Manual de instalacion de Ubuntu para trabajar con CRUMB

Sergio González Muriel

17 de enero de 2019

Índice

1. Instalación de Ubuntu 14.04.5 LTS

Primero tendremos que instalar el sistema operativo Ubuntu 14.04 LTS en una partición de nuestro sistema operativo:

1. Descargar Ubuntu 14.04.5 LTS (Trusty Tahr) desktop image de la pagina oficial de Ubuntu - *http://releases.ubuntu.com/14.04/* [8]
2. Descargar Rufus desde *https://rufus.ie/* [1]
3. Convertir un dispositivo USB en uno de arranque con Rufus utilizando el .iso anteriormente descargado
4. Crear una partición vacía (con 50GB de almacenamiento bastaría)
5. Instalar el sistema operativo en la partición ejecutando el dispositivo USB

2. Actualización de cmake

Ahora debemos actualizar cmake, debido a que vamos a trabajar con V-REP 3.4, el cual necesita como mínimo cmake 3.0, mientras que Ubuntu 14.04 LTS trae por defecto cmake 2.8 [1]

```
sudo apt-get remove cmake cmake-data
sudo apt-get install cmake3 cmake3-data
```

3. Instalación y preparación de ROS Indigo

Ahora que tenemos nuestro sistema operativo y nuestro cmake, vamos a instalar ROS.

1. Seguimos las indicaciones del siguiente tutorial, instalando ros-indigo-desktop-full: http://wiki.ros.org/indigo/Installation/Ubuntu [3] Si da error despues del install, debemos ejecutar

```
sudo apt-get update
sudo apt-get -f upgrade
sudo apt-get autoremove
```

[1] Códigos escogidos de [9]

2. Creamos el directorio de trabajo siguiendo el tercer punto del siguiente tutorial (catkin):
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment [4]

3. Copiamos los siguientes paquetes del netbook a la carpeta *src* del directorio de trabajo:

   - arbotix_ros
   - kobuki_testing
   - widow_arm
   - widow_arm_testing

4. Vamos a la carpeta *path/to/catkin_ws/src/arbotix_ros/arbotix_python/bin* y le damos permiso de usuario a los archivos de la carpeta con el siguiente comando

```
sudo chmod +x *
```

5. Instalamos varios paquetes necesarios con la siguiente instrucción:

```
sudo apt-get install ros-indigo-hokuyo-node ros-indigo-
   kobuki ros-indigo-kobuki-core ros-indigo-kobuki-
   desktop ros-indigo-kobuki-msgs ros-indigo-yocs-msgs
   ros-indigo-yujin-ocs ros-indigo-freenect-camera ros-
   indigo-freenect-launch ros-indigo-freenect-stack ros-
   indigo-map-store ros-indigo-turtlebot ros-indigo-
   turtlebot-apps ros-indigo-turtlebot-create ros-indigo-
   turtlebot-create-desktop ros-indigo-turtlebot-msgs ros
   -indigo-turtlebot-simulator ros-indigo-turtlebot-
   dashboard ros-indigo-turtlebot-interactive-markers ros
   -indigo-turtlebot-rviz-launchers ros-indigo-widowx-arm
    ros-indigo-widowx-arm-controller ros-indigo-arbotix-
   msgs
```

6. Nos movemos a la carpeta donde está nuestro espacio de trabajo, en este caso /catkin_ws y ejecutamos

```
catkin-make
```

7. Para mayor comodidad, ponemos el comando "source devel/setup.bash.[en] el bash de Linux, para no tener que ejecutarlo en cada terminal:

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

4. Instalación de simuladores

Ahora que ya hemos instalado Ubuntu y la distribución ROS pertinente, procederemos a instalar los simuladores, para poder realizar las pruebas con el robot sin que éste sufra.

4.1. Instalación de Gazebo ROS

Para la instalación de este simulador, vamos a proceder a seguir la guía proporcionada por Marina Aguilar Moreno en su TFM. En concreto, vamos a seguirla desde el punto A.2 (Página 87 del trabajo) [5]
Como el trabajo mencionado no es público, vamos a mencionar en el propio apartado los puntos a los que nos referimos:

```
$ source devel/setup.bash
```

Para comprobar que el espacio de trabajo está correcto, en el entorno *ROS_PACKAGE_PATH* debe incluir el directorio:

```
$ echo $ROS_PACKAGE_PATH
```

Con lo que debería obtenerse:

```
/home/marina/catkin_ws/src:/opt/ros/indigo/share:/opt/ros/
    indigo/stacks
```

Si se desea más información se puede consultar la página de ROS correspondiente a la instalación [37].

A.2. Instalación y configuración de Gazebo

1. Comprobar que Gazebo funciona correctamente sólo.

```
$ gazebo
```

Con lo que debería obtenerse la pantalla principal de Gazebo.

2. Comprobar que la versión está instalada correctamente escribiento:

```
$ which gzserver
$ which gzclient
```

Con lo que debería obtenerse:

```
/usr/bin/gzserver
/usr/bin/gzclient
```

3. Instalar *gazebo-ros-packages*:

```
$ sudo apt-get install ros-indigo-gazebo-ros-pkgs ros-indigo
    -gazebo-ros-control
```

4. Comprobar integración ROS/Gazebo. Para ello primero se abrirá el núcleo de ROS:

```
$ source /opt/ros/indigo/setup.bash
$ roscore
```

Abrir otra terminal e iniciar Gazebo desde ROS:

```
$ source /opt/ros/indigo/setup.bash
$ rosrun gazebo_ros gazebo
```

Una vez ejecutado debería abrirse la ventana de Gazebo. Finalmente para comprobar las conexiones entre ROS y Gazebo, abrir otro terminal y escribir:

```
$ rostopic list
```

Con lo que debería obtenerse:

```
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/rosout
/rosout_agg
```

Y en la misma terminal escribir:

```
$ rosservice list
```

Y se obtienen los servicios de ROS y Gazebo:

```
/gazebo/apply_body_wrench
/gazebo/apply_joint_effort
/gazebo/clear_body_wrenches
/gazebo/clear_joint_forces
/gazebo/delete_model
/gazebo/get_joint_properties
/gazebo/get_link_properties
/gazebo/get_link_state
/gazebo/get_loggers
/gazebo/get_model_properties
/gazebo/get_model_state
/gazebo/get_physics_properties
/gazebo/get_world_properties
/gazebo/pause_physics
/gazebo/reset_simulation
/gazebo/reset_world
/gazebo/set_joint_properties
/gazebo/set_link_properties
/gazebo/set_link_state
/gazebo/set_logger_level
/gazebo/set_model_configuration
```

```
/gazebo/set_model_state
/gazebo/set_parameters
/gazebo/set_physics_properties
/gazebo/spawn_gazebo_model
/gazebo/spawn_sdf_model
/gazebo/spawn_urdf_model
/gazebo/unpause_physics
/rosout/get_loggers
/rosout/set_logger_level
```

Con esto queda instalado Gazebo y conectado con ROS.

A.3. Instalación Turtlebot

Como se ha indicado a lo largo de la memoria, CRUMB utilizará los paquetes de Turtlebot tal y como aparecen en el repositorio de ROS. Aunque se han realizado algunas modificaciones, éstas se encuentran en los ficheros adjuntos al proyecto relativos a CRUMB.

Así pues, para instalar los paquetes de Turtlebot, únicamente hay que descargarlos desde la terminal:

```
$ sudo apt-get install ros-indigo-turtlebot ros-indigo-turtlebot
  -apps ros-indigo-turtlebot-interactions ros-indigo-turtlebot-
  simulator ros-indigo-kobuki-ftdi ros-indigo-rocon-remocon ros
  -indigo-rocon-qt-library ros-indigo-ar-track-alvar-msgs
```

Una vez instalados ya se puede visualizar Turtlebot desde Gazebo:

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

Con lo que debería obtenerse la Figura 6.1.

Y para manejarla desde teclado, abrir otra terminal:

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

A.4. Instalación CRUMB

Finalmente, hay que descagar los paquetes de CRUMB y compilarlos desde la terminal. Para ello, guardar los paquetes en la carpeta *catkin_ws/src* creada previamente y pegar la carpeta CRUMB, el *plugin roboticsgroup_gazebo_plugins* y el paquete *hello_turtlebot* que contiene la librería para navegación. Además, debería descargarse el paquete *arbotix_ros* de github y pegarlo en la carpeta src junto con CRUMB.

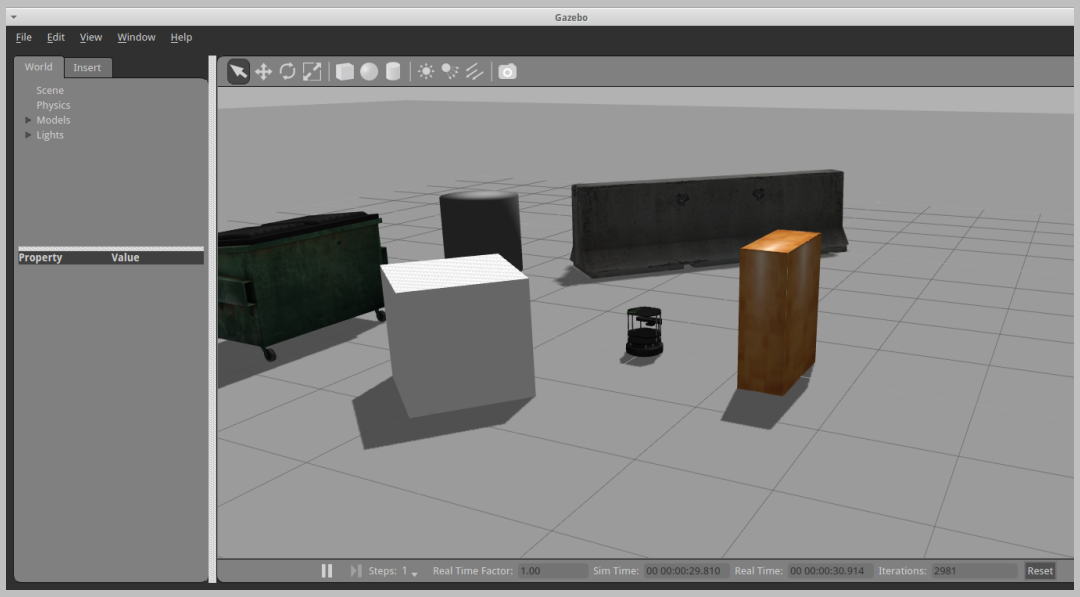


Figura 6.1: Simulación Turtlebot en Gazebo.

Los paquetes de CRUMB se encuentran en el repositorio *GitHub* [38].

Una vez que se tengan los paquetes, hay que compilarlos desde la ventana terminal:

```
$ cd ~/catkin_ws/
$ catkin_make
```

Ya debería poder ejecutarse en el simulador Gazebo al robot CRUMB:

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch crumb_gazebo crumb_world.launch
```

Ejecutándo ese código debería obtenerse la Figura 6.2.

Para trabajar con el simulador sólo hay que darle al botón de *play* y cargar alguna aplicación, como se verá en el *Anexo 2*.

Para ver todos los *topics* que se han incluido en CRUMB sólamente hay que introducir por la terminal:

```
$ rostopic list
```

Estos *topics* se pueden dividir en 3 grupos, dependiendo de si son relativos al brazo, a la plataforma robótica o a ambos.

- *Topics* para el brazo
  - /arm_1_joint/command

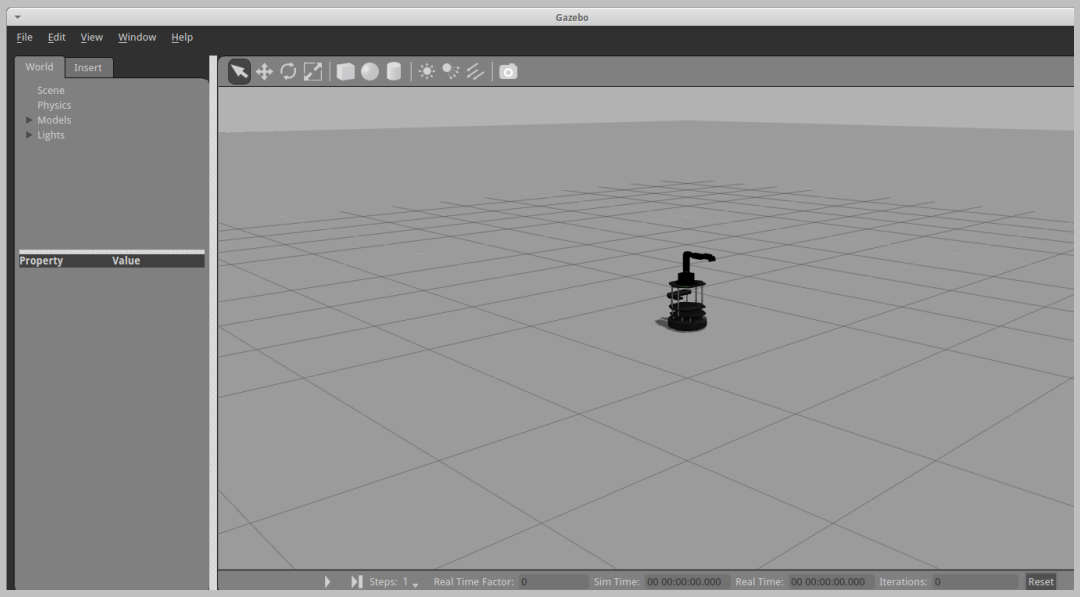


Figura 6.2: Simulación CRUMB en Gazebo.

- /arm_1_joint/pid/parameter_descriptions
- /arm_1_joint/pid/parameter_updates
- /arm_1_joint/state
- /arm_2_joint/command
- /arm_2_joint/pid/parameter_descriptions
- /arm_2_joint/pid/parameter_updates
- /arm_2_joint/state
- /arm_3_joint/command
- /arm_3_joint/pid/parameter_descriptions
- /arm_3_joint/pid/parameter_updates
- /arm_3_joint/state
- /arm_4_joint/command
- /arm_4_joint/pid/parameter_descriptions
- /arm_4_joint/pid/parameter_updates
- /arm_4_joint/state
- /arm_5_joint/command
- /arm_5_joint/pid/parameter_descriptions

- /arm_5_joint/pid/parameter_updates
- /arm_5_joint/state
- /gripper_1_joint/command
- /gripper_1_joint/pid/parameter_descriptions
- /gripper_1_joint/pid/parameter_updates
- /gripper_1_joint/state

- *Topics* para la plataforma
  - /camera/depth/camera_info
  - /camera/depth/image_raw
  - /camera/depth/points
  - /camera/parameter_descriptions
  - /camera/parameter_updates
  - /camera/rgb/camera_info
  - /camera/rgb/image_color
  - /camera/rgb/image_color/compressed
  - /camera/rgb/image_color/compressed/parameter_descriptions
  - /camera/rgb/image_color/compressed/parameter_updates
  - /camera/rgb/image_color/compressedDepth
  - /camera/rgb/image_color/compressedDepth/parameter_descriptions
  - /camera/rgb/image_color/compressedDepth/parameter_updates
  - /camera/rgb/image_color/theora
  - /camera/rgb/image_color/theora/parameter_descriptions
  - /camera/rgb/image_color/theora/parameter_updates
  - /cmd_vel_mux/active
  - /cmd_vel_mux/input/navi
  - /cmd_vel_mux/input/safety_controller
  - /cmd_vel_mux/input/teleop
  - /cmd_vel_mux/parameter_descriptions
  - /cmd_vel_mux/parameter_updates
  - /depthimage_to_laserscan/parameter_descriptions

- /depthimage_to_laserscan/parameter_updates
- /laserscan_nodelet_manager/bond
- /mobile_base/commands/led1
- /mobile_base/commands/led2
- /mobile_base/commands/motor_power
- /mobile_base/commands/reset_odometry
- /mobile_base/commands/velocity
- /mobile_base/events/bumper
- /mobile_base/events/cliff
- /mobile_base/events/wheel_drop
- /mobile_base/sensors/bumper_pointcloud
- /mobile_base/sensors/core
- /mobile_base/sensors/imu_data
- /mobile_base/sensors/imu_data_raw
- /mobile_base_nodelet_manager/bond
- /odom

- *Topics* compartidos

  - /clock
  - /gazebo/link_states
  - /gazebo/model_states
  - /gazebo/parameter_descriptions
  - /gazebo/parameter_updates
  - /gazebo/set_link_state
  - /gazebo/set_model_state
  - /joint_states
  - /rosout
  - /rosout_agg
  - /scan
  - /simulation/noise
  - /simulation/pose

- /simulation/real_time
- /simulation/sim_time
- /tf
- /tf_static

Anexo B. Manual de usuario

B.1. Cargar CRUMB en Gazebo y moverlo

Una vez que se han realizado los apartados anteriores se puede utilizar el modelo para realizar aplicaciones.

Como se ha visto en el Capítulo 3, existe la posibilidad de simular CRUMB con ruido o sin ruido, para ello se puede elegir una de las dos opciones que hay a continuación:

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch crumb_gazebo crumb_world.launch
```

Si se ejecutan estas líneas se obtiene el modelo del robot en Gazebo sin ruido.

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch crumb_gazebo crumb_world_noise.launch
```

En cambio, si se ejecutan estos comandos se abrirá Gazebo y el robot tendrá ruido en los sensores.

Si se quiere mover el robot, primero hay que pulsar *play* en la ventana de Gazebo y, a continuación, escribir en una nueva terminal los siguientes comandos de movimiento:

```
$ . ~/catkin_ws/devel/setup.bash
$ rosrun crumb_home crumb_home
```

Con estos comandos el brazo se coloca en su posición *home*. Esto es útil cada vez que se ejecute un movimiento con el brazo para volverlo a poner en su posición inicial.

En cambio si se quiere mover la plataforma Turtlebot, escribiendo estos comandos en una nueva terminal es posible teleoperarla desde teclado.

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

B.2. Cargar CRUMB en Gazebo y coger objetos

Para que CRUMB pueda coger objetos del escenario, primero hay que incluirlos. En el fichero *crumb_prueba1.launch* se encuentra la información para recrear el escenario del segundo experimento de este Trabajo Fin de Máster.

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch crumb_gazebo crumb_prueba1.launch
```

Si se ejecuta este código se obtiene la Figura 5.1. En él se encuentra el robot delante de dos objetos con los que puede interactuar. Para ejecutar los movimientos primero dar a *play* en la ventana de Gazebo.

Se pueden realizar los movimientos de los objetos de manera automática o seleccionando los que se desee mover en cada instante.

Con los programas *crumb_cajas_mcd* y *crumb_cajas_mci* se realizará el movimiento de manera automática. La diferencia entre estos dos ficheros es que el primero utiliza los valores de las articulaciones para indicar los movimientos. En cambio, el segundo calcula los valores articulares a partir del modelo cinemático inverso.

```
$ . ~/catkin_ws/devel/setup.bash
$ rosrun crumb_home crumb_cajas_mcd
```

```
$ . ~/catkin_ws/devel/setup.bash
$ rosrun crumb_home crumb_cajas_mci
```

Con el programa *crumb_cajas_selector* se tiene la posibilidad de elegir el objeto que se desea mover. Este programa utiliza también el modelo cinemático inverso.

```
$ . ~/catkin_ws/devel/setup.bash
$ rosrun crumb_home crumb_cajas_selector
```

B.3. Cargar aplicación navegación y objetos

Introduciendo en la terminal los siguientes comandos se puede abrir en Gazebo el escenario donde CRUMB irá hasta un objeto, lo cogerá y lo llevará a otro sitio.

```
$ . ~/catkin_ws/devel/setup.bash
$ roslaunch crumb_gazebo crumb_pick_place.launch
```

Una vez Gazebo se haya abierto y se halla pulsado *play* en la ventana, se cargará el fichero *.cpp* con la aplicación. Para ello se introducirá lo siguiente en una nueva terminal:

4.1.1. Posibles errores

- En éste primer apartado A.2, al ejecutar gazebo nos puede aparecer el error *namespace not found*. Para arreglar este error debemos ejecutar los siguientes comandos: [2]

```
sudo sh -c 'echo "deb deb http://packages.osrfoundation.
   org/gazebo/ubuntu trusty main" > /etc/apt/sources.list
   .d/gazebo-latest.list
wget http://packages.osrfoundation.org/gazebo.key -O - |
   sudo apt-key add -
sudo apt update
sudo apt upgrade
echo "export LC_NUMERIC=C" >> ~/.bashrc
source ~/.bashrc
```

- En el apartado A.3, se puede quedar la pantalla de Gazebo en negro. Eso es debido a que la anterior ejecución de éste no se interrumpió correctamente. Para arreglarlo ejecutar

```
killall gzserver
```

- En el apartado A.4, puede que no aparezcan los *topics* del brazo al ejecutar *rostopic list*, apareciendo errores al ejecutar la simulación de Gazebo de *effort-controllers*. Para arreglarlo ejecutar

```
sudo apt-get install ros-indigo-ros-controllers
```

- En al apartado B.2, al ejecutar el fichero *crumb_prueba1.launch*, puede que nos aparezca un error de permisos en /home/marina. Para arreglarlo

  1. Ir a la carpeta /catkin_ws/src/crumb/crumb_gazebo/worlds
  2. Abrir tres_cajas.world con un editor de texto
  3. Dos opciones:
     - Cambiar en la linea 86 */home/marina/vídeos* por una dirección de tu partición.
     - Cambiar en la linea 85 *save enabled* a *false* (Con esta segunda opción no se guardara ningún vídeo de la simulación)

[2] Código recogido de [6]

4.2. Instalación de VREP

Para instalar este simulador en Ubuntu 14.04 LTS y con las prestaciones del plugin RosInterface, debemos seguir los siguientes pasos:

1. Desgarcar la versión 3.4.0 PRO EDU de la página web *www.coppeliarobotics.com/previousversions.html* [12]

2. Descomprimir

3. Descargar modelos de la página web de CRUMB [2], moviéndonos a la carpeta deseada y ejecutando

```
git clone https://github.com/CRUMBproject/V-REP.git
```

4. Entrar a la carpeta *V-REP_PRO_EDU_V3_4_0_Linux/programming/ros_packages* y copiar la carpeta *v_repExtRosInterface* en el *src* de nuestro directorio de trabajo de catkin

5. Inicializar la variable de entorno *VREP_ROOT*

```
echo 'export VREP_ROOT="path/to/vrep/V-REP_PRO_EDU_V3
   _4_0_Linux"' >> ~/.bashrc
```

6. Ejecutar en la carpeta *catkin_ws*

```
catkin build
```

   - Si da fallos, debido a que ya existen las carpetas build y devel, eliminarlas y volver a ejecutar el comando

5. Preparación para usar el WidowX arm

Para poder utilizar el brazo, primero hay que realizar una serie de preparativos.

1. Conectamos el robot al puerto ttyUSB0 de nuestro ordenador. (Lo comprobamos con el comando *ls -l /dev/* y viendo que al conectar el USB del brazo aparece */dev/ttyUSB0*)

2. Le damos permiso de lectura al puerto

```
sudo chmod a+rw /dev/ttyUSB0
```

3. Vamos a la carpeta *catkin_ws/src/arbotix_ros/arbotix_python/bin* y ejecutamos

```
./arbotix_terminal
```

4. Ejecutamos el comando *ls* y comprobamos que reconoce todas las articulaciones del brazo (Deben aparecer del 1 al 6 consecutivamente)

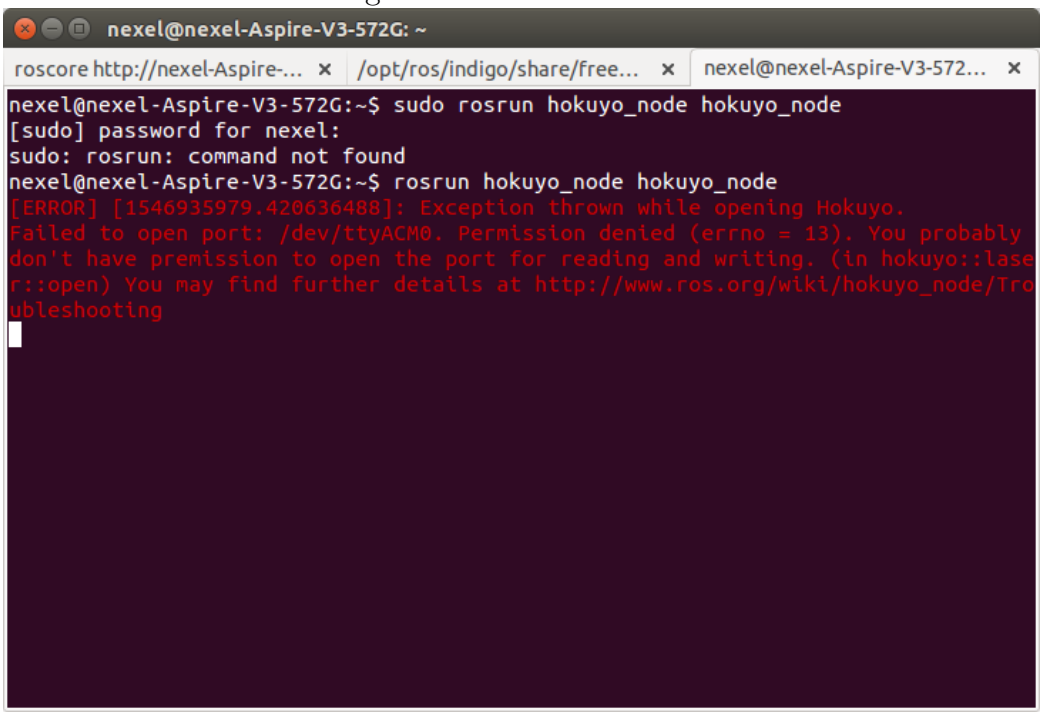
   - Si no aparecen todas, hay que seguir los manuales de las páginas web *https://learn.trossenrobotics.com/arbotix/arbotix-quick-start.html* [10] y *https://learn.trossenrobotics.com/arbotix/1-using-the-tr-dynamixel-servo-tool#&panel1-1* [11], con unas variaciones:

     - En el segundo punto de la segunda página no hace falta instalar nada, ya que Ubuntu tiene los FTDI necesarios por defecto
     - Para que DynaManager reconozca las articulaciones, hay que conectarlas una a una a la placa.

6. Configuración del láser

El láser del robot también nos puede dar el siguiente fallo al ejecutarlo por primera vez:

Figura 1: Error con el láser



Ésto se arregla siguiendo el punto 0.1 de la página web *http://wiki.ros.org/hokuyo_node/Tutorials/UsingTheHokuyoNode* [7]

A. Apéndice

Referencias

[1] Pete Batard. *Rufus. Cree unidades USB arrancables fácilmente.* URL: https://rufus.ie/ (visitado 16-01-2019).

[2] CRUMBproject. *CRUMBproject/V-REP.* URL: https://github.com/CRUMBproject/V-REP.git (visitado 16-01-2019).

[3] CesarHoyosM. *Ubuntu install of ROS Indigo.* URL: http://wiki.ros.org/indigo/Installation/Ubuntu (visitado 16-01-2019).

[4] Marguedas. *Installing and Configuring Your ROS Environment.* URL: http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment (visitado 16-01-2019).

[5] Marina Aguilar Moreno. "Modelado y simulación de un robot TurtleBot 2 con brazo manipulador WidowX sobre ROS y Gazebo". Tesis de lic. Universidad de Málaga, dic. de 2016.

[6] Stefano Primatesta. *Problem with Indigo and Gazebo 2.2.* URL: https://answers.ros.org/question/199401/problem-with-indigo-and-gazebo-22/ (visitado 16-01-2019).

[7] Isaac Saito. *How to use Hokuyo Laser Scanners with the hokuyo_node.* URL: http://wiki.ros.org/hokuyo_node/Tutorials/UsingTheHokuyoNode (visitado 16-01-2019).

[8] Ubuntu. *Ubuntu 14.04.5 LTS (Trusty Tahr).* URL: http://releases.ubuntu.com/14.04/ (visitado 16-01-2019).

[9] ellen. *CMake Error: Could not find CMAKE_ROOT?* URL: https://askubuntu.com/questions/1014670/cmake-error-could-not-find-cmake-root (visitado 16-01-2019).

[10] learn.trossenrobotics.com. *ArbotiX-M Robocontroller Getting Started Guide.* URL: https://learn.trossenrobotics.com/arbotix/arbotix-quick-start.html (visitado 16-01-2019).

[11] learn.trossenrobotics.com. *Setting DYNAMIXEL IDs with the Dyna-Manager.* URL: https://learn.trossenrobotics.com/arbotix/1-using-the-tr-dynamixel-servo-tool\#\&panel1-1 (visitado 16-01-2019).

[12] Coppelia robotics. *Previous Versions of V-REP.* URL: www.coppeliarobotics.com/previousversions.html (visitado 16-01-2019).