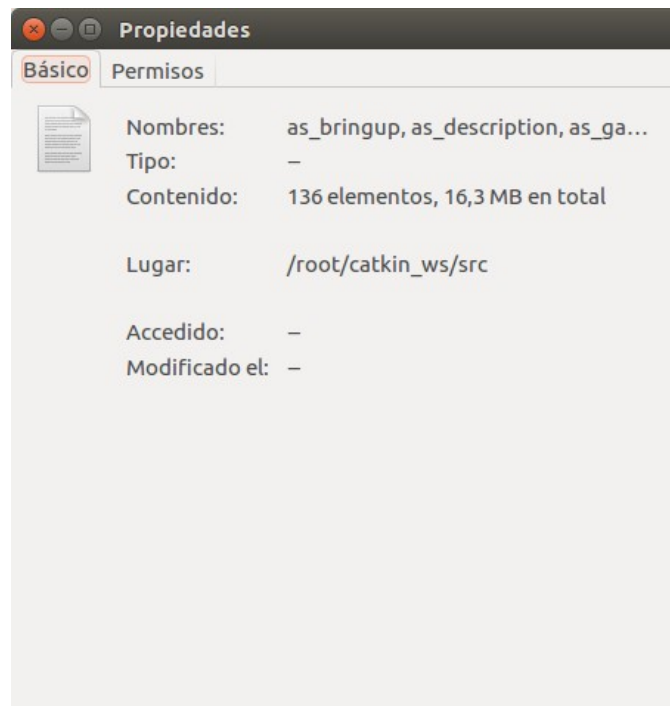
## 6. catkin\_create\_pkg as\_teleop

```
root@sergio-desktop: ~/catkin_ws/src
root@sergio-desktop:~/catkin_ws/src# catkin_create_pkg as_teleop
Created file as_teleop/package.xml
Created file as_teleop/CMakeLists.txt
Successfully created files in /root/catkin_ws/src/as_teleop. Please adjust the values in package.xml.
root@sergio-desktop:~/catkin_ws/src#
```

## 6. Ahora tenemos 2 opciones para copiar los ficheros de la carpeta catkin\_ws:

1. Copiar todos los archivos de cada carpeta correspondiente, y editar los archivos 'CmakeLists.txt' y 'package.xml'
2. Copiar todos los archivos excepto 'CmakeLists.txt' y 'package.xml' (esto es debido a que ya los ha generado los comandos del paso anterior).

Recomendamos utilizar la 2ª opción por asegurar que estos archivos son coherentes.



7. Editamos archivo ~/.bashrc, dónde añadiremos:

- 'source ~/catkin\_ws/devel/setup.bash'
- export TURTLEBOT3\_MODEL=burger

```
root@sergio-desktop: ~/as
root@sergio-desktop:~/as# gedit ~/.bashrc
```

```
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=burger
```

8. Hacer `catkin_make` otra vez para recompilar todo.

```
root@sergio-desktop: ~/catkin_ws
root@sergio-desktop:~/catkin_ws# catkin_make
Base path: /root/catkin_ws
Source space: /root/catkin_ws/src
Build space: /root/catkin_ws/build
Devel space: /root/catkin_ws/devel
Install space: /root/catkin_ws/install
####
#### Running command: "cmake /root/catkin_ws/src -DCATKIN_DEVEL_PREFIX=/root/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/root/catkin_ws/install -G Unix Makefiles" in "/root/catkin_ws/build"
####
-- Using CATKIN_DEVEL_PREFIX: /root/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /root/as/devel;/opt/ros/kinetic
-- This workspace overlays: /opt/ros/kinetic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.12", minimum required is "2")
-- Using PYTHON_EXECUTABLE: /usr/bin/python2
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
```

9. Copiamos las siguientes carpetas en la ruta dónde se guardan los modelos de gazebo, en nuestro caso `/root/.gazebo/models`:

1. AS\_robot

```
root@sergio-desktop: /home/sergio/Escritorio/Autonomous_Stretcher
root@sergio-desktop:/home/sergio/Escritorio/Autonomous_Stretcher# cp -r ./AS_robot /root/.gazebo/models/
root@sergio-desktop:/home/sergio/Escritorio/Autonomous_Stretcher#
```

2. Todas las subcarpetas de models

```
root@sergio-desktop: /home/sergio/Escritorio/Autonomous_Stretcher
root@sergio-desktop:/home/sergio/Escritorio/Autonomous_Stretcher# cp -r ./models/* /root/.gazebo/models/
root@sergio-desktop:/home/sergio/Escritorio/Autonomous_Stretcher#
```

10. Una vez hecho esto, ya podemos ejecutar el proyecto. Para ejecutarlo usaremos el siguiente comando: `roslaunch as_gazebo as_world.launch`

```
/root/catkin_ws/src/as_gazebo/launch/as_world.launch http://localhost:11311
root@sergio-desktop:~/catkin_ws# roslaunch as_gazebo as_world.launch
... logging to /root/.ros/log/6f1494ee-bd85-11eb-a009-e03f496dafdf/roslaunch-ser
gio-desktop-7422.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

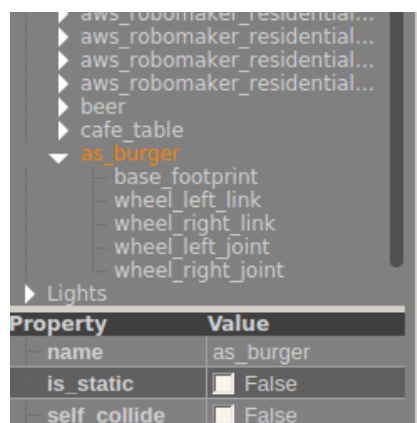
started roslaunch server http://sergio-desktop:38759/

SUMMARY
=====

PARAMETERS
* /robot_description: <?xml version="1....
* /rostdistro: kinetic
* /rosversion: 1.12.17
* /use_sim_time: True

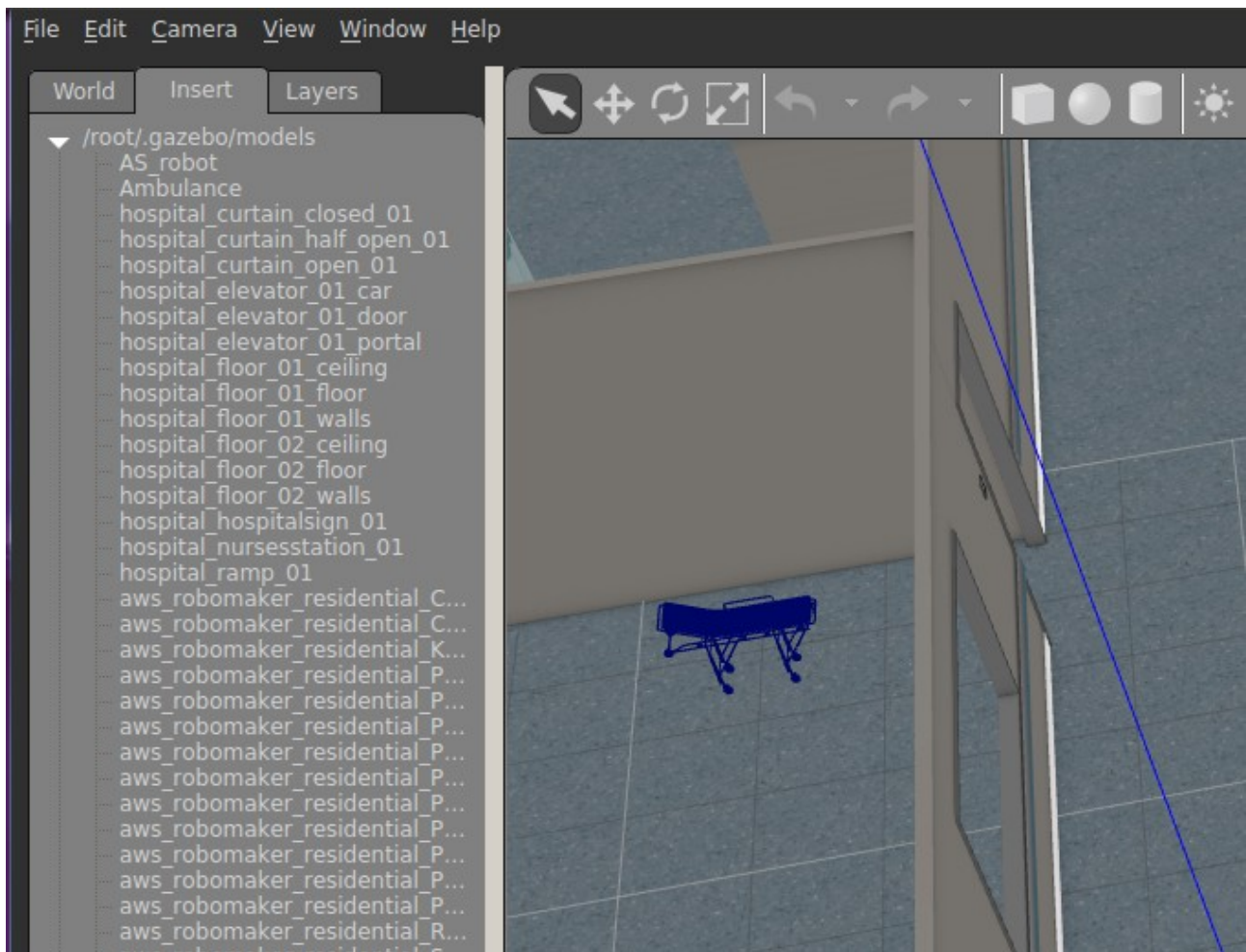
NODES
/
  gazebo (gazebo_ros/gzserver)
  gazebo_gui (gazebo_ros/gzclient)
  spawn_urdf (gazebo_ros/spawn_model)
```

11. Cuando inicie gazebo, en el menú de la izquierda desplegamos la sección ‘Models’ y eliminamos el siguiente elemento: as\_burger.



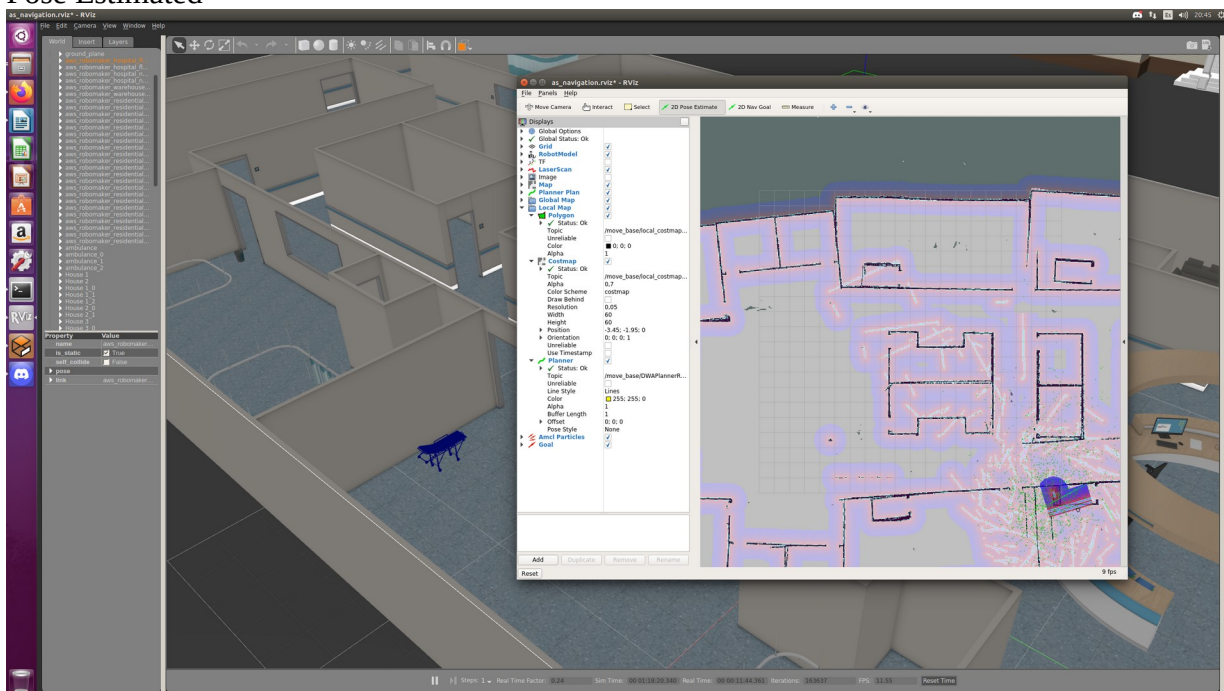
12. En la pestaña ‘Insert’ insertamos nuestro modelo de camilla: AS\_robot





13. Ahora ya podemos correr navigation: `roslaunch as_navigation as_navigation.launch`

Una vez iniciado tenemos que marcar en rviz dónde se encuentra nuestro robot, seleccionando 2D Pose Estimated



Una vez tenemos esto ya podemos mover nuestro robot autónomo marcandole el punto al que queremos viajar mediante 2D

