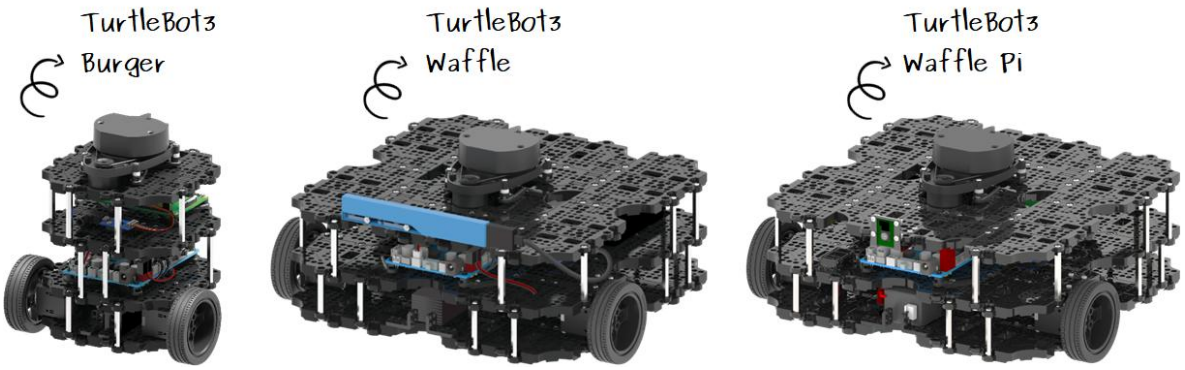
MANUAL DE USUARIO DEL ROBOT MOVIL TURTLEBOT3 MODELO BURGER

Tabla de contenido

1. DESCRIPCIÓN DE LA PLATAFORMA

TurtleBot3 es la plataforma mas reciente del conjunto de plataformas TurtleBot que existen, estas plataformas están en constante evolución y son accesibles gracias a su software de código abierto y bajo costo. Para TurtleBot3 existen tres versiones; waffle, waffle pi y burger que es la que se describe en este manual.

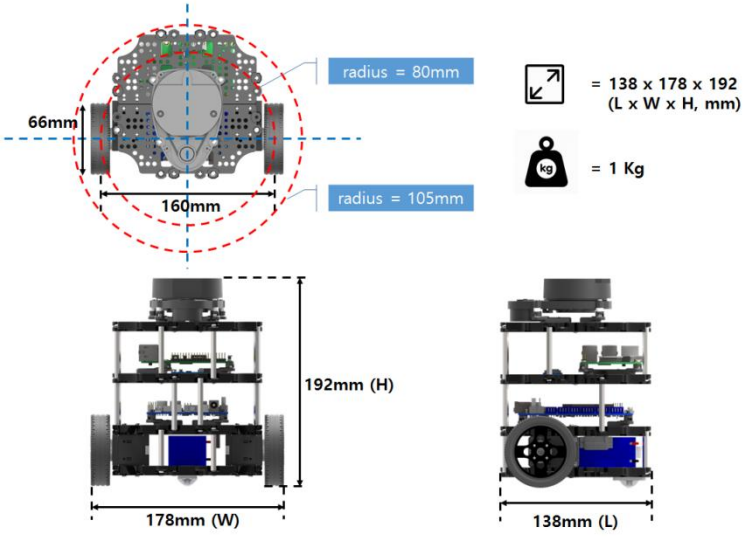


1.1 Características

Las características dependen de la versión de TurtleBot3, para este caso se pone en funcionamiento el modelo burger y se muestra toda la descripción del mismo.

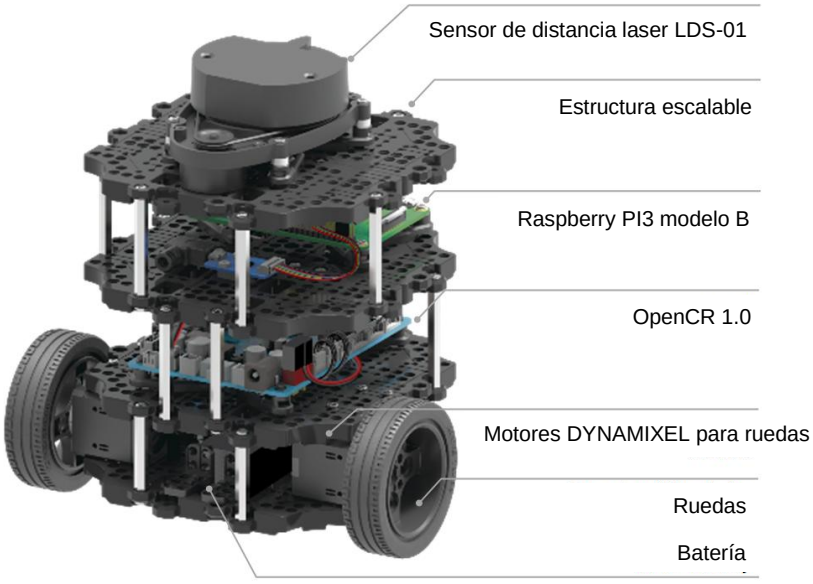
CARACTERISTICAS	
Máxima velocidad	0.22 m/s
Máxima velocidad de rotación	2.84 rad/s
Tiempo de funcionamiento esperado	2h 30m
Tiempo de carga previsto	2h 30m
Unidad de medición inercial	Giroscopio, acelerómetro, magnetómetro
Conectores de alimentación	3.3V/800mA 5V/4A 12V/1ª
Audio	Secuencias de pitidos programables
Batería	Polímero de litio 11.1V
Conexión a PC	USB
Adaptador de corriente	Entrada: 100-240V, CA 50/60Hz, 1.5A Salida: 12V DC, 5A
Sensor de distancia Laser (LDS)	Sensor de distancia laser 360 LDS-01
Pines	GPIO 18 pines Arduino 32 pines

1.2 Dimensiones



1.3 Componentes

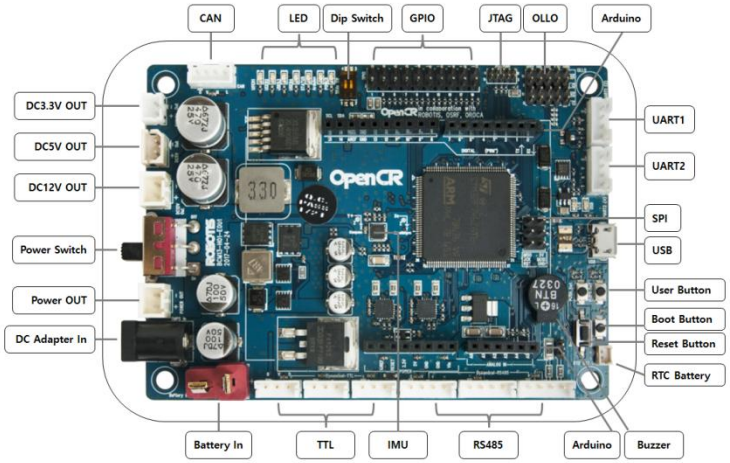
Al ensamblar la plataforma TurtleBot3 modelo burger quedo distribuida en cuatro niveles. Comenzando de abajo hacia arriba el primer nivel es para la batería y los motores que cuentan también con las ruedas; en el segundo nivel se encuentra la tarjeta OpenCR; en el tercer nivel se encuentra la Raspberry PI3 y en el ultimo nivel se encuentra el sensor de distancia laser LDS-01.



1.3.1 Motores DYNAMIXEL XL430

CARACTERISTICAS	
Motor	Húmedo
Velocidad de transmisión	9600[bps] ~ 4.5 [Mbps]
Resolución	4096[pulso/rev]
Temperatura de funcionamiento	-5 ~ +72 [°C]
Voltaje de entrada	6.5 ~ 12 [V] (recomendado: 11.1V)
Conexión	Bus multipunto TTL

1.3.2 Tarjeta OpenCR 1.0



OpenCR es una tarjeta incorporada a la plataforma turtlebot3 que proporciona pines de entrada/salida digitales y analógicos que pueden interactuar con la Raspberry PI3 o varios sensores. También cuenta con salidas de potencia de 12V, 5V, 3.3V para la Raspberry Pi3 y sensores.

CARACTERISTICAS	
Microcontrolador	STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS)
Sensores	Giroscopio, acelerómetro, magnetómetro
Conectores	32 pines (L 14, R18)*Conectividad Arduino Módulo sensor x4 pines Conector de extensión x18 pines
Comunicación	USB, TTL, RS485, UART x2, CAN

1.3.3 Raspberry PI3

CARACTERISTICAS	
RAM	1GB
Pines	GPIO extendido de 40 pines
Puertos	USB x2, HDMI, Puerto de cámara CSI, Puerto micro SD
Fuente de alimentación	Micro USB conmutada actualizada hasta 2.5A

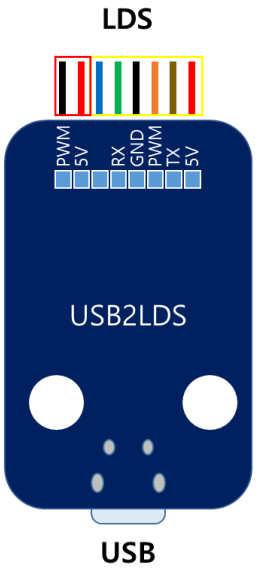
1.3.4 Sensor Laser LDS-01

El sensor laser LDS-01 es uno de los sensores incorporados a turtlebot3; este sensor es capaz de recopilar información alrededor del

robot gracias a que gira 360°, además es accesible gracias a su fácil conexión y configuración.

CARACTERISTICAS	
Voltaje de operación	5V DC
Consumo de corriente	400mA o menos
Corriente de acometida	1A
Tasa de muestreo	1.8KHz
Distancia de detección	120mm ~ 3500mm
Exactitud de distancia (120mm ~ 499mm)	±15mm
Exactitud de distancia (50mm ~ 3500mm)	±5%
Precisión de distancia (120mm ~ 499 mm)	±10mm
Precisión de distancia (500mm ~ 3500mm)	±3.5%
Rango angular	360°
Resolución	1°

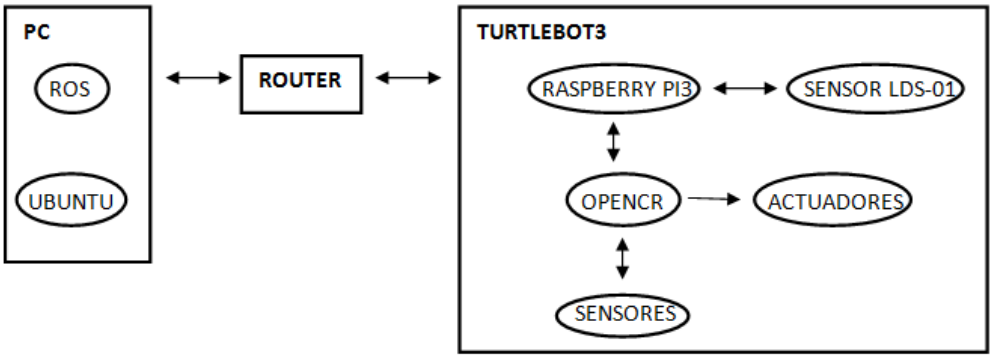
En la plataforma turtlebot3 el sensor se conecta por medio de un cable USB a la Raspberry PI3 gracias al conecto USB2LDS con el que cuenta el sensor.



Desde el sensor hay 8 cables, dos del motor que hace que gire el sensor y del sensor mismo. Los dos cables encerrados en el recuadro rojo hacen que el motor funcione y los encerrados en el recuadro amarillo son los del funcionamiento del sensor. El cable azul no tiene un nombre ya que este cable se utiliza solamente si la turtlebot3 cuenta con la actualización para que el sensor comience su funcionamiento desde que se enciende el robot.

2. ARQUITECTURA DE COMUNICACIÓN

Para comenzar con el manejo de la plataforma móvil TurtleBot3 es necesario conocer la manera en que se comunica con la PC para que se pueda hacer la configuración, manejo y programación de la plataforma. Para explicar la arquitectura de comunicación se tiene el siguiente diagrama:



TurtleBot3 se comunica con la PC por medio de WI-FI por eso el intermediario entre estos dos es un router al cual están conectados tanto la PC, como la plataforma. Las flechas en el diagrama indican que la PC y el robot pueden enviar o recibir información. Dentro de cada uno de estos hay más componentes que ayudan a que la plataforma funcione. La PC cuenta con dos sistemas operativos, ROS que es el sistema operativo que se utiliza para el funcionamiento de la plataforma y este es instalado sobre la distribución Ubuntu que es uno de los sistemas operativos de Linux. TurtleBot3 cuenta con dos tarjetas, una que es la Raspberry PI3 que también es un intermediario de comunicación, sin ella no sería posible que el robot se conecte al router por medio de WI-FI, además a ella están conectadas la tarjeta OpenCR y el sensor laser LDS-01 y por medio de ella este sensor y la tarjeta envían o reciben información; la tarjeta OpenCR controla los actuadores y sensores que tiene incorporados la tarjeta o externos que sean conectados por el usuario.

3. REQUERIMIENTOS

Como se mencionó anteriormente el manejo de la plataforma requiere de ROS y Ubuntu en la PC desde la cual se hace todo el proceso de configuración y funcionamiento. Para este manual la instalación de Ubuntu y ROS se hizo en un computador totalmente formateado; sin embargo esta instalación se puede hacer en una PC que tenga otro sistema operativo,

pero se recomienda hacer una partición del disco duro e instalar Ubuntu en esta partición.

3.1 Ubuntu

Para la instalación de Ubuntu se requiere que la PC tenga un procesador de 1GHz o mejor, RAM de 1.5 GB o más, en el disco duro se necesita tener disponibilidad de 7GB y contar con conexión a internet.

Se recomienda además instalar Ubuntu 16.04 LTS ya que es compatible con la versión de ROS y con la Raspberry PI3.

3.2 Raspberry PI3 y OpenCR

Para la instalación del sistema operativo de Raspberry y la configuración de la tarjeta OpenCR se requiere tener la plataforma totalmente ensamblada ya que así es más fácil conectar el voltaje de alimentación de ambas tarjetas (el manual de ensamblado se incluye con la plataforma), también se requiere tener conexión a internet.

Para la Raspberry PI3 se requiere una memoria micro SD no menor a 8GB y un adaptador para la misma.

4. CONFIGURACIÓN BÁSICA

Se comienza con la configuración de la PC y la plataforma teniendo en cuenta los requerimientos necesarios para cada configuración.

4.1 Configuración de la PC (Linux)

Se descarga Ubuntu 16.04 LTS[1] desde un computador con conexión a internet y luego se instala en el computador desde el cual se va a colocar en funcionamiento la plataforma. Se sugiere el siguiente enlace para descargar la distribución Ubuntu de Linux:

https://ubuntu.com/download/alternative-downloads

[1] El presente manual de usuario fue desarrollado con la distribución Ubuntu 16.04 LTS que usa el kernel de Linux 4.15.0-45-generic y sobre esta distribución el sistema operativo ROS kinetic, se recomienda hacer todos estos pasos con las distribuciones que a la fecha existan. Posiblemente se tenga que hacer actualización de kernel y algunos drivers o es posible que algunos paquetes de ROS no funcionen, por ende se sugiere revisar el foro https://answers.ros.org/questions/ este foro también puede ayudar a resolver problemas al hacer instalación de paquetes de ROS con la distribuciones del presente manual.

4.2 Configuración de ROS

Se debe instalar el sistema operativo ROS Kinetic[2] (recomendado para turtlebot3), para eso en la PC con el sistema operativo Ubuntu debe abrir un terminal, al cual puede acceder fácilmente con la combinación de teclas de acceso directo CTRL + ALT + t; en el terminal debe ejecutar los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ wget https://raw.githubusercontent.com/ROBOTIS-
GIT/robotis_tools/master/install_ros_kinetic.sh && chmod 755
./install_ros_kinetic.sh && bash ./install_ros_kinetic.sh
```

NOTA: Es posible que al ejecutar el ultimo comando se muestre un error o no termine de ejecutarse, se recomienda revisar la conexión a internet y si no se resuelve, revisar el foro sugerido en el pie de página.

Al terminar este paso se debe reiniciar la PC.

Después de la instalación de ROS, se deben instalar los paquetes dependientes de ROS que son necesarios para el manejo de la plataforma turtlebot3; para eso abra un terminal y ejecute el siguiente comando:

```
$ 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-image-view
ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-
interactive-markers
```

Cuando esta instalación finalice, debe ejecutar los siguientes comandos en el mismo terminal:

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ cd ~/catkin_ws && catkin_make
```

NOTA: es importante que el comando CATKIN_MAKE se ejecute sin ningún error ya que con este comando se está modificando el kernel del

[2] Si desea saber más respecto al sistema operativo ROS puede consultar la siguiente pagina web: https://openwebinars.net/blog/que-es-ros/

sistema. Si se presenta algún error con este comando se recomienda consultar el foro de ayuda.[3]

4.3 Configuración Raspberry PI3

El robot turtlebot3 cuenta con una Raspberry PI3, la cual se debe configurar para que la plataforma pueda funcionar adecuadamente. A esta tarjeta se le debe colocar una memoria SD en la cual se debe instalar el sistema operativo (imagen de distribución de Linux basado en rasbian) que se puede descargar desde la PC en el siguiente link:

https://www.robotis.com/service/download.php?no=1738

El sistema operativo se debe descargar solamente de este link, ya que este además cuenta con ROS y los paquetes de ROS necesarios para que turtlebot3 modelo burger funcione.

Para pasar el sistema operativo a la memoria SD, es recomendable instalar en la PC el programa balenaEtcher que se puede descargar del siguiente link:

https://www.balena.io/etcher/

Una vez instalado el programa, ejecútelo y siga los pasos que este mismo indica para grabar la imagen (sistema operativo) en la memoria SD.

4.4 Configuración del router y direcciones IP

Por último ROS necesita la dirección IP de la PC para tener comunicación con TurtleBot3.

NOTA: Debe tener en cuenta que la PC y TurtleBot3 deben estar conectados al mismo router.

[3] Si se tiene algún error al ejecutar alguno de los paquetes de la instalación de paquetes de ROS se sugiere consultar el siguiente foro: https://answers.ros.org/questions/

4.4.1 Router

En el presente manual el router que se utilizo fue un celular[4], si se usa un router como por ejemplo el que da WI-FI a un área determinada y en el que muchas personas se pueden conectar o pueden manipular este router es posible que se tengan problemas a futuro como que la dirección IP cambie. Para evitar este tipo de problemas se utiliza el celular como router.

Para poder hacer esta configuración y la configuración de direcciones IP se debe visualizar la Raspberry. Para eso debe conectar la batería al robot y encender la tarjeta OpenCR esta da alimentación a la Raspberry, luego debe conectar la Raspberry a una pantalla por medio del puerto HDMI y conectar un mouse y un teclado por medio de los puertos USB.

Ahora se debe activar el punto de acceso móvil del celular, es decir activar la opción de compartir datos a otros celular (no es necesario contar con datos, solo es necesario activarlo), al activar esta opción en la PC y en la Raspberry aparece el nombre de el punto de acceso móvil. Tanto en la PC como en la Raspberry debe conectarse a esta señal WI-FI, así ya están conectados los dos dispositivos al mismo router y se puede comenzar a trabajar; además teniendo los dispositivos conectados a este router la dirección IP no va a cambiar así se apague o se reinicie.

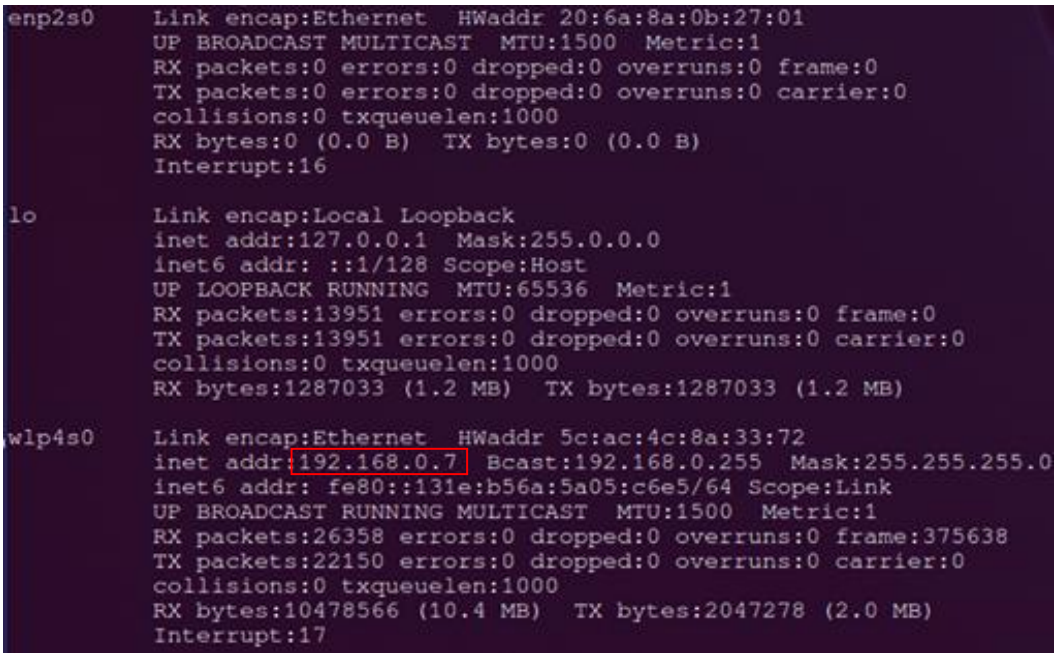
4.4.2 Direcciones IP

Para obtener la dirección IP de la PC, debe ejecutar en una ventana de terminal el siguiente comando:

```
$ ifconfig
```

[4] Para la configuración del router en este manual se utilizo un celular con android 7.0 NRD90M, se recomienda hacer la configuración en versiones de androir mas actualizadas o en un celular que tenga otro tipo de sistema operativo.

Con el anterior comando debe aparecer la siguiente información:



El recuadro rojo representa la IP_PC (dirección IP de la PC) para el caso del manual.

Después debe ejecutar el comando:

```
$ nano ~/.bashrc
```

Este comando abre un archivo el cual se debe modificar, para esto se debe ir al final del archivo y modificar las siguientes dos líneas:

export ROS_MASTER_URI = http://IP_PC:11311
export ROS_HOSTNAME = IP_PC

IP_PC debe cambiarlo por la IP encontrada en la PC que para el caso de este manual es la encerrada en el recuadro rojo.

Para el caso del manual las dos últimas líneas del archivo quedaron de la siguiente manera:

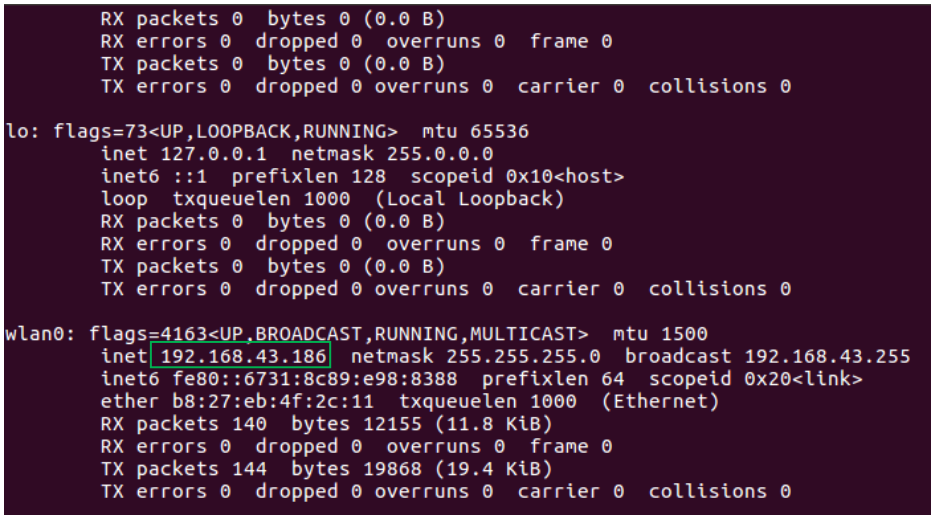
```
export ROS_MASTER_URI=http://192.168.0.7:11311
export ROS_HOSTNAME=192.168.0.7
```

Luego se debe oprimir CTRL + X, escribir Y y oprimir ENTER para que queden guardados los cambios y se cierre el archivo.

Debe ejecutar el siguiente comando para finalizar con la configuración de la dirección IP de la PC:

```
$ source ~/.bashrc
```

Ahora en la Raspberry debe abrir un terminal y ejecutar el comando ifconfig, este comando mostrara la misma información mostrada en la PC al ejecutarlo y también se deberá tener en cuenta la dirección IP que se muestra.



En el mismo terminal debe ejecutar el comando

```
nano ~/.bashrc
```

En este caso también este comando abre un archivo; debe ir al final de este archivo y modificar las siguientes dos líneas:

export ROS_MASTER_URI = http://IP_PC:11311
export ROS_HOSTNAME = IP_RASPBERRY

IP_PC debe cambiarlo por la IP encontrada en la PC que para el caso de este manual es la encerrada en el recuadro rojo anteriormente.

IP_RASPBERRY debe cambiarla por la IP que se encontró al ejecutar el comando ifconfig en la Raspberry PI3 (IP encerrada en el recuadro verde).

Para el caso de este manual, queda de la siguiente manera:

```
export ROS_MASTER_URI=http://192.168.0.7:11311
export ROS_HOSTNAME=192.168.43.186
```

Luego se debe oprimir CTRL + X, escribir Y y oprimir ENTER para que queden guardados los cambios y se cierre el archivo.

Debe ejecutar el siguiente comando para finalizar con la configuración de la dirección IP de la Raspberry:

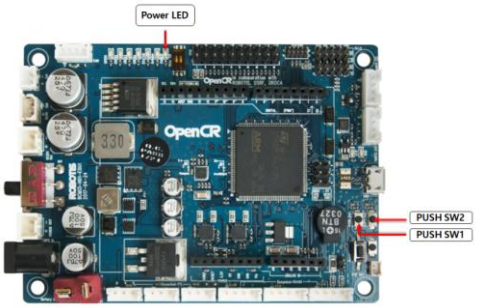
```
$ source ~/.bashrc
```

4.5 Configuración OpenCR

La configuración de la tarjeta OpenCR se puede hacer desde la PC o desde la Raspberry, sin embargo se recomienda hacerla desde la Raspberry ya que el manual de ensamblado de la plataforma deja conectadas entre sí la Raspberry y la tarjeta OpenCR, debido a esto no es necesario hacer conexiones de más y al hacer la configuración solo es necesario encender la tarjeta OpenCR, visualizar la Raspberry y abrir un terminal. Al abrir el terminal debe ejecutar los siguientes comandos:

```
$ export OPENCR_PORT=/dev/ttyACM0
$ export OPENCR_MODEL=burger
$ rm -rf ./opencr_update.tar.bz2
$ wget https://github.com/ROBOTIS-GIT/OpenCR-
Binaries/raw/master/turtlebot3/ROS1/latest/opencr_update.tar.bz2
&& tar -xvf opencr_update.tar.bz2 && cd ./opencr_update &&
./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr && cd ..
```

Para probar si la configuración de OpenCR fue completada sin errores, puede oprimir PUSH SW1 durante unos segundos para que el robot avance hacia adelante y PUSH SW2 también durante unos segundos para que gire 180°.



4.6 Configuración sensor laser LDS-01

Para la configuración del sensor laser, este se debe conectar a la PC directamente. Se debe abrir una ventana de terminal y se debe ejecutar el siguiente comando:

```
$ sudo apt-get install ros-kinetic-hls-lfcd-lds-driver
```

Este comando descarga e instala los drivers necesarios para el funcionamiento del sensor laser. Luego se descarga el controlador del sensor con el comando:

```
$ git clone https://github.com/ROBOTIS-
GIT/hls_lfcd_lds_driver.git
```

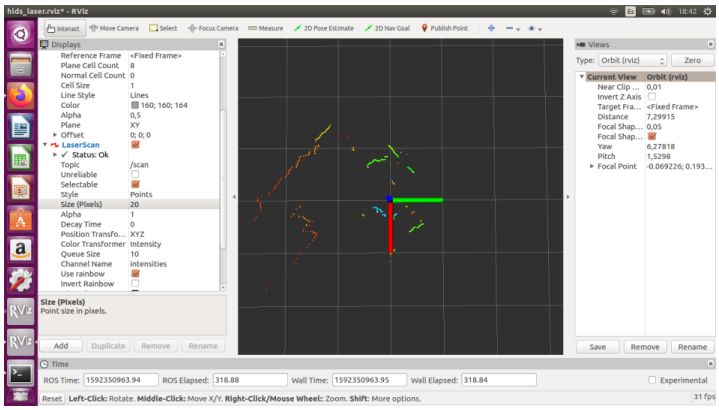
Cuando la instalación de estos dos comandos termine se establecen los permisos en el sistema para que el sensor pueda funcionar. Se establecen con el siguiente comando:

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

Ejecutando estos comandos ya se tiene configurado el sensor para su uso en la TurtleBot3, ahora se debe verificar que si este funcionando el sensor, para eso se debe tener el sensor conectado a la PC y está conectada al router que se usa para la conexión con la plataforma, se abre una ventana de terminal y se ejecuta el comando roscore, luego en otra ventana de terminal se ejecuta el siguiente comando:

```
roslaunch hls_lfcd_lds_driver view_hlds_laser.launch
```

Al ejecutar este comando se abre el simulador de entornos Rviz



Lo que muestra el simulador es el entorno que tiene alrededor el sensor en el momento de la configuración, teniendo esto se puede comprobar que el sensor ya está configurado y está funcionando correctamente[5].

En las ventanas de terminal abiertas anteriormente se ejecuta la combinación de teclas CTRL + c para terminar con el proceso de verificación del sensor, luego se puede conectar el sensor a la Raspberry y hacer que el sensor funcione para cualquier entorno en que se requiera el sensor y el robot.

4.7 Configuración para antes de ejecutar un programa

Este paso es importante ya que se hace antes de ejecutar cualquier programa en ROS. A este paso técnicamente se le conoce como bringup, cuenta con varias instrucciones, estas ejecutadas sin que aparezca ningún error hacen que el robot pueda ser controlado. Primeramente debe asegurarse de que las direcciones IP estén correctamente configuradas (parte 4.4.2 del manual).

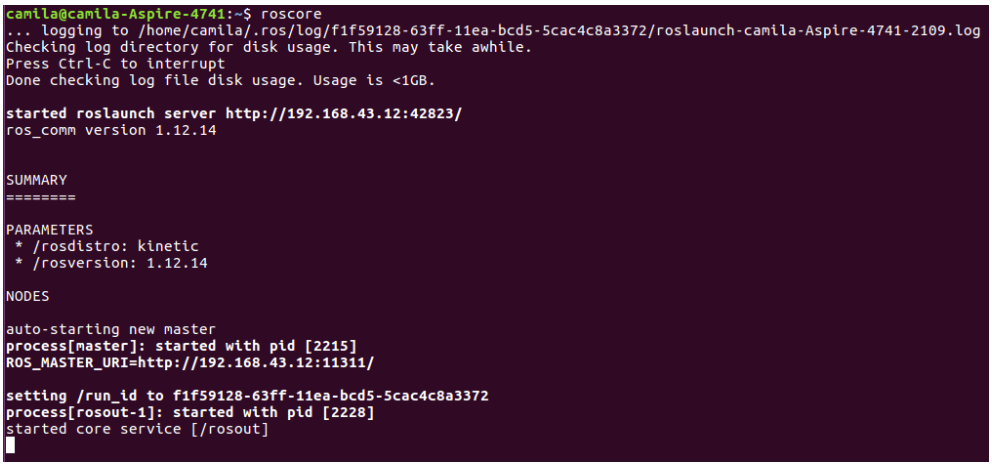
NOTA: No es necesario tener la Raspberry conectada a una pantalla y con un mouse y teclado, solamente es necesario tenerla alimentada.

Para comenzar debe abrir un terminal y ejecutar el comando

```
$ roscore
```

[5] Si al abrir el simulador de entornos Rviz no se ve como la imagen, se recomienda configurar el simulador de entornos desde la barra de opciones del lado izquierdo.

Este comando permite que los nodos y programas con los que cuenta ROS se comuniquen entre sí. Al ejecutar el anterior comando, debe aparecer la siguiente información:



Ahora debe abrir otro terminal con la combinación de teclas CTRL + SHIF + t y acceder a la Raspberry desde la PC a través de SSH, para ello debe ingresar el siguiente comando:

ssh pi@IP_RASPBERRY

IP_RASPBERRY debe reemplazarlo por la dirección IP de la Raspberry encontrada en el apartado 4.4.2 del manual. Para el caso de este manual queda de la siguiente manera:

```
camila@camila-Aspire-4741:~$ ssh pi@192.168.43.186
```

Al ejecutar el comando en el terminal aparece que debe ingresar la clave de la Raspberry; para cualquier caso y en cualquiera de los modelos del robot la clave de la Raspberry es turtlebot; si a accedido a la Raspberry correctamente, se dará cuenta que en el terminal por defecto, ya no aparece el nombre del computador, aparece el nombre Raspberry. Ahora en este mismo terminal debe ingresar el siguiente comando:

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

Este comando abre los paquetes necesarios para iniciar turtlebot3.

Aparecerán los siguientes mensajes en el terminal:

```
pi@raspberrypi:~ $ roslaunch turtlebot3_bringup turtlebot3_robot.launch
... logging to /home/pi/.ros/log/f1f59128-63ff-11ea-bcd5-5cac4c8a3372/roslaunch-raspberrypi-1086.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://192.168.43.186:43425/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.14
 * /turtlebot3_core/baud: 115200
 * /turtlebot3_core/port: /dev/ttyACM0
 * /turtlebot3_core/tf_prefix:
 * /turtlebot3_lds/frame_id: base_scan
 * /turtlebot3_lds/port: /dev/ttyUSB0

NODES
  /
    turtlebot3_core (rosserial_python/serial_node.py)
    turtlebot3_diagnostics (turtlebot3_bringup/turtlebot3_diagnostics)
    turtlebot3_lds (hls_lfcd_lds_driver/hlds_laser_publisher)

ROS_MASTER_URI=http://192.168.43.12:11311

process[turtlebot3_core-1]: started with pid [1095]
process[turtlebot3_lds-2]: started with pid [1096]
process[turtlebot3_diagnostics-3]: started with pid [1097]
[INFO] [1583976217.308886]: ROS Serial Python Node
```

```
ROS_MASTER_URI=http://192.168.43.12:11311

process[turtlebot3_core-1]: started with pid [1281]
process[turtlebot3_lds-2]: started with pid [1282]
process[turtlebot3_diagnostics-3]: started with pid [1283]
[INFO] [1587329082.213178]: ROS Serial Python Node
[INFO] [1587329082.757367]: Connecting to /dev/ttyACM0 at 115200 baud
[INFO] [1587329084.959111]: Note: publish buffer size is 1024 bytes
[INFO] [1587329084.963516]: Setup publisher on sensor_state [turtlebot3_msgs/SensorState]
[INFO] [1587329085.000819]: Setup publisher on firmware_version [turtlebot3_msgs/VersionInfo]
[INFO] [1587329085.112844]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1587329085.165465]: Setup publisher on cmd_vel_rc100 [geometry_msgs/Twist]
[INFO] [1587329085.259543]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1587329085.354142]: Setup publisher on joint_states [sensor_msgs/JointState]
[INFO] [1587329085.383768]: Setup publisher on battery_state [sensor_msgs/BatteryState]
[INFO] [1587329085.491250]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]
[INFO] [1587329088.216690]: Setup publisher on /tf [tf/tfMessage]
[INFO] [1587329088.273765]: Note: subscribe buffer size is 1024 bytes
[INFO] [1587329088.275897]: Setup subscriber on cmd_vel [geometry_msgs/Twist]
[INFO] [1587329088.325226]: Setup subscriber on sound [turtlebot3_msgs/Sound]
[INFO] [1587329088.382814]: Setup subscriber on motor_power [std_msgs/Bool]
[INFO] [1587329088.434202]: Setup subscriber on reset [std_msgs/Empty]
[INFO] [1587329088.487397]: Setup TF on Odometry [odom]
[INFO] [1587329088.491224]: Setup TF on IMU [imu_link]
[INFO] [1587329088.494877]: Setup TF on MagneticField [mag_link]
[INFO] [1587329088.498781]: Setup TF on JointState [base_link]
[INFO] [1587329088.506985]: --------------------------
[INFO] [1587329088.510729]: Connected to OpenCR board!
[INFO] [1587329088.514477]: This core(v1.2.3) is compatible with TB3 Burger
[INFO] [1587329088.518133]: --------------------------
[INFO] [1587329088.521845]: Start Calibration of Gyro
[INFO] [1587329091.048767]: Calibration End
```

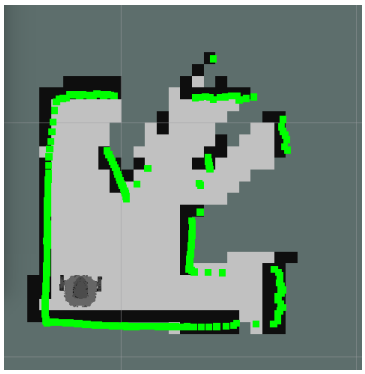
Si desea dejar de ejecutar el bringup debe oprimir CTRL + c en todas las terminales que tenga en ejecución.

5. PRÁCTICAS DE ROBÓTICA BÁSICA

Se hizo dos prácticas de robótica básica con el robot TurtleBot3, estas se enfocan en el mapeo de un entorno de manera teleoperada y autónoma. En estas prácticas se incluye el sensor laser LDS-01 y en la práctica de mapeo de manera autónoma se explica cómo crear un programa en ROS desde cero con el fin de que el usuario tenga algunas bases para el manejo del robot y de la creación de programas en ROS.

Comenzando con las prácticas, se tiene un entorno de trabajo real que para el caso de este manual es el que se muestra en la imagen; si se desea hacer en otro entorno se puede hacer sin ningún problema y el proceso es el mismo.

Antes de iniciar debe seguir todos los pasos del apartado 4.7 del manual. Ahora en otra venta de terminal debe ejecutar el comando roslaunch turtlebot3_slam turtlebot_slam.launch slam_methods:=gmapping, este comando ejecuta un paquete para el proceso de mapeo, al final del comando se define el método de mapeo que se utiliza ya que para este robot existen varios métodos de mapeo que pueden ser descargados, este en particular se descarga junto a los paquetes instalados en el apartado 4.2 del manual. Al ejecutar este comando se abre el simulador de entornos Rviz y muestra el entorno en el que está el robot ya que el sensor está funcionando.



Las líneas verdes son las muestras que está tomando el robot y las líneas negras son el mapa que se está creando al mismo tiempo.

Ahora se debe mover el robot por medio del entorno en el cual está para poder crear todo el mapa, para esto se tienen tres opciones, dos opciones en manejo del robot en forma teleoperada y el manejo de forma autónoma.

5.1 Mapeo de forma teleoperada por medio del teclado

En una nueva ventana de terminal debe ejecutar los siguientes comandos:

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

El primer comando exporta el modelo del robot ya que este paquete funciona para cualquiera de los modelos de robot. El segundo comando ejecuta el paquete teleop_key que es el que contiene toda la programación que hace que el robot funcione por medio del teclado. Al ejecutar el comando aparece la siguiente información:

```
... logging to /home/camila/.ros/log/f1f59128-63ff-11ea-bcd5-5cac4c8a3372/roslaunch-camila-Aspire-4741-2806.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://192.168.43.12:40673/

SUMMARY
========

PARAMETERS
 * /model: burger
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)

ROS_MASTER_URI=http://192.168.43.12:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [2825]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit
```

NOTA: Se debe tratar de que el simulador de entornos Rviz y la ejecución de este programa se puedan ver al mismo tiempo en la PC para que se vea el proceso de movimiento en el entorno real y en el simulador y que también se vea el proceso de mapeo.

Por medio de las teclas w, a, d, x, puede hacer que el robot se mueva en diferentes direcciones y con la letra s que se detenga por completo; si se oprime una tecla una sola vez el robot empieza moviéndose muy

lentamente sin embargo si oprime una sola tecla varias veces, este aumentara su velocidad. Además se puede dar cuenta de la velocidad del robot en el terminal.

```
currently:      linear vel 0.01  angular vel 0.0
currently:      linear vel 0.02  angular vel 0.0
currently:      linear vel 0.03  angular vel 0.0
currently:      linear vel 0.04  angular vel 0.0
currently:      linear vel 0.05  angular vel 0.0
currently:      linear vel 0.06  angular vel 0.0
currently:      linear vel 0.06  angular vel -0.1
currently:      linear vel 0.06  angular vel -0.2
currently:      linear vel 0.06  angular vel -0.3
currently:      linear vel 0.06  angular vel -0.4
currently:      linear vel 0.06  angular vel -0.5
currently:      linear vel 0.06  angular vel -0.6
currently:      linear vel 0.06  angular vel -0.7
```

Si se desea salir de la ejecución de programa, debe oprimir CTRL + c.

También debe oprimir CTRL + c en el terminal en el que está ejecutando el paquete de mapeo para que termine el proceso completamente el proceso.

5.2 Mapeo de forma teleoperada por medio del simulador de entornos Rviz

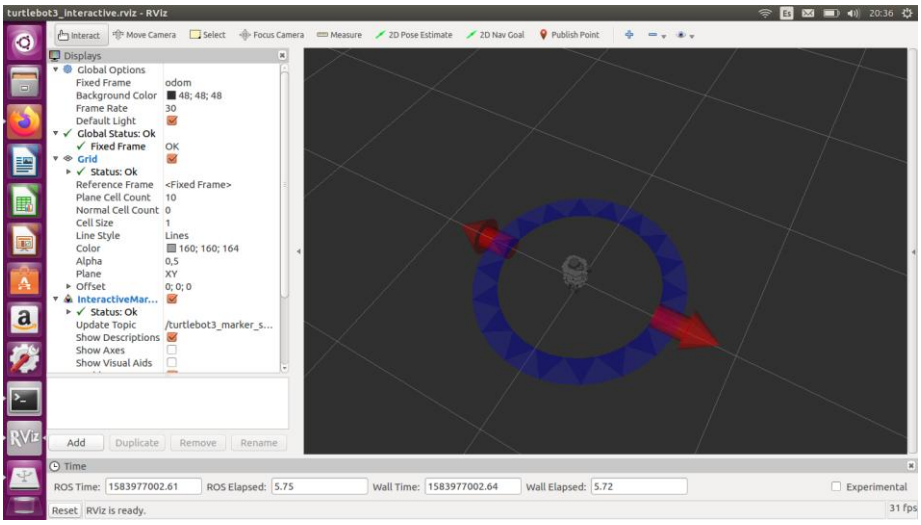
Debe ejecutar en un nuevo terminal el siguiente comando:

```
$ roslaunch turtlebot3_example interactive_markers.launch
```

Este comando ejecuta los marcadores interactivos que son necesarios en el simulador de entornos Rviz para el movimiento del robot. Ahora en otro terminal ejecute el simulador con el siguiente comando:

```
$ rosrun rviz rviz -d `rospack find
turtlebot3_example`/rviz/turtlebot3_interactive.rviz
```

Al ejecutar el comando se abre una ventana aparte de la ventana que contiene el proceso de mapeo, en esta ventana moviendo las flechas rojas con el mouse puede mover el robot tanto en el simulador como en tiempo real. Además también puede controlar la velocidad del robot haciendo uso de las flechas rojas y puede dar giro al robot con el círculo azul.



Si se desea salir de la ejecución de programa, debe oprimir CTRL + c.

También debe oprimir CTRL + c en el terminal en el que está ejecutando el paquete de mapeo para que termine el proceso completamente el proceso.

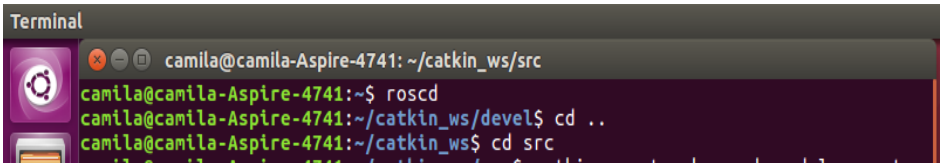
5.3 Mapeo de forma autónoma

Ahora se hace la explicación del proceso de mapeo de manera autónoma, es decir se hace un código fuente en el cual se le dan instrucciones al robot para que sean realizadas de manera automática. Para este proceso se creó un programa desde cero en ROS, el usuario puede utilizar este programa o hacer otro programa desde ceros siguiendo cada paso.

Se crea un programa que detecte los obstáculos que el robot tenga en su parte frontal, cabe aclarar que si el obstáculo es más bajo que el robot TurtleBot3, no lo va a detectar ya que esto se hace por medio del sensor laser LDS-01 y este se encuentra en la parte superior del robot.

5.3.1 Crear un paquete

Se debe crear el paquete el cual contiene todo lo necesario para la ejecución del programa. Primero se debe entrar a la dirección en la cual se debe crear el paquete ejecutando los siguientes comandos:



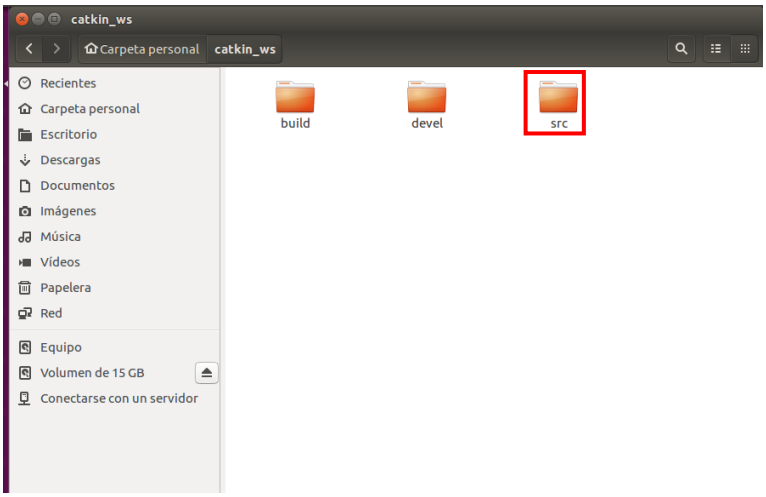
Para crear el paquete se ejecuta el siguiente comando:

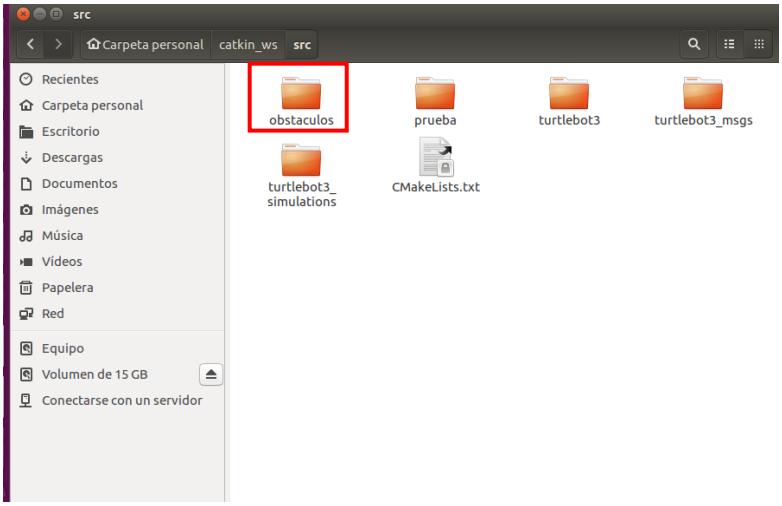
catkin_create_pkg nombre_del_paquete rospy

nombre_del_paquete lo puede cambiar por el nombre que se desee, para el caso de este manual se le asigna el nombre de obstaculos. Además en el comando se está indicando que este paquete depende de la librería rospy que es necesaria para que se pueda programar en lenguaje Python (si se desea programar por medio del lenguaje C++ la palabra rospy se cambia por roscpp). Para el caso del programa que se construye en este manual el comando queda de la siguiente manera:

catkin_create_pkg obstaculos rospy

Se oprime enter y queda creado el paquete. Para verificar que si fue creado el paquete, debe ir al icono de archivos de Ubuntu que se encuentra en la parte izquierda de la pantalla, si lo abre encuentra la carpeta catkin_ws dentro de esta carpeta encuentra la carpeta src y dentro de esta encuentra que ya esta creado el paquete obstaculos.





Si se abre la carpeta obstaculos encuentra que dentro de esta se creó también una carpeta llamada src y dos archivos, uno llamado CMakeList.txt y otro llamado package.xml.

El archivo CMakeList.txt es el que se le da al sistema de compilación CMake para compilar y el archivo package.xml contiene información general del paquete.

5.3.2 Crear el archivo .launch (lanzador)

Luego de verificar que todo se ha creado correctamente, en la carpeta obstaculos se crea la carpeta launch.

Dentro de esta carpeta se crea un archivo el cual se llamara nombre_del_paquete.launch. Para el caso de este manual este archivo queda con el nombre obstaculos.launch (se le llama igual que al paquete por comodidad, pero se le puede dar el nombre que desee).

Es importante que este archivo lleve en su nombre .launch para cualquier programa que se realice, ya sea el de este manual o cualquier otro que desee realizar por su cuenta, debido a que este archivo ejecuta el nodo (archivo que contiene la programación) y en caso de que se realicen múltiples nodos este archivo se encarga de ejecutarlos todos al tiempo.

En este archivo se coloca lo siguiente:

```
<launch>
    <node pkg="obstaculos" type="obstaculos.py" name="obstaculos" output="screen">
    </node>
</launch>
```

Es un código basado en XML en el cual se utilizan etiquetas launch y dentro de estas se agrega el nodo que se desea ejecutar (si son múltiples nodos se agregan de la misma manera que se agrega el nodo en este manual dentro de las mismas etiquetas launch).

Pkg es el nombre del paquete, type es el nombre del nodo (para el caso de este manual se crea más adelante y obtaculos.py es el nombre del archivo), name es el nombre que se le da al nodo para ejecutarlo (se ve más adelante en este manual) y output se coloca en caso de que una salida tenga que ser publicada en el terminal en el que se ejecuta el programa (para el caso de este manual en el terminal se puede observar la distancia a la cual está el obstáculo).

5.3.3 Crear el nodo

El nodo es el archivo que contiene la programación que se desea ejecutar.

Dentro de la carpeta obstáculos se encuentra la carpeta src, dentro de esta carpeta se crea el archivo con el nombre nombre_del_paquete.py se coloca el nombre del paquete por comodidad pero si se desea colocar otro nombre se pude hacer, se coloca el .py para saber que el nodo se programa en lenguaje Python.

En el archivo se coloca el siguiente código:

obstaculos.py (~/catkin_ws/src/obstaculos/src) - gedit

Abrir Guardar

```
#!/usr/bin/env python

import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

rospy.init_node('obstaculos')

public = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
mov = Twist()
mov.linear.x = 0.22
mov.angular.z = 0
public.publish(mov)

def printLaser(dato):
        dis=dato.ranges[0]
        print dato.ranges[0]
        if dis < 0.3:
                mov.linear.x = 0
                mov.angular.z = 1.5708
                public.publish(mov)
        else:
                mov.linear.x = 0.22
                mov.angular.z = 0
                public.publish(mov)

rospy.Subscriber('/scan', LaserScan, printLaser)
rospy.spin()
```

Python Anchura de la pestaña: 8 Ln 3, Col 13 INS

5.3.4 Explicación del código fuente (nodo)

Es importante que para cualquier código que se desee hacer en ROS y en lenguaje de programación Python la primera línea sea #!/usr/bin/env python ya que #! Indica al sistema encargado de ejecutar los programas que el archivo es script y ejecuta el archivo utilizando el interprete (/usr/bin/env python), para el caso de este manual el interprete es Python.

En las siguientes tres líneas del código primero se importa la librería de Python para ROS; segundo, para poder procesar la información que proporciona el sensor laser se importa el tipo de mensaje LaserScan de la librería sensor_msgs.msg y tercero para poder procesar la información de movimiento del robot se importa el tipo de mensaje Twist de la librería geometry_msgs.msg.

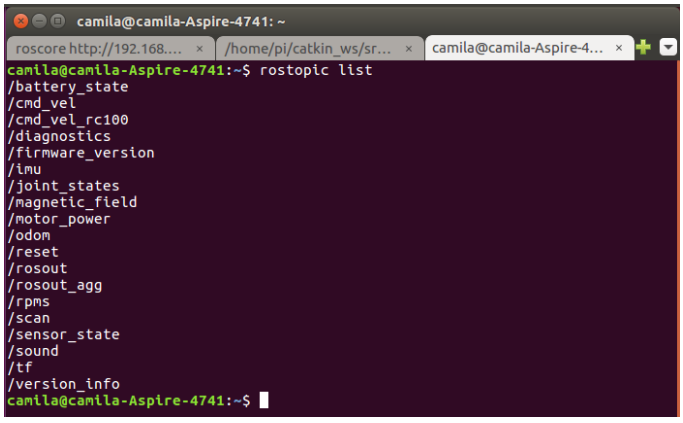
Se debe tener en cuenta que el mensaje Twist está compuesto por tres componentes lineales y tres componentes angulares (x, y, z para cada caso). Estas componentes son de tipo float64 y las unidades de las componentes lineales son metros segundos y de las unidades angulares son radianes sobre segundos.

El nodo en cualquier código se debe inicializar. En la siguiente línea se inicializa dándole un nombre, el cual también ya se había colocado al

crear el archivo obstaculos.launch, el nombre que se había dado en este archivo es obstaculos, así que en el código para inicializar el nodo se coloca el mismo nombre.

Para las siguientes líneas se debe tener claro el concepto de tópico, publicador y subscriptor. Los tópicos en ROS son aquellos mensajes que están disponibles para ser enviados al robot o para ser escuchados desde el robot, el publicador se encarga de enviar los mensajes al robot y el subscriptor se encarga de escuchar los mensajes que llegan desde el robot.

Para esta línea de código primero se debe saber que tópicos están disponibles para usar, para ello en una terminal se debe colocar el comando rostopic list que muestra la siguiente información:



Como se ve en la imagen, en esta lista están disponibles el tópico /cmd/vel que es el mensaje de la velocidad del robot y el tópico /scan que es el mensaje del sensor laser. En este manual solo se tiene en cuenta estos dos tópicos ya que son los que se utilizan en el código fuente.

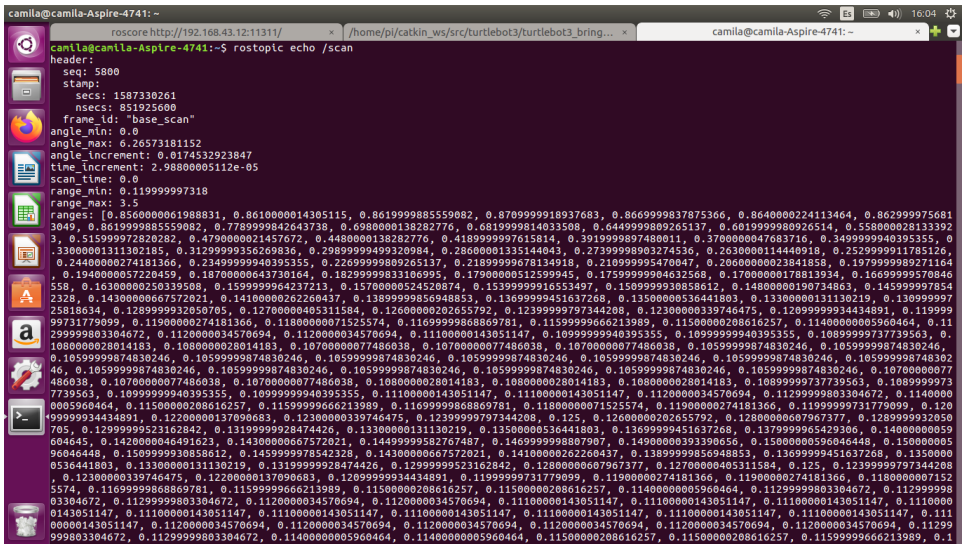
En esta línea se está creando un publicador con tres definiciones, la primera es el mensaje que se envía (tópico), la segunda es el tipo de mensaje que en este caso es tipo twist y la tercera (queue_size) es el tamaño de la cola de mensajes salientes que para este caso es de uno además en esta línea este publicador se iguala a una variable para que pueda ser llamado en cualquier momento.

En las líneas encerradas en el recuadro amarillo se le da valores a la componente lineal en X y a la componente angular en Z del mensaje tipo twist. El mensaje tipo twist trabaja con estas componentes,

sabiendo que X es hacia adelante, Y es hacia la izquierda y Z es el giro. En este caso solo se usa la componente X lineal y Z angular ya que el robot inicia moviéndose hacia adelante y cuando encuentra un obstáculo da un giro determinado y sigue moviéndose hacia adelante. En la línea final del recuadro amarillo se llama al publicador para que envié el mensaje tipo twist con los valores dados a las componentes, las componentes Y, Z lineal y X, Y angular se envían en cero ya que no se les hace ningún cambio.

Aunque las líneas encerradas en el recuadro verde están al final del código, es necesario explicar primero estas líneas para poder entender lo siguiente. En la primera línea se está creando un subscriptor con tres definiciones, la primera es el mensaje que se recibe (tópico), la segunda es el tipo de mensaje que en este caso es de tipo LaserScan y la tercera es el nombre del subscriptor. La segunda línea hace que el código sea repetitivo hasta que se oprima la combinación de teclas CTRL + c.

Antes de continuar, se debe saber que datos entrega el sensor laser, para saberlo se ejecuta en un terminal rostopic list echo /scan una vez ejecutado el comando, inmediatamente se debe oprimir la combinación de teclas CTRL + c ya que este comando muestra todos los datos que está tomando el sensor en el momento en que se ejecute; luego hay que ir a los primeros datos que muestra el comando y puede observar la siguiente información:



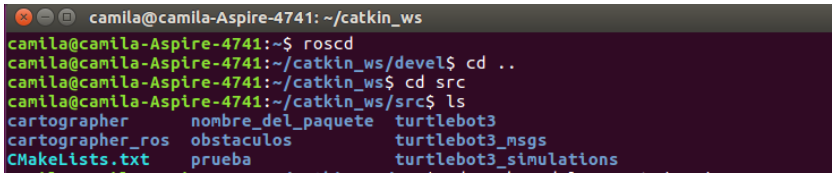
Para este caso, el dato de interés es el ranges, se puede observar que el comando muestra en el ranges varias muestras en las que el sensor laser muestra inf si no detecta ningún objeto a su alrededor y muestra la distancia a la que está el objeto si detecta alguno a su alrededor; estos también depende de la distancia que tenga el sensor laser para la detección de objetos. Se debe tener en cuenta que este sensor gira 360° y comienza a contar cada dato desde la parte frontal del robot, es decir la parte frontal del robot seria la muestra 0 y cuanto el sensor a girado 360° cuenta la muestra numero 360 (por cada grado toma una muestra).

En las líneas de código del recuadro azul se define una función con el nombre del subscriptor ya que en esta función se va a definir que va a hacer el código con los mensajes que recibe el subscriptor, entre paréntesis esta la variable dato ya que los mensajes que lleguen al subscriptor se guardaran en esta variable. En la siguiente línea de la variable dato se substrae el dato ranges en la posición cero y este dato se guarda en la variable dis. Se hace con la posición cero ya que se quiere que el robot esquive obstáculos que tiene en su parte frontal.

La variable dis se usa para hacer un if que es importante en el código, si la variable dis que esta guardando los datos que toma el sensor en la posición cero es menor a 0.3 el robot para y da un giro de 1.5708 rad y se envían estos dos datos por medio del publicador pero si la variable dis es mayor a 0.3 el robot tiene su velocidad inicial y no tiene ningún ángulo de giro, es decir sigue su recorrido y estos datos también se envían por medio del publicador.

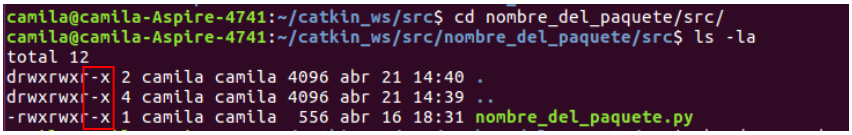
5.3.5 Dar permisos y compilar

Antes de ejecutar el código se debe dar los permisos necesarios para el funcionamiento del programa además de compilarlo. Para dar los permiso primero se debe acceder a el paquete src de la PC y ver los paquetes nuevos que tiene el sistema.



Para el caso de este manual dentro de la información de la imagen anterior debe salir el nombre obstaculos que es el nombre del paquete que se creó.

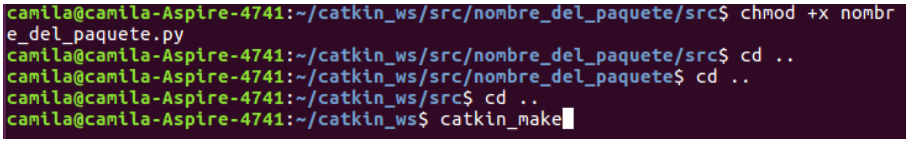
Luego se accede al paquete al que se quiere dar los permisos con el comando cd nombre_del_paquete/src/, para el caso de este manual el comando queda como cd obstaculos/src/ y por medio del comando ls -la se muestran los permisos que aun no están dados para este paquete



Para este caso en el terminal se muestra una información similar a la que se puede observar en la imagen pero en la parte que se puede observar el nombre nombre_del_paquete.py sale el nombre obstaculos.py

En el recuadro rojo se puede observar el permiso que no se le ha otorgado al paquete; –x significa que no se le ha otorgado el permiso de ejecución; para otorgar este permiso se ejecuta el comando chmod +x nombre_del_paquete.py, para el caso de este manual el comando queda chmod +x obstaculos.py.

Con el permiso otorgado ya se puede compilar el paquete, para esto se ejecuta en este mismo terminal los siguientes comandos:



El comando catkin_make compila todos los paquetes que se encuentran en la carpeta catkin_ws así que si sale algún error de algún otro paquete en particular pero no sale ningún error con respecto al paquete que se acaba de crear puede ejecutar el paquete creado sin ningún problema.

5.3.6 Ejecutar el programa

Para ejecutar cualquier programa en ROS se utiliza el comando roslaunch (en otro terminal) seguido del nombre del paquete y del nombre del archivo .launch que se creo

roslaunch nombre_del_paquete nombre_del_paquete.launch

Para ejecutar el programa en este caso es de la siguiente manera

roslaunch obstaculos obstaculos.launch

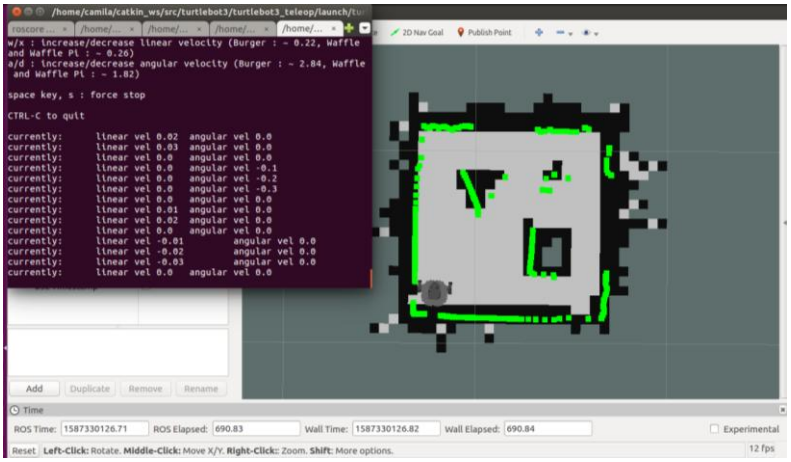
Después de ejecutar el comando el robot empieza a moverse y en el terminal se comienza a ver el dato del sensor laser en la posición cero, es decir en la parte frontal del robot. A la vez se va creando el mapa en el simulador de entornos Rviz.

Si se desea salir de la ejecución de programa, debe oprimir CTRL + c.

También debe oprimir CTRL + c en el terminal en el que está ejecutando el paquete de mapeo para que termine el proceso completamente el proceso.

5.4 Resultado

Como resultado de las prácticas de mapeo se obtiene un mapa de la siguiente forma:



El tipo de mapeo más eficiente es el tipo teleoperado debido a que se puede recorres todo el espacio sin dejar ningún espacio vacío, mientras que el mapeo de tipo autónomo puede que deje espacios en vacio.

Como se puede observar el mapa no queda parecido al entorno real, esto se debe primero a cuestiones de espacio y tamaño, el espacio de mapeo es muy pequeño y segundo se debe al tamaño de las figuras; como se puede observar el cilindro es la figura menos parecida en el mapa a el entorno real esto se debe a que el mapa toma el cilindro como una figura a la que no puede dibujar su contorno debido a su tamaño pequeño y a que el mapa se dibuja por medio de cuadros. En cambio el cuadrado es la figura más parecida al entorno real ya que le favorece el tamaño y también que el mapa se dibuje por medio de cuadrados. También se puede ver que uno de los lados del triangulo, en

el mapa si esta dibujado perfectamente ya que este lado esta recto al espacio por el que circula el robot para hacer el mapa.

BIBLIOGRAFÍA

[1] Y. Pyo, H. Cho, J. Ryuwoon, and T. Lim, *Robot Programming From The Basic Concept To Practical Programming and Robot Application*. 2017.

[2] “es - ROS Wiki.” [Online]. Available: http://wiki.ros.org/es. [Accessed: 30-Jun-2020].

[3] “Questions - ROS Answers: Open Source Q&A Forum.” [Online]. Available: https://answers.ros.org/questions/. [Accessed: 30-Jun-2020].

[4] “TurtleBot3.” [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. [Accessed: 30-Jun-2020].

[5] “Permisos básicos en GNU/Linux con chmod.” [Online]. Available: https://blog.desdelinux.net/permisos-basicos-en-gnulinux-con-chmod/. [Accessed: 30-Jun-2020].

[6] “The Construct – Learn and Develop for Robots using ROS.” [Online]. Available: https://www.theconstructsim.com/. [Accessed: 30-Jun-2020].

[7] “(12) [ROS Tutorials Español] ROS Basics: Publish on a Topic | Ep.1 - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=EYpxVdJVy3c&list=PLK0b4e05LnzbnuhkdBW13lezuiE-1vyzS. [Accessed: 30-Jun-2020].