



Escuela de Invierno de Robótica (EIR 2021)

**Introducción a Ambientes de Simulación y
Herramientas de Software para
Drones Autónomos**

Enero 2021

Guía de Instalación del Sistema Operativo
Ubuntu y el Framework ROS

Impartido por:

Dr. José Martínez Carranza
MC. Leticia Oyuki Rojas Pérez

Contacto:

carranza@inaoep.mx
oyukirojas@inaoep.mx

Instituto Nacional de Astrofísica, Óptica y Electrónica
INAOE
Luis Enrique Erro 1
Santa María Tonantzintla
72840, Puebla, México.



Índice

1. Introducción	3
2. Instalación de Ubuntu	3
2.1. Requisitos mínimos de hardware para instalación de Ubuntu	3
2.2. Pasos para la creación del USB Bootable con el instalador de Ubuntu	3
2.3. Configuración previa a la instalación	5
2.4. Instalación	5
3. Instrucciones para ubuntu 16.04 Lts	10
3.1. Instalación de ROS Kinetic Kame	10
3.2. Instalación de Paquete de simulación de Parrot Ar drone 2.0	11
3.3. Instrucciones de instalación de TensorFlow y Keras para equipos sin GPU	12
3.4. Instrucciones de instalación de Cuda 9.0, CuDNN 7.3, TensorFlow y Keras para equipos con GPU	13
4. Instrucciones para ubuntu 18.04 Lts	20
4.1. Instalación de ROS Melodic Morenia	20
4.2. Instalación de Paquete de simulación de Parrot Ar drone 2.0	21
4.3. Instrucciones de instalación de TensorFlow y Keras para equipos sin GPU	23
4.4. Instrucciones de instalación de Cuda 11.2, CuDNN 8.0.5, TensorFlow y Keras para equipos con GPU	24
5. Paquetes de ROS	29
5.1. Ejecución de Simulador	29
6. Referencias	31

1. Introducción

Este documento es una guía para la instalación del sistema operativo Ubuntu, ROS, la instalación del simulador del vehículo Parrot Ar Drone 2.0, la compilación y ejecución de paquetes de ROS, así como la instalación de TensorFlow y Keras. En este documento se incluye la creación de una USB bootable con el instalador del sistema operativo, la configuración de las particiones para instalar correctamente Ubuntu y la actualización de los drivers del equipo. Así mismo se indica la configuración el entorno de ROS, la creación de un espacio de trabajo, la adquisición de paquetes desde un repositorio en GitHub y posteriormente la instalación de las librerías necesarias para la compilación de paquete de simulación. Los paquetes de ROS se han probado únicamente en las versiones ROS Kinetic Kame y Melodic Morenia.

Lea todo el instructivo antes de iniciar la instalación.

2. Instalación de Ubuntu

Antes de comenzar con las instalación, se recomienda realizar un respaldo de la toda la información importante que pueda contener el equipo antes de instalar el sistema operativo Ubuntu, ya que es posible que se presente algún fallo en el sistema durante o después de la instalación.

2.1. Requisitos mínimos de hardware para instalación de Ubuntu

- Procesador Dual Core 2 GHz (o superior).
- 2GB de Memoria Ram.
- 25GB de espacio libre en disco duro.
- Memoria USB de 4GB (Bootable).
- Acceso a Internet.

2.2. Pasos para la creación del USB Bootable con el instalador de Ubuntu

El primer paso es descargar e instalar el programa *Linux Live USB creator*, éste programa permitirá crear un USB para la instalación del sistema operativo *LINUX*. Una vez instalado Linux live se procede a descargar la imagen ISO de Ubuntu 16.04/18.04 Lts . Recuerde descargar la versión correspondiente a procesador, 32 bits o bien 64 bits.

Al ejecutar el programa Linux Live USB Creator se desplegará una ventana como lo muestra la Figura 1(a), en ella se indica el dispositivo de almacenamiento, Figura 1(c) y la imagen ISO del sistema Operativo,

Figura 2. Para finalizar la creación del Bootable se formatea la memoria USB y se da clic en el icono del rayo, Figura 3, la creación del bootable tarda aproximadamente 30 minutos.



Figura 1: Programa Linux Live USB Creator y Selección de dispositivo de almacenamiento.

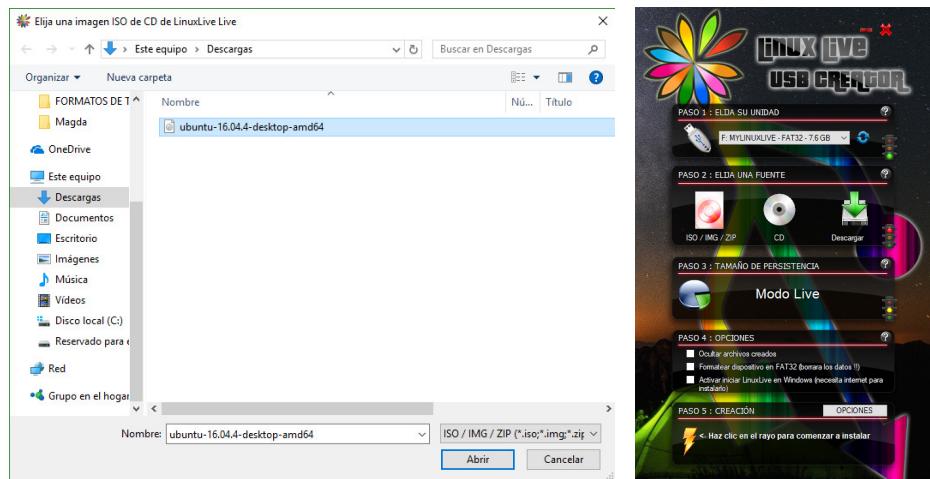


Figura 2: Selección de imagen ISO.



Figura 3: Creación del Bootable.

2.3. Configuración previa a la instalación

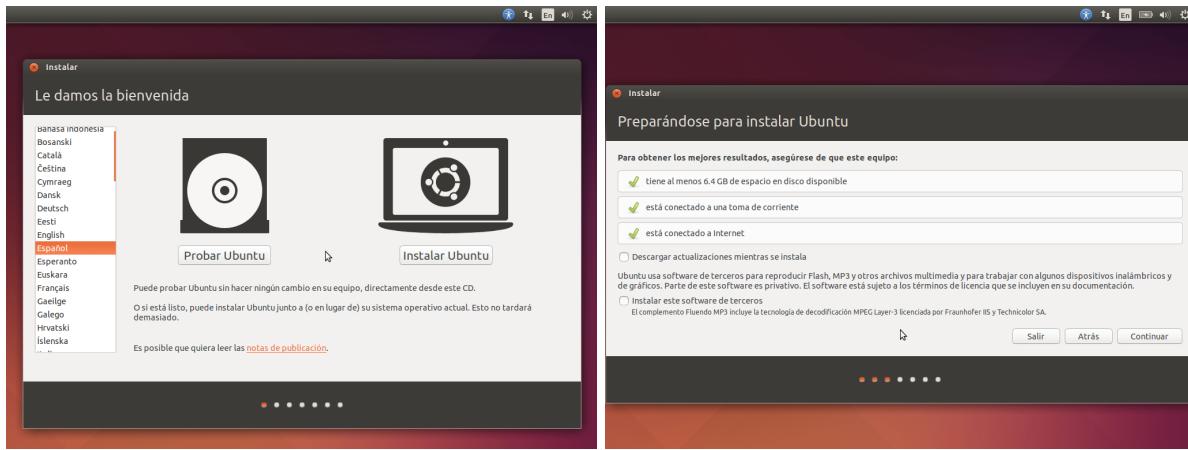
Es necesario acceder a la BIOS del Equipo y realizar los siguientes cambios en el menú Boot:

- Cambiar el Boot Mode de UEFI a LEGACY.
- Deshabilitar el Secure Boot.
- Modificar la prioridad de Boot para iniciar desde una USB.
- Guardar cambios y reiniciar.

Nota: si estos cambios no se realizan el equipo no permitirá la instalación de otro sistema operativo

2.4. Instalación

Una vez haya cargado el instalador, lo primero es elegir el idioma a utilizar. Ubuntu ofrece la posibilidad de probar el sistema sin instalación sin afectar al sistema operativo instalado, ejemplo Windows, Figura 4(a). Si se escoge esa opción se accederá al escritorio de Ubuntu, desde donde es posible continuar la instalación. El asistente de instalación verifica que el equipo tenga al menos 6.4GB de espacio libre en el disco duro, que este conectado a una toma de corriente y la conexión a Internet, Figura 4(b). Sino dispone de una conexión por cable a Internet, el sistema le ofrecerá configurar una red WiFi.



(a) Inicio del asistente de instalación

(b) Verificación previa a la instalación

Figura 4: Asistente de instalación de Ubuntu 16.04 Lts.

El siguiente paso es seleccionar el tipo de instalación, Figura 5(a). Lo recomendable es separar el sistema de los dato, creando tres particiones, Figura 5(b): una para el sistema operativo, otra para datos y una tercera como memoria virtual, esto como medida de seguridad ante un eventual fallo del sistema, la re-instalación no provocaría la pérdida de archivos y/o configuraciones personales.

La Figura 5(c), muestra un ejemplo del espacio disponible en disco duro, el tamaño se muestra en MB (megabytes). Las Figuras 5(d), 5(e) y 5(f), muestran la configuración para cada partición.

- La primera partición se designa al sistema operativo, y éste no necesita de gran espacio, pueden ser 20000 MB, en el campo “Utilizar como” se selecciona *Sistema de ficheros ext4 transaccional*. Finalmente se asigna el punto de montaje, que en este caso sería el símbolo “/”, equivalente a la partición raíz del sistema.
- La segunda partición a crear es el área de intercambio o memoria swap, en el campo “Utilizar como” se selecciona ”área de intercambio”. Se recomienda fijar al área de intercambio el mismo tamaño que la memoria RAM, por ejemplo, en equipos con más de 4 GB de memoria RAM un término medio es fijar 4240 MB. La partición de swap es imprescindible para activar la hibernación del equipo.
- La tercera partición corresponde a los datos personales del usuario, en esta partición es posible utilizar todo el espacio en disco restante. En el campo “Utilizar como” se selecciona *Sistema de ficheros ext4 transaccional* y el ”punto de montaje”se selecciona */home*.

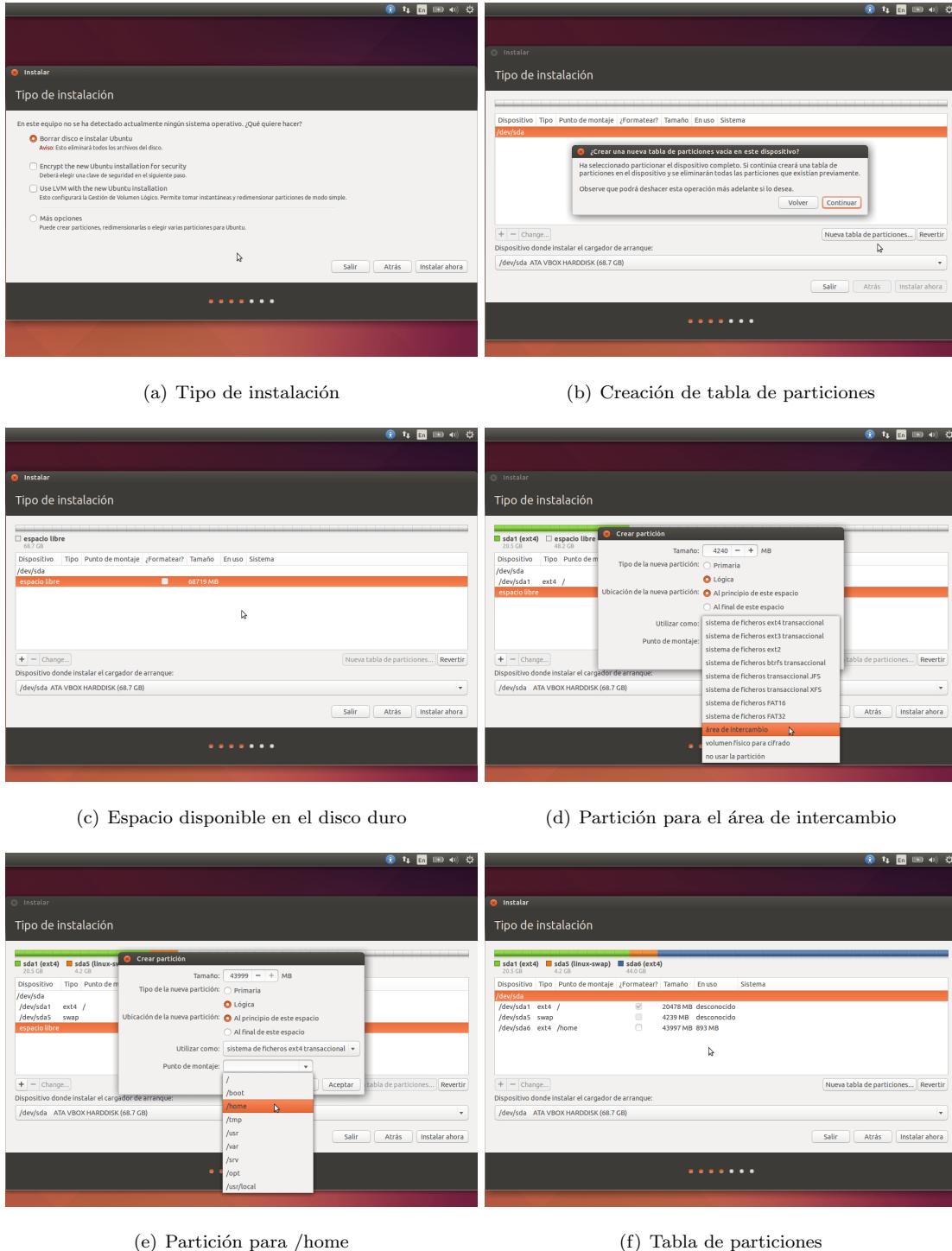


Figura 5: Tipo de instalación y creación de particiones.

Hasta este punto todos los cambios son reversibles, una vez particionado el disco comenzará la instalación del sistema y no habrá vuelta atrás. Se continua la instalación indicando la ubicación geográfica, Figura 6(a), la distribución del teclado, Figura 6(b) y finaliza solicitando nombre del propietario del equipo, nombre del equipo y contraseña. Una vez realizados estos pasos el asistente comienza la instalación de sistema operativo Ubuntu 16.04 Lts, el proceso tarda aproximadamente 2 horas.

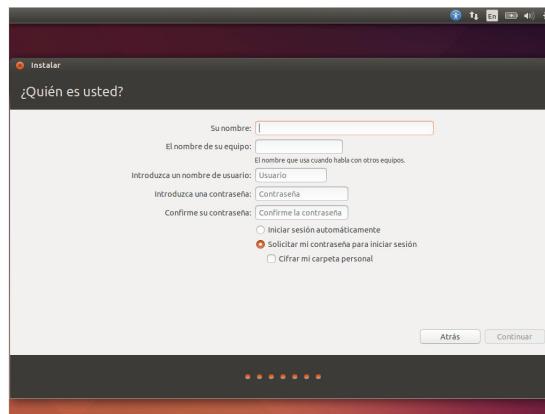
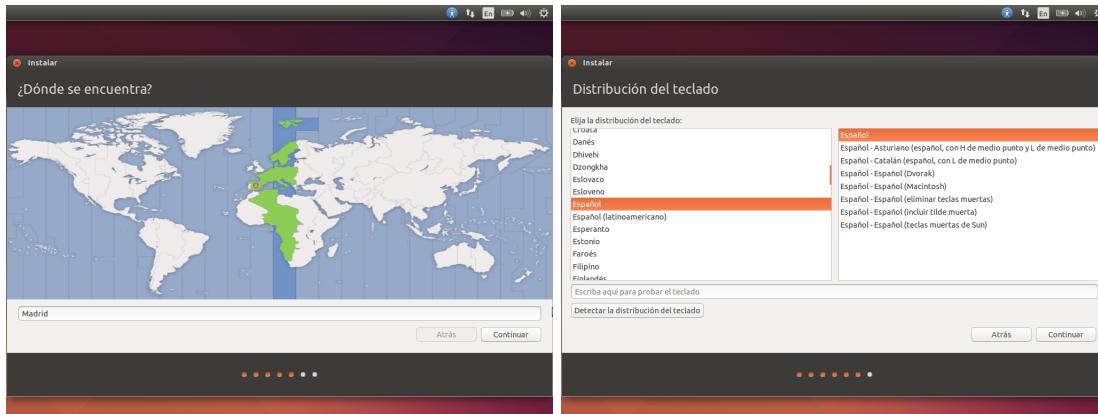


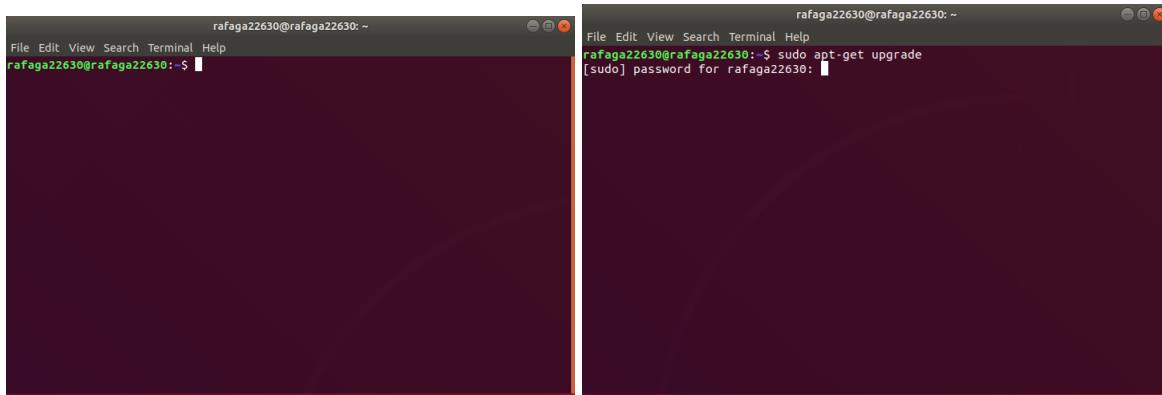
Figura 6: Configuración final del asistente de instalación.

El último paso es la actualización de los drivers del equipo, de la siguiente manera se abre una terminal presionando las teclas *Ctrl + Alt + t*, como se muestra en la Figura 7(a) y se escribe en ella el siguiente comando:

```
sudo apt-get upgrade
```

La contraseña que se solicita es la misma con la que accede a su cuenta al iniciar Ubuntu. Al termino del

proceso de ejecución se debe reiniciar el equipo.



(a) Terminal

(b) Actuaización de drivers

Figura 7: Configuración final del equipo.

3. Instrucciones para ubuntu 16.04 Lts

3.1. Instalación de ROS Kinetic Kame

A continuación se enlistan los pasos para la instalación de ROS.

1. Configuración para adquirir paquetes de ROS "source.list"

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu\\$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Key

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyserver.net:80 --recv-key421C365BD9FF1F717815A3895523BAEEB01FA116
```

3. Actualización de paquetes Debian

```
sudo apt-get update
```

4. Descarga e instalación de ROS

```
sudo apt-get install ros-kinetic-desktop-full
```

5. Inicialización de rosdep

```
sudo rosdep init  
rosdep update
```

6. Configuración del entorno

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

7. Instalación de dependencias para la construcción de paquetes

```
sudo apt-get install python-rosinstall python-rosinstall-generator  
python-wstool build-essential
```

La instalación de ROS incluye las librerías OpenCV, usada para procesamiento de imágenes, Qt, usada para creación de Widgets, entre otras. Así como herramientas de visualización como Rviz o Rqt y de simulación como Gazebo. No es necesario instalar librerías adicionales.

3.2. Instalación de Paquete de simulación de Parrot Ar drone 2.0

La instalación del paquete para simular el vehículo Parrot Ar Drone 2.0, requiere la creación del espacio de trabajo. Este se crea con los siguientes comandos en la terminal:

1. Creación de un espacio de trabajo

```
mkdir -p ~/EIR2021/src  
cd ~/EIR2021  
catkin_make
```

2. Actualización del archivo de configuración

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

3. Verificación de la variable de entorno

```
echo $ROS_PACKAGE_PATH
```

Una vez ejecutado este comando la respuesta en la terminal será la siguiente

```
/home/SuUsuario/EIR2021/src:/opt/ros/kinetic/share
```

4. Path del espacio de trabajo

Escriba en una terminal

```
gedit ~/.bashrc
```

Al final del documento abajo de la línea siguiente

```
source /opt/ros/kinetic/setup.bash
```

Se escribe el path del espacio de trabajo

```
source ~/EIR2021/devel/setup.bash
```

5. Instalación de ardrone_autonomy

Abra una terminal con los comandos *Ctr + Alt + t*, y acceda a la *src* que se encuentra dentro del espacio de trabajo como lo indica la Figura 8. Se procede a la instalación con el siguiente comando en la terminal.

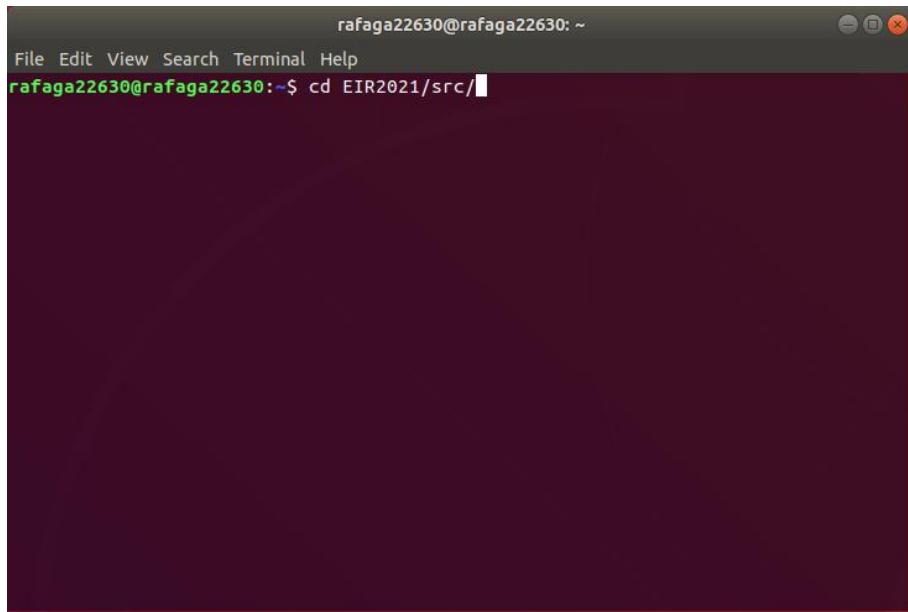


Figura 8: Terminal Ubuntu.

```
sudo apt-get install ros-kinetic-ardrone-autonomy
```

6. Descarga de repositorio tum_simulator

Acceda nuevamente a la carpeta *src* con el comando *cd /src* y clonar el repositorio

```
sudo apt-get install ros-kinetic-hector-*  
git clone https://github.com/angelsantamaria/tum_simulator.git
```

7. Compilación, en la terminal se escriben los siguientes comandos

```
cd ..  
catkin_make
```

3.3. Instrucciones de instalación de TensorFlow y Keras para equipos sin GPU

TensorFlow y Keras son dos librerías Open Source que nos permiten utilizar herramientas para Deep Learning de forma sencilla. Las instrucciones de instalación sirven para ubuntu 16 o superior.

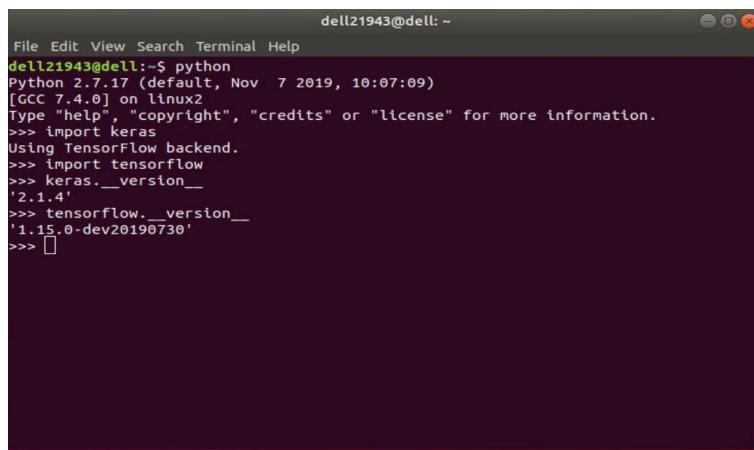
1. Abra una terminal, presionando Crtl + Alt + T y ejecute los siguientes comandos:

```
sudo apt install python-pip  
sudo pip install keras==2.1.4  
sudo pip install tensorflow==1.4.0
```

Nota: En caso de error ejecutar lo siguiente:

```
sudo pip install --ignore-installed tensorflow==1.4.0
```

2. Comprobación de instalación exitosa, en la misma terminal ejecute los siguientes comandos, como lo muestra la Figura 16



```
dell21943@dell: ~  
File Edit View Search Terminal Help  
dell21943@dell:~$ python  
Python 2.7.17 (default, Nov  7 2019, 10:07:09)  
[GCC 7.4.0] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import keras  
Using TensorFlow backend.  
>>> import tensorflow  
>>> keras.__version__  
'2.1.4'  
>>> tensorflow.__version__  
'1.15.0-dev20190730'  
>>> █
```

Figura 9: Comprobación de instalación exitosa de tensorflow y Keras.

Comandos a ejecutar:

```
python  
import keras  
import tensorflow  
keras.__version__  
tensorflow.__version__
```

3.4. Instrucciones de instalación de Cuda 9.0, CuDNN 7.3, TensorFlow y Keras para equipos con GPU

1. Descargue el archivo libs.tar.gz del enlace siguiente <https://drive.google.com/file/d/1JochFRi-LTRfu78Ftk9IoGvoLj7nkkhW/view?usp=sharing> y descomprima libs.tar.gz en la carpeta personal (Home), debe estar de la siguiente manera:

- Carpeta Personal

- EIR2021 (workspace)
- libs
 - CUDA_9
 - CUDNN
 - libcudnn7_7.3.0.29-1+cuda9.0_amd64.deb
 - libcudnn7-dev_7.3.0.29-1+cuda9.0_amd64.deb
 - libcudnn7-doc_7.3.0.29-1+cuda9.0_amd64.deb

2. Instalación de driver NVIDIA. La duración de la instalación es aproximadamente 10 minutos, abra una terminal y ejecute el siguiente comando

```
sudo apt-get install nvidia-384 nvidia-modprobe
```

3. Reinicie su máquina, abra una terminal y ejecute lo siguiente:

```
nvidia-smi
```

El resultado de ser como el que muestra la Figura 17.

```
rafaga22038@rafaga22038:~$ nvidia-smi
Fri Feb 28 14:22:01 2020
+-----+
| NVIDIA-SMI 384.130                 Driver Version: 384.130 |
+-----+
| GPU  Name     Persistence-M| Bus-Id     Disp.A  Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|=====+=====+=====+=====+=====+=====+=====+=====+=====|
| 0  GeForce GTX 970M     Off  00000000:01:00.0 Off           N/A |
| N/A   60C   P0    24W / N/A |      567MiB /  3018MiB |     13%     Default |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name        Usage          |
| =====+=====+=====+=====
| 0    1012  G    /usr/lib/xorg/Xorg      282MiB |
| 0    2070  G    compiz                146MiB |
| 0    3787  G    unity-control-center      2MiB |
| 0    3853  G    ...AAAAAAAAAAAAgAAAAAAA --shared-files  134MiB |
+-----+
rafaga22038@rafaga22038:~$
```

Figura 10: Comprobación de instalación exitosa del driver NVIDIA.

4. Instalación de CUDA 9.0 via runfile

```
wget https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/  
cuda_9.0.176_384.81_linux-run
```

5. Dar permisos al archivo y extracción del contenido.

```
chmod +x cuda_9.0.176_384.81_linux-run  
. ./cuda_9.0.176_384.81_linux-run --extract=~/libs/CUDA_9
```

6. verifique que dentro de la carpeta /libs/CUDA_9 se encuentren los siguientes archivos:

- cuda_9.0.176_384.81_linux-run
- cuda-linux.9.0.176-22781540.run (el instalador de CUDA 9.0)
- cuda-samples.9.0.176-22781540-linux.run (las muestras de CUDA 9.0)
- NVIDIA-Linux-x86_64-384.81.run

7. En la misma terminal ejecute los siguientes comandos para instalar el CUDA toolkit 9.0:

```
cd ~/libs/CUDA_9  
sudo ./cuda-linux.9.0.176-22781540.run
```

Pulse la tecla “d” para avanzar y acepte los valores predeterminados. Después ejecute lo siguiente:

```
sudo ./cuda-samples.9.0.176-22781540-linux.run
```

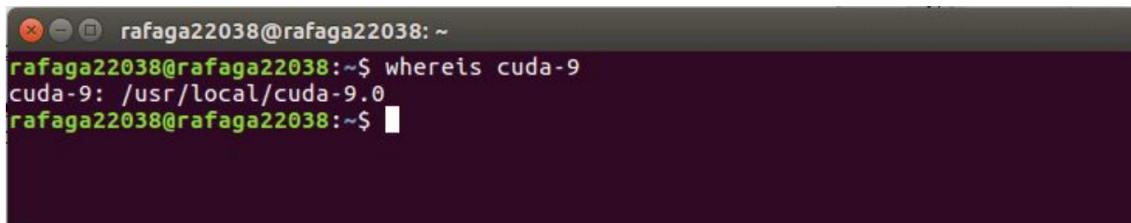
8. Añada el path de las librerías de CUDA 9.0 en el archivo bashrc, de la siguiente manera, la misma terminal ejecute:

```
gedit ~/.bashrc
```

una vez abierto el archivo bashrc copie y pegue las siguientes líneas:

```
export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64:$LD_LIBRARY_PATH  
export PATH=/usr/local/cuda-9.0/bin:$PATH
```

Nota: verifique que su path de cuda-9.0 coincida con export LD_LIBRARY_PATH y export PATH. con el siguiente comando:



```
rafaga22038@rafaga22038:~$ whereis cuda-9
cuda-9: /usr/local/cuda-9.0
rafaga22038@rafaga22038:~$
```

Figura 11: Comprobación de path de CUDA-9.0.

```
whereis cuda-9
```

La Figura 18, muestra el resultado que debe obtener.

9. Instalación de cuDNN 7.0. Localice los paquetes de instalación ubicados en la carpeta CUDNN, los archivos son los siguientes:

- libcudnn7_7.3.0.29-1+cuda9.0_amd64.deb
- libcudnn7-dev_7.3.0.29-1+cuda9.0_amd64.deb
- libcudnn7-doc_7.3.0.29-1+cuda9.0_amd64.deb

10. En la misma terminal ejecute los siguientes comandos:

```
cd ~/libs/CUDNN
sudo dpkg -i libcudnn7_7.3.0.29-1+cuda9.0_amd64.deb
sudo dpkg -i libcudnn7-dev_7.3.0.29-1+cuda9.0_amd64.deb
sudo dpkg -i libcudnn7-doc_7.3.0.29-1+cuda9.0_amd64.deb
```

11. Una vez concluido el paso anterior, procedemos a verificar si la instalación se ha realizado de manera exitosa, en la misma terminal ejecutamos lo siguiente:

```
cp -r /usr/src/cudnn_samples_v7/ ~/libs/CUDNN/
cd ~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN
```

12. Compile los ejemplos con el siguiente comando (esto puede tardar unos minutos)

```
make clean && make
```

La Figura 19, muestra el resultado que debe obtener.

```

rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN$ make clean && make
rm -rf *o
rm -rf mnistCUDNN
/usr/local/cuda/bin/nvcc -ccbin g++ -I/usr/local/cuda/include -IFreeImage/include -m64 -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=sm_50 -gencode arch=compute_53,code=sm_53 -gencode arch=compute_53,code=compute_53 -o fp16_dev.o -c fp16_dev.cu
g++ -I/usr/local/cuda/include -IFreeImage/include -o fp16_emu.o -c fp16_emu.cpp
g++ -I/usr/local/cuda/include -IFreeImage/include -o mnistCUDNN.o -c mnistCUDNN.cpp
/usr/local/cuda/bin/nvcc -ccbin g++ -m64 -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=sm_50 -gencode arch=compute_53,code=sm_53 -gencode arch=compute_53,code=compute_53 -o mnistCUDNN fp16_dev.o fp16_emu.o mnistCUDNN.o -I/usr/local/cuda/include -IFreeImage/include -LFreeImage/lib/linux/x86_64 -LFreeImage/lib/linux -lcudart -lcublas -lcudnn -lfreeimage -lstdc++ -lm
rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN$ 

```

Figura 12: Compilación de los ejemplos de CUDNN.

13. Por último se ejecuta uno de los ejemplos 3 con el siguiente comando:

```
./mnistCUDNN
```

Si la instalación fue exitosa, debe de aparecer en la terminal la leyenda Test passed!, como lo muestran las Figura 20.

14. Instalación de Tensorflow. Abra una terminal, presionando Crtl + Alt + T y ejecute los siguientes comandos:

```
sudo apt install python-pip
sudo python -m pip install numpy
sudo python -m pip install tensorflow-gpu==1.12.0
```

Nota: En caso de error ejecutar lo siguiente:

```
sudo python -m pip install --ignore-installed tensorflow-gpu==1.12.0
```

15. Instalación de Keras

```
sudo python -m pip install keras==2.2.4
sudo python -m pip install imutils
```

```

rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN
rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN$ ./mnistCUDNN
cudnnGetVersion() : 7300 , CUDNN VERSION from cudnn.h : 7300 (7.3.0)
Host compiler version : GCC 5.4.0
There are 1 CUDA capable devices on your machine :
device 0 : sms 10 Capabilities 5.2, SmClock 1038.0 Mhz, MemSize (Mb) 3018, MemClock 2
505.0 Mhz, Ecc=0, boardGroupID=0
Using device 0

Testing single precision
Loading image data/one_28x28.pgm
Performing forward propagation ...
Testing cudnnGetConvolutionForwardAlgorithm ...
Fastest algorithm is Algo 1
Testing cudnnFindConvolutionForwardAlgorithm ...
~~~~~ CUDNN_STATUS_SUCCESS for Algo 0: 0.035840 time requiring 0 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 1: 0.054144 time requiring 3464 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 2: 0.068288 time requiring 57600 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 4: 0.129888 time requiring 207360 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 7: 0.181568 time requiring 2057744 memory
Resulting weights from Softmax:
0.000000 0.9999399 0.0000000 0.0000000 0.0000561 0.0000000 0.0000012 0.0000017 0.0000
010 0.000000
Loading image data/three_28x28.pgm
Performing forward propagation ...

```



```

rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN
Performing forward propagation ...
Resulting weights from Softmax:
0.0000000 0.0000008 0.0000000 0.0000002 0.0000000 0.9999820 0.0000154 0.0000000 0.0000
012 0.0000006

Result of classification: 1 3 5

Test passed!

Testing half precision (math in single precision)
Loading image data/one_28x28.pgm
Performing forward propagation ...
Testing cudnnGetConvolutionForwardAlgorithm ...
Fastest algorithm is Algo 1
Testing cudnnFindConvolutionForwardAlgorithm ...
~~~~~ CUDNN_STATUS_SUCCESS for Algo 0: 0.037536 time requiring 0 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 1: 0.042784 time requiring 3464 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 2: 0.069440 time requiring 28800 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 4: 0.138272 time requiring 207360 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 7: 0.175680 time requiring 2057744 memory
Resulting weights from Softmax:
0.0000001 0.0000000 0.0000000 0.0000563 0.0000001 0.0000012 0.0000017 0.0000
010 0.0000001
Loading image data/three_28x28.pgm

```



```

rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN
Testing cudnnFindConvolutionForwardAlgorithm ...
~~~~~ CUDNN_STATUS_SUCCESS for Algo 0: 0.037536 time requiring 0 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 1: 0.042784 time requiring 3464 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 2: 0.069440 time requiring 28800 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 4: 0.138272 time requiring 207360 memory
~~~~~ CUDNN_STATUS_SUCCESS for Algo 7: 0.175680 time requiring 2057744 memory
Resulting weights from Softmax:
0.0000001 0.0000000 0.0000000 0.0000563 0.0000001 0.0000012 0.0000017 0.0000
010 0.0000001
Loading image data/three_28x28.pgm
Performing forward propagation ...
Resulting weights from Softmax:
0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000714 0.0000000 0.0000000 0.0000
000 0.0000000
Loading image data/five_28x28.pgm
Performing forward propagation ...
Resulting weights from Softmax:
0.0000000 0.0000008 0.0000000 0.0000002 0.0000000 1.0000000 0.0000154 0.0000000 0.0000
012 0.0000006

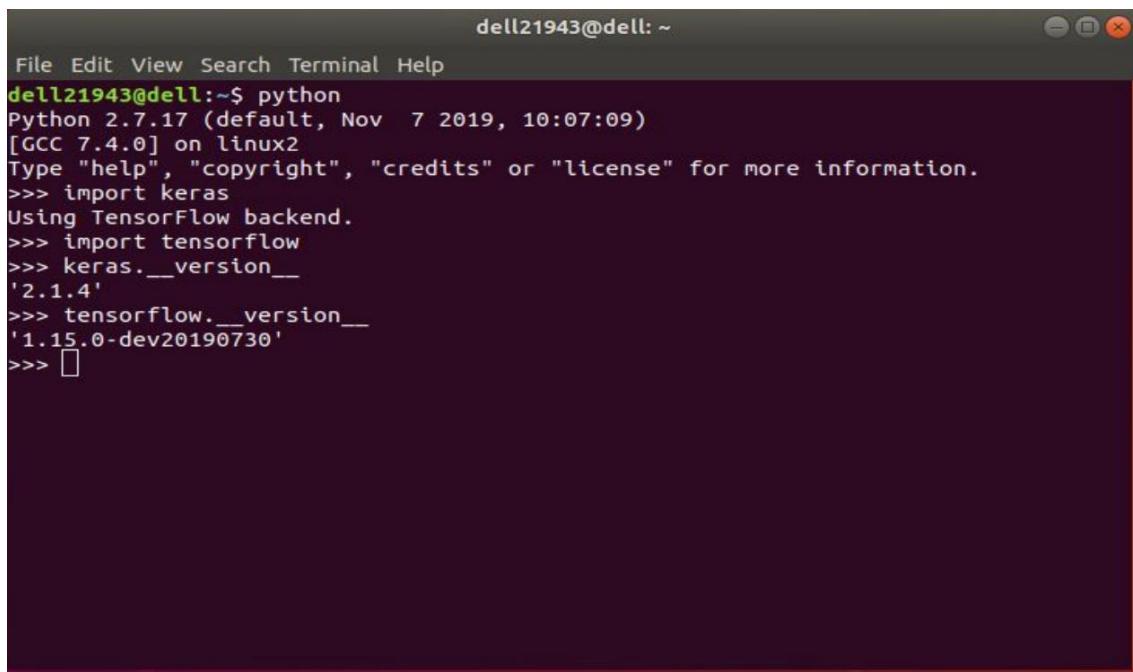
Result of classification: 1 3 5

Test passed!
rafaga22038@rafaga22038:~/libs/CUDNN/cudnn_samples_v7/mnistCUDNN$ 

```

Figura 13: Ejecución del ejemplo mnistCUDNN.

16. Comprobación de instalación exitosa, en la misma terminal ejecute lo siguientes comandos, como lo muestra la Figura 21

A screenshot of a terminal window titled "dell21943@dell: ~". The window shows the following Python session:

```
File Edit View Search Terminal Help
dell21943@dell:~$ python
Python 2.7.17 (default, Nov  7 2019, 10:07:09)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>> import tensorflow
>>> keras.__version__
'2.1.4'
>>> tensorflow.__version__
'1.15.0-dev20190730'
>>> 
```

The terminal has a dark background and light-colored text.

Figura 14: Comprobación de instalación exitosa de tensorflow y Keras.

Comandos a ejecutar:

```
python
import keras
import tensorflow
keras.__version__
tensorflow.__version__
```

4. Instrucciones para ubuntu 18.04 Lts

4.1. Instalación de ROS Melodic Morenia

A continuación se enlistan los pasos para la instalación de ROS.

1. Configuración para adquirir paquetes de ROS "source.list"

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Key

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

3. Actualización de paquetes Debian

```
sudo apt-get update
```

4. Descarga e instalación de ROS

```
sudo apt install ros-melodic-desktop-full
```

5. Inicialización de rosdep

```
sudo rosdep init  
rosdep update
```

6. Configuración del entorno

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc  
source ~/.bashrc  
source /opt/ros/melodic/setup.bash
```

7. Instalación de dependencias para la construcción de paquetes

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

La instalación de ROS incluye las librerías OpenCV, usada para procesamiento de imágenes, Qt, usada para creación de Widgets, entre otras. Así como herramientas de visualización como Rviz o Rqt y de simulación como Gazebo. No es necesario instalar librerías adicionales.

4.2. Instalación de Paquete de simulación de Parrot Ar drone 2.0

La instalación del paquete para simular el vehículo Parrot Ar Drone 2.0, requiere la creación del espacio de trabajo. Este se crea con los siguientes comandos en la terminal:

1. Creación de un espacio de trabajo

```
mkdir -p ~/EIR2021/src  
cd ~/EIR2021  
catkin_make
```

2. Actualización del archivo de configuración

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

3. Verificación de la variable de entorno

```
echo $ROS_PACKAGE_PATH
```

Una vez ejecutado este comando la respuesta en la terminal será la siguiente

```
/home/SuUsuario/EIR2021/src:/opt/ros/melodic/share
```

4. Path del espacio de trabajo

Escriba en una terminal

```
gedit ~/.bashrc
```

Al final del documento abajo de la línea siguiente

```
source /opt/ros/melodic/setup.bash
```

Se escribe el path del espacio de trabajo

```
source ~/EIR2021/devel/setup.bash
```

5. Descarga del paquete ardrone_autonomy

Abra una terminal con las teclas *Ctr + Alt + t*, y acceda a la *src* que se encuentra dentro del espacio de trabajo como lo indica la Figura 15.

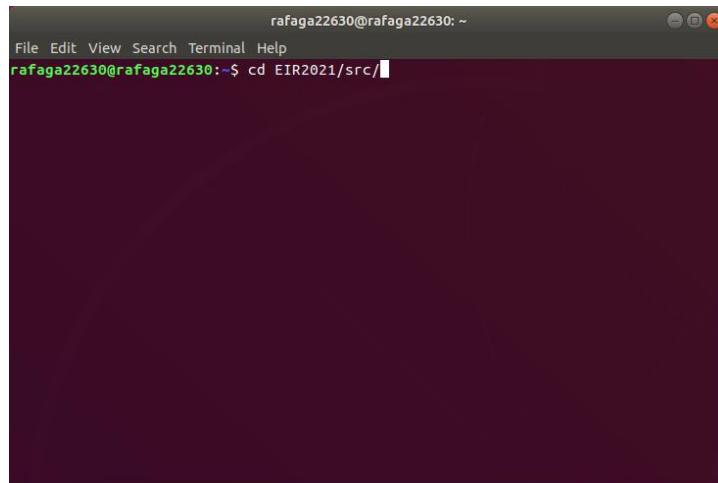


Figura 15: Terminal Ubuntu.

Se procede a la instalación con los siguientes comandos:

```
cd ~/EIR2021/src  
sudo apt install qt4-default  
sudo apt-get install ros-melodic-hector-*  
git clone https://github.com/dsapandora/ardrone_autonomy.git -b melodic-devel  
rosdep install --from-paths src --ignore-src -r -y  
catkin_make  
sudo su  
source devel/setup.bash  
cd ..  
catkin_make -DCMAKE_INSTALL_PREFIX=/opt/ros/melodic install
```

Una vez instalado el paquete ardrone_autonomy cierre la terminal.

6. Descarga de repositorio tum_simulator

Descargue y descomprima la carpeta tum_simulator que se encuentra en el siguiente enlace https://drive.google.com/file/d/1YXniR04-c9J016z1w3qx5K_RDfIAJ03k/view?usp=sharing dentro de la siguiente ruta: home/EIR2021/src

abran una terminal y accedan al workspace y compilen el paquete

7. Compilación, abra una terminal y ejecute los siguientes comandos:

```
cd ~/EIR2021/src  
catkin_make
```

4.3. Instrucciones de instalación de TensorFlow y Keras para equipos sin GPU

TensorFlow y Keras son dos librerías Open Source que nos permiten utilizar herramientas para Deep Learning de forma sencilla. Las instrucciones de instalación sirven para ubuntu 16 o superior.

1. Abra una terminal, presionando Crtl + Alt + T y ejecute los siguientes comandos:

```
sudo apt install python-pip  
sudo pip install keras==2.1.4  
sudo pip install tensorflow==1.4.0
```

Nota: En caso de error ejecutar lo siguiente:

```
sudo pip install --ignore-installed tensorflow==1.4.0
```

2. Comprobación de instalación exitosa, en la misma terminal ejecute lo siguientes comandos, como lo muestra la Figura 16

```
dell21943@dell: ~  
File Edit View Search Terminal Help  
dell21943@dell:~$ python  
Python 2.7.17 (default, Nov  7 2019, 10:07:09)  
[GCC 7.4.0] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import keras  
Using TensorFlow backend.  
>>> import tensorflow  
>>> keras.__version__  
'2.1.4'  
>>> tensorflow.__version__  
'1.15.0-dev20190730'  
>>> [REDACTED]
```

Figura 16: Comprobación de instalación exitosa de tensorflow y Keras.

Comandos a ejecutar:

```
python  
import keras
```

```
import tensorflow
keras.__version__
tensorflow.__version__
```

4.4. Instrucciones de instalación de Cuda 11.2, CuDNN 8.0.5, TensorFlow y Keras para equipos con GPU

1. Debe crear una carpeta con el nombre libs y dentro de ella 2 carpetas mas, una con el nombre cuda y otra llamada cudnn

- Carpeta Personal

- EIR2021 (workspace)
- libs
 - cuda
 - cudnn

2. Instalación de driver NVIDIA. La duración de la instalación es aproximadamente 10 minutos, abra una terminal y ejecute el siguiente comando

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt update
sudo apt install -y nvidia-driver-460
```

3. Reinicie su máquina, abra una terminal y ejecute lo siguiente:

```
nvidia-smi
```

El resultado de ser como el que muestra la Figura 17.

```

rafaga22630@rafaga22630:~$ nvidia-smi
Tue Jan 19 19:08:21 2021
+-----+
| NVIDIA-SMI 460.32.03     Driver Version: 460.32.03    CUDA Version: 11.2 |
| Persistence-M| Bus-Id     Disp.A  Volatile Uncorr. ECC | | | |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|          |          |          |          |      MIG M. |
+-----+
| 0  GeForce GTX 1070   Off  00000000:01:00.0 On           N/A |
| N/A  57C   P0  37W / N/A   573MiB /  8111MiB |   0%     Default |
|          |          |          |          |      N/A |
+-----+
Processes:
+-----+
| GPU  GI  CT      PID   Type  Process name        GPU Memory |
| ID   ID              |          |          | Usage      |
+-----+
| 0   N/A N/A  1312    G   /usr/lib/xorg/Xorg    26MiB |
| 0   N/A N/A  1373    G   /usr/bin/gnome-shell  89MiB |
| 0   N/A N/A  1597    G   /usr/lib/xorg/Xorg    198MiB |
| 0   N/A N/A  1743    G   /usr/bin/gnome-shell  43MiB |
| 0   N/A N/A  2104    G   ...AAAAAAA== ...shared-files  210MiB |
+-----+
rafaga22630@rafaga22630:~$ 

```

Figura 17: Comprobación de instalación exitosa del driver NVIDIA.

4. Instalación de CUDA 11.2 via runfile

```

cd ~/libs/cuda
wget https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/
cuda_11.2.0_460.27.04_linux.run

```

5. Dar permisos al archivo e instalación de CUDA toolkit 11.2

```

chmod +x cuda_11.2.0_460.27.04_linux.run
sudo ./cuda_11.2.0_460.27.04_linux.run

```

6. Añada el path de las librerías de CUDA 11.2 en el archivo bashrc, de la siguiente manera, la misma terminal ejecute:

```
gedit ~/.bashrc
```

una vez abierto el archivo bashrc copie y pegue las siguientes líneas al final del documento:

```

export PATH=/usr/local/cuda/bin${PATH:+:$PATH}
export LD_LIBRARY_PATH=/usr/local/cuda/lib64${LD_LIBRARY_PATH:+:$LD_LIBRARY_PATH}

```

Nota: verifique que su path de cuda coincida con export LD_LIBRARY_PATH y export PATH. con el siguiente comando: whereis cuda

La Figura 18, muestra el resultado que debe obtener.

```
rafaga22630@rafaga22630: ~
File Edit View Search Terminal Help
rafaga22630@rafaga22630:~$ whereis cuda
cuda: /usr/local/cuda
rafaga22630@rafaga22630:~$
```

Figura 18: Comprobación de path de CUDA

7. Instalación de cuDNN 8.0.5. Descargue los siguientes archivos de <https://developer.nvidia.com/rdp/cudnn-download> y coloquelos dentro de la carpeta /libs/cudnn.

- libcudnn8_8.0.5.39-1+cuda11.1_amd64.deb
- libcudnn8-dev_8.0.5.39-1+cuda11.1_amd64.deb
- libcudnn8-samples_8.0.5.39-1+cuda11.1_amd64.deb

8. En la misma terminal ejecute los siguientes comandos:

```
cd cudnn/
sudo dpkg -i libcudnn8_8.0.5.39-1+cuda11.1_amd64.deb
sudo dpkg -i libcudnn8-dev_8.0.5.39-1+cuda11.1_amd64.deb
sudo dpkg -i libcudnn8-samples_8.0.5.39-1+cuda11.1_amd64.deb
```

9. Una vez concluido el paso anterior, procedemos a verificar si la instalación se ha realizado de manera exitosa, en la misma terminal ejecutamos lo siguiente:

```
cp -r /usr/src/cudnn_samples_v8/ ~/libs/cudnn/
cd cudnn_samples_v8/
cd mnistCUDNN/
```

10. Compile los ejemplos con el siguiente comando (esto puede tardar unos minutos)

```
make clean && make
```

La Figura 19, muestra el resultado que debe obtener.

11. Por último se ejecuta uno de los ejemplos 3 con el siguiente comando:

```
./mnistCUDNN
```

Si la instalación fue exitosa, debe de aparecer en la terminal la leyenda Test passed!, como lo muestran las Figura 20.

```

rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN
File Edit View Search Terminal Help
rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN$ make clean && make
rm -rf *
rm -rf mnistCUDNN
Linking against cublasLt = true
CUDA VERSION: 11020
TARGET ARCH: x86_64
HOST ARCH: x86_64
TARGET OS: linux
SM5: 35 50 53 60 61 62 70 72 75
/usr/local/cuda/bin/nvcc -ccbin g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/include -m64 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=sm_50 -gencode arch=compute_53,code=sm_53 -gencode arch=compute_60,code=sm_60 -gencode arch=compute_61,code=sm_61 -gencode arch=compute_62,code=sm_62 -gencode arch=compute_70,code=sm_70 -gencode arch=compute_72,code=sm_72 -gencode arch=compute_75,code=sm_75 -gencode arch=compute_75,code=sm_75 -o fp16_dev.o -c fp16_dev.cu
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).
g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/include -o fp16_emu.o -c fp16_emu.cpp
g++ -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/include -o mnistCUDNN.o -c mnistCUDNN.cpp
/usr/local/cuda/bin/nvcc -ccbin g++ -m64 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_50,code=sm_50 -gencode arch=compute_53,code=sm_53 -gencode arch=compute_60,code=sm_60 -gencode arch=compute_61,code=sm_61 -gencode arch=compute_62,code=sm_62 -gencode arch=compute_70,code=sm_70 -gencode arch=compute_72,code=sm_72 -gencode arch=compute_75,code=sm_75 -gencode arch=compute_75,code=sm_75 -o mnistCUDNN fp16_dev.o mnistCUDNN.o -I/usr/local/cuda/include -I/usr/local/cuda/targets/ppc64le-linux/include -IFreeImage/include -L/usr/local/cuda/lib64 -L/usr/local/cuda/targets/ppc64le-linux/lib -lcublasLt -lFreeImage/lib/linux/x86_64 -LFreeImage/lib/linux -lcudart -lcublas -lcudnn -lfreeimage -lstdc++ -lm
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).
rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN$ 
```

Figura 19: Compilación de los ejemplos de CUDNN.

```

rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN
File Edit View Search Terminal Help
rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN$ ./mnistCUDNN
Executing: mnistCUDNN
cudnnGetVersion() : 8005 , CUDDN_VERSION from cudnn.h : 8005 (8.0.5)
Host compiler version : GCC 7.5.0

There are 1 CUDA capable devices on your machine :
device 0 : sms 16 Capabilities 6.1, SmClock 1695.0 Mhz, MemSize (Mb) 8111, MemClock 4094.0 Mhz, Ecc=0, boardGroupId=0
Using device 0

Testing single precision
Loading binary file data/conv1.bin
Loading binary file data/conv1.bias.bin
Loading binary file data/conv2.bin
Loading binary file data/conv2.bias.bin
Loading binary file data/ip1.bin
Loading binary file data/ip1.bias.bin
Loading binary file data/ip2.bin
Loading binary file data/ip2.bias.bin
Loading image data/one_28x28.pgm
Performing forward propagation...
Testing cudnnGetConvolutionForwardAlgorithm_v7 ...

rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN
File Edit View Search Terminal Help
Testing cudnnGetConvolutionForwardAlgorithm_v7 ...
^^^^ CUDNN_STATUS_SUCCESS for Algo 1: 1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 0: 1.000000 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 2: 1.000000 time requiring 57600 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 5: 1.000000 time requiring 178432 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 7: 1.000000 time requiring 2057744 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 4: 1.000000 time requiring 184784 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 6: 1.000000 time requiring 0 memory
Algo 3 is not supported for Algo 0: 1.000000 time requiring 0 memory
Testing cudnnFindConvolutionForwardAlgorithm ...
^^^^ CUDNN_STATUS_SUCCESS for Algo 1: 0.042208 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 0: 0.057192 time requiring 0 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 2: 0.123744 time requiring 57600 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 5: 0.286016 time requiring 178432 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 7: 18.398848 time requiring 2057744 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 4: 53.757248 time requiring 184784 memory
^^^^ CUDNN_STATUS_NOT_SUPPORTED for Algo 6: 1.000000 time requiring 0 memory
Algo 3 is not supported for Algo 0: 1.000000 time requiring 0 memory
Testing cudnnGetConvolutionForwardAlgorithm_v7 ...
^^^^ CUDNN_STATUS_SUCCESS for Algo 4: 1.000000 time requiring 2450080 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 7: 1.000000 time requiring 1433120 memory
^^^^ CUDNN_STATUS_SUCCESS for Algo 5: 1.000000 time requiring 4056640 memory

rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN
File Edit View Search Terminal Help
Testing forward propagation ...
Resulting weights from Softmax:
0.0000000 0.0000000 0.0000002 0.0000000 0.9999820 0.0000154 0.0000000 0.0000000
012 0.0000006

Result of classification: 1 3 5
Test passed!

Testing half precision (math in single precision)
Loading binary file data/conv1.bin
Loading binary file data/conv1.bias.bin
Loading binary file data/conv2.bin
Loading binary file data/conv2.bias.bin
Loading binary file data/ip1.bin
Loading binary file data/ip1.bias.bin
Loading binary file data/ip2.bin
Loading binary file data/ip2.bias.bin
Loading image data/one_28x28.pgm
Performing forward propagation...
Testing cudnnGetConvolutionForwardAlgorithm_v7 ...

rafaga22630@rafaga22630:~/libs/cudnn/cudnn_samples_v8/mnistCUDNN$ 
```

Figura 20: Ejecución del ejemplo mnistCUDNN.

12. Instalación de Tensorflow. Abra una terminal, presionando Crtl + Alt + T y ejecute los siguientes comandos:

```
sudo apt install python-pip  
sudo python -m pip install numpy  
sudo python -m pip install tensorflow-gpu==1.14.0
```

13. Instalación de Keras

```
sudo python -m pip install keras==2.2.4  
sudo python -m pip install imutils
```

14. Comprobación de instalación exitosa, en la misma terminal ejecute lo siguientes comandos, como lo muestra la Figura 21

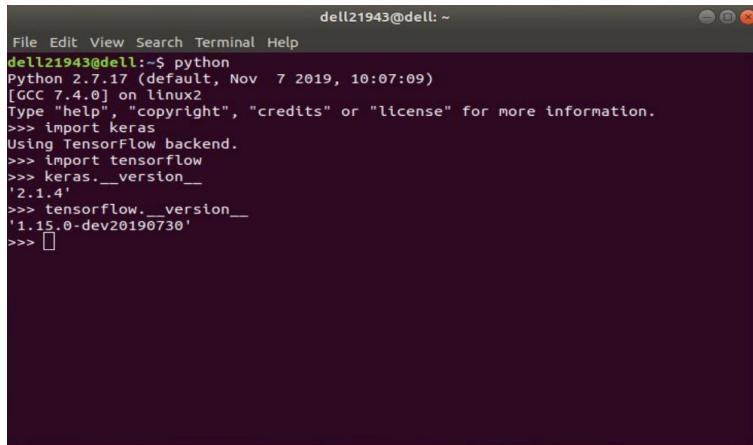
A screenshot of a terminal window titled "dell21943@dell: ~". The window shows a Python session starting with "File Edit View Search Terminal Help" at the top. The command "python" is run, followed by imports for keras and tensorflow. Then, the __version__ attribute is checked for both, displaying '2.1.4' for keras and '1.15.0-dev20190730' for tensorflow.

Figura 21: Comprobación de instalación exitosa de tensorflow y Keras.

Comandos a ejecutar:

```
python  
import keras  
import tensorflow  
keras.__version__  
tensorflow.__version__
```

5. Paquetes de ROS

En nuestro repositorio de GitHub https://github.com/QuetzalCpp/ROS_Packages podrán encontrar los siguientes paquetes, los cuales serán usados en el tutorial.

- Image_viewer. Paquete para visualizar cámara publicada en un nodo de ROS.
- teclado Paquete para realizar control manual del vehículo, funciona para ardrone simulado y bebop version 1 y 2.
- dataset. Paquete para generar conjuntos de datos de entrenamiento.
- seguidor. Paquete para realizar seguimiento autónomo de un objeto usando aprendizaje profundo.

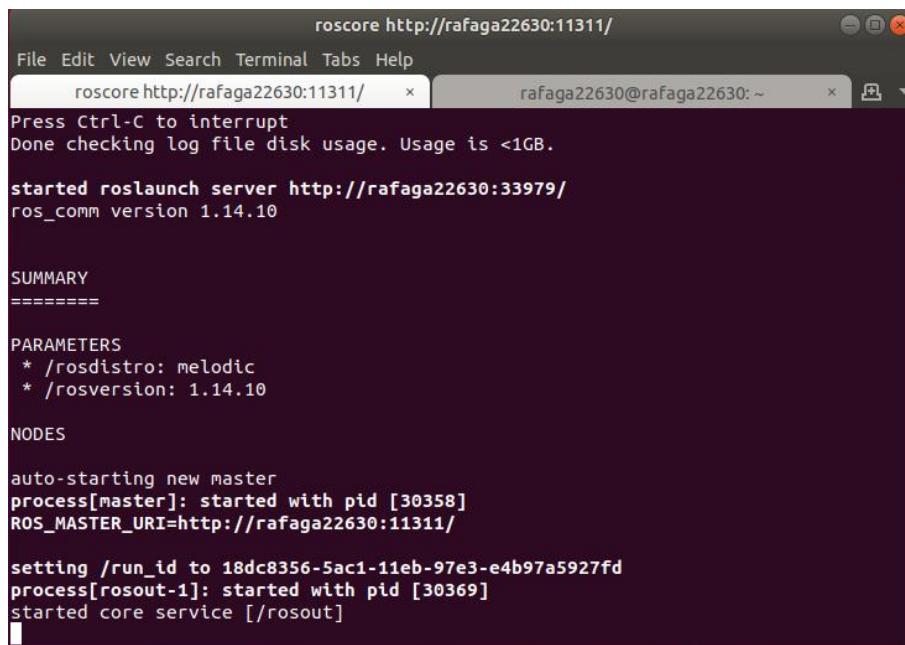
Estos paquetes deben estar dentro de la siguiente ruta: home/EIR2021/src

Posteriormente deben compilar los paquetes para ello deben abrir una terminal y ejecutar los siguientes comandos:

```
cd ~/EIR2021  
catkin_make
```

5.1. Ejecución de Simulador

1. Comando de inicio de la comunicación del sistema ROS *roscore*, ver Figura 22.



```

roscore http://rafaga22630:11311/
File Edit View Search Terminal Tabs Help
roscore http://rafaga22630:11311/ x rafaga22630@rafaga22630: ~ x
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rafaga22630:33979/
ros_comm version 1.14.10

SUMMARY
=====
PARAMETERS
* /rosdistro: melodic
* /rosversion: 1.14.10

NODES

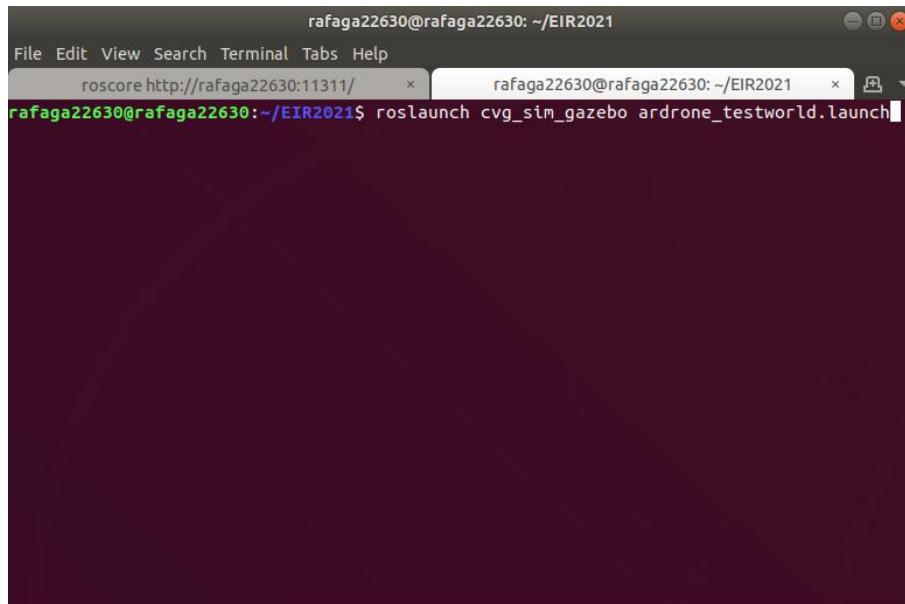
auto-starting new master
process[master]: started with pid [30358]
ROS_MASTER_URI=http://rafaga22630:11311/

setting /run_id to 18dc8356-5ac1-11eb-97e3-e4b97a5927fd
process[rosout-1]: started with pid [30369]
started core service [/rosout]

```

Figura 22: Terminal Ubuntu con el comando *roscore*.

2. Abra una segunda terminal presionando las teclas *Ctrl + Shift + t* y escriba en ella *roslaunch cvg_sim_gazebo ardrone_testworld.launch*, ver Figura 23. Después de ejecutar el comando se desplegará el simulador Gazebo como muestra la Figura 24.



```

rafaga22630@rafaga22630: ~/EIR2021
File Edit View Search Terminal Tabs Help
roscore http://rafaga22630:11311/ x rafaga22630@rafaga22630: ~/EIR2021 x
rafaga22630@rafaga22630:~/EIR2021$ roslaunch cvg_sim_gazebo ardrone_testworld.launch

```

Figura 23: Terminal Ubuntu con comando para ejecución del simulador.

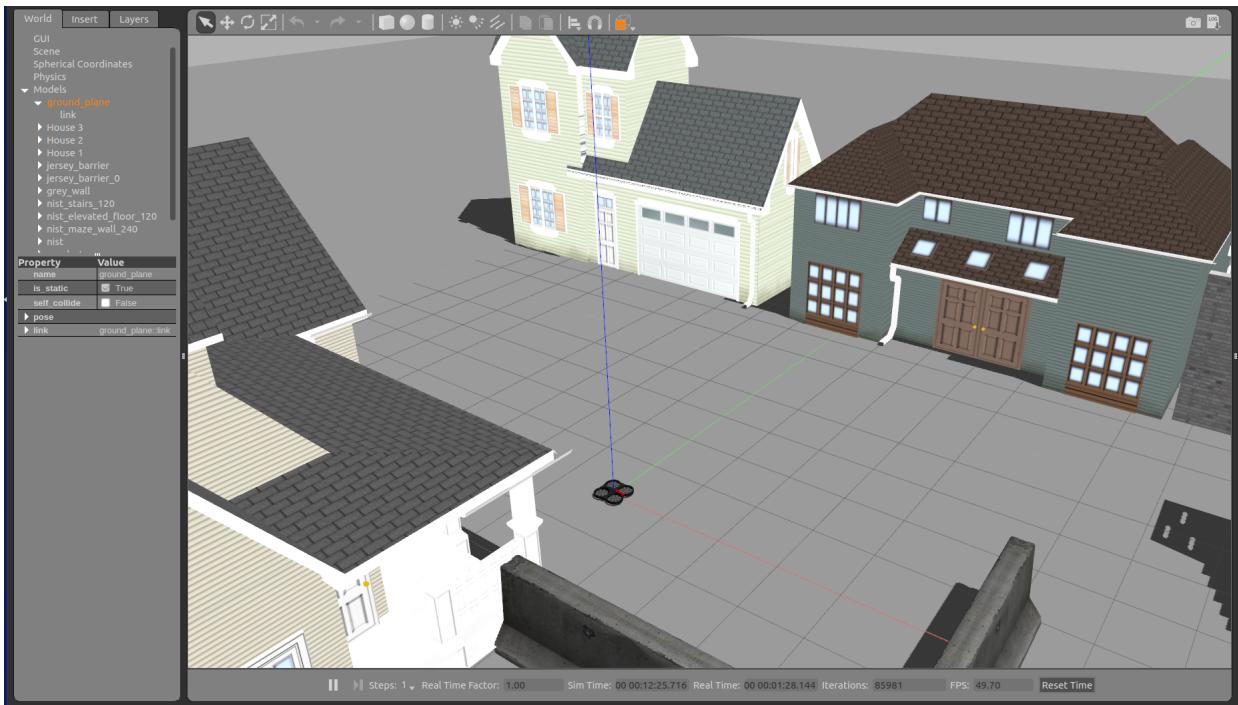


Figura 24: Simulador Gazebo con el modelo Ar drone 2.0.

6. Referencias

1. <https://www.muylinux.com/2014/07/09/guia-instalacion-basica-ubuntu/>
2. <http://releases.ubuntu.com/16.04/>
3. <http://wiki.ros.org/kinetic/Installation/Ubuntu>
4. https://github.com/AutonomyLab/ardrone_autonomy.git
5. https://github.com/angelsantamaria/tum_simulator.git
6. <https://www.tensorflow.org/install>
7. <https://www.luisllamas.es/machine-learning-con-tensorflow-y-keras-en-python/>
8. <https://www.nvidia.com/es-es/about-nvidia/>
9. <https://docs.nvidia.com/deploy/cuda-compatibility/>
10. <https://towardsdatascience.com/tensorflow-gpu-installation-made-easy-ubuntu-version-4260a52dd7b0>