

**DEPARTEMENT INFORMATIQUE 2015-2016**

# **Projet Tutoré**

---

**Thème : *Drones et véhicules terrestres***

---



Réalisé par Mourad HAMOUNI & Smail KESRAOUI

Dirigé par : Adrien Revault d'Allonnes

## *Sommaire*

<b>I.</b>	<b>Introduction.....</b>	<b>1</b>
<b>II.</b>	<b>Historique des drones .....</b>	<b>1</b>
<b>III.</b>	<b>Les différents types de Drones.....</b>	<b>2</b>
	1- Drones militaires.....	2
	2- Drones classiques.....	2
<b>IV.</b>	<b>Les Caractéristiques Ar-Drone 2 .....</b>	<b>3</b>
<b>V.</b>	<b>Système d’exploitation pour robots (ROS).....</b>	<b>5</b>
<b>VI.</b>	<b>Diagramme de cas d’utilisation .....</b>	<b>8</b>
<b>VII.</b>	<b>Simulation avec TurtleSim sous ROS .....</b>	<b>8</b>
<b>VIII.</b>	<b>Résultats de la simulation .....</b>	<b>9</b>
<b>IX.</b>	<b>Solution proposée pour Télé opéré le drone avec ROS.....</b>	<b>10</b>
<b>X.</b>	<b>Commandes de télé-Opération du drone.....</b>	<b>10</b>
	1- Démarrer le drone.....	10
	2- Arrêter le drone .....	10
	3- Redémarrer le système du drone .....	10
<b>XI.</b>	<b>Etat d’avancement .....</b>	<b>11</b>
<b>XII.</b>	<b>Conclusion.....</b>	<b>11</b>
<b>XIII.</b>	<b>Perspective .....</b>	<b>12</b>

## **I. Introduction :**

Les drones connaissent à l'heure actuelle un vif succès dans le monde civil aussi bien auprès des particuliers qu'auprès des professionnels. Le marché des drones est en pleine expansion et des applications de plus en plus variées voient le jour.

Dans un premier axe, nous tenterons de comprendre ce que nous entendons par drone. A l'heure actuelle, beaucoup parlent des drones sans réellement savoir ce qu'il en est sur le plan technique.

Le second axe de cette étude portera sur l'environnement de développement ROS (**Robot Operating System**) puis de télé opère TurtleSim avec clavier existant et module autonome de télé-opération.

En fin parvenir à connecter via la plateforme ROS et faire télé commandé le drone

## **II. Historique :**

Le drone est un aéronef télécommandé, c'est-à-dire sans pilote à bord. Il embarque une charge utile qui lui permet de réaliser des missions diverses et variées : surveillance, renseignement, cartographie, transport, vidéo...

La conceptualisation du drone remonte à la fin de la Première Guerre Mondiale. Alors qu'aux Etats-Unis se développe le projet Hewitt-Sperry automatic airplane, en France George Clémenceau, alors Président de la Commission sénatoriale de l'Armée, lance un projet

« d'avions sans pilote » : le capitaine Max Boucher met au point un système de pilotage automatique qui fait voler sur plus de cent kilomètres un avion Voisin BN3. Ainsi, dans les années 1920, des avions sans pilote radio-commandés voient le jour, avec les tentatives de torpilles aériennes télécommandées par des ondes de télégraphie sans fil.

- Les premiers drones apparurent en France dans les années 1960, tel le R 20 de Nord-Aviation, dérivé de l'engin cible CT 20. Mais les exemples significatifs d'une utilisation opérationnelle des drones sont encore peu nombreux.

Pendant la guerre du Vietnam, les Américains ont utilisé des drones (Firebee) pour localiser les rampes de lancement des missiles sol-air soviétiques «SAM-2» : 3500 missions furent recensées.

Plus tard, en 1991, lors de la guerre du Golfe, ils ont fait appel au drone (Pioneer) pour la surveillance jour/nuit, l'acquisition des objectifs, et les réglages de l'artillerie. Dans ce même conflit, les Britanniques et les Français commencèrent à servir des drones.

Le secteur de la défense constitue la principale raison de l'apparition des drones, qui seront par la suite réutilisés dans le domaine civil. Depuis plus de vingt ans maintenant, les drones ont été développés et utilisés en France dans un cadre militaire, pour des missions de surveillance et de renseignement. Suivant l'exemple de nombreuses innovations majeures initialement militaires (le satellite, le moteur à réaction, le GPS, internet), la technologie des drones a été adaptée depuis quelques années au secteur civil.

Même s'il reconnaît que le marché civil se développera rapidement, les usages militaires resteront dominants, Ils demeureront ainsi à 72 % militaire, à 23 % loisir et à 5 % civil professionnel

### **III. Les différents types de Drones :**

Après cette petite introduction historique nous allons à présent découvrir les différents types et utilisations des drones actuels :

#### **1. Le drone militaire :**

Le drone militaire est utilisé pour des missions de reconnaissance, généralement à hauts risques pour les soldats. Il est très en vogue chez les grandes puissances militaires comme la France, les Etats-Unis ou l'Angleterre.

De plus il permet une discrétion inégalée. En effet actuellement des drones patrouillent au proche orient pour démanteler des réseaux de terroristes. En temps de guerre il est utilisé pour l'espionnage mais peut également servir comme lance-missiles.



Figure 1 : General Atomics MQ-9 Reaper

#### **2. Drone Classique :**

##### **Drone Classique :**

Ce drone peut être acheté par des particuliers pour se divertir, Il est généralement utilisé pour filmer des paysages d'un point de vue nouveau. Le seul point négatif est qu'il peut permettre l'espionnage de ses voisins ou autres.



Figure 2 : L'AR. Drone 2.0

L'AR. Drone 2.0 est un drone accessible au grand public. D'une valeur d'environ 300 euros, il a une autonomie d'environ 12 minutes. Comme dans la plupart des drones des commandes permettent de réaliser des figures préenregistrées pour permettre un amusement total à l'utilisateur. De plus il possède une caméra HD qui permet de réaliser de bons films. Différents capteurs permettent au drone de garder une stabilité quel que soit l'altitude.

On peut citer beaucoup d'autres Drone qui existent sur le marché :

- ✓ **Le drone aquatique.**
- ✓ **Le drone sauveteur.**
- ✓ **Le drone lorrain.**
- ✓ **L'octocopter.**
- ✓ **Le nanodrone.**

C'est sur ce type de drone qu'on a pratiqué lors de notre projet, et nous allons définir les différentes caractéristiques **AR. Drone 2**

#### **IV. Les caractéristiques AR. Drone 2 :**

##### **Système électronique embarqué**

- Processeur ARM Cortex A8 Cadencé @1 GHz
- DSP Vidéo Cadencé @800 MHz
- 128 Mo de RAM DDR2 à 200 MHz
- 128 Mo de Flash
- Wi-Fi b/g/n
- USB high speed (2.0)
- Linux OS 2.6.32

##### **Système de guidage inertiel**

- Accéléromètre MEMS 3-axes
- Gyroscope à 3-axes
- Magnétomètre à 3-axes
- Capteur de Pression

##### **Spécifications**

- Vitesse en vol : 5 m/s ; 18 km/h (16,4 fps ; 11,2 mph)
- Poids :
  - 380 g avec la coque d'extérieur
  - 420 g avec la coque d'intérieur
- Autonomie : environ 12 minutes

##### **Sécurité**

- Coque en EPP pour les vols en intérieur.
- Arrêt automatique des hélices en cas de contact
- Batterie UL2054
- Interface de contrôle avec un bouton d'arrêt d'urgence (coupe l'alimentation des moteurs)

##### **Structure aéronautique**

- Hélices à haute efficacité spécifique.

- Structure en fibre de carbone.

## Moteurs et énergie



- 4 moteurs brushless (28 500 tr/min, puissance : 14,5 W)
- Batterie Lithium-polymère (trois cellules ; 11,1 V ; 1 000 mAh)
- Capacité de décharge : 10 C
- Temps de rechargement de la batterie : 90 minutes

## Caméra frontale

- Caméra grand angle 92°, capteur CMOS
- Enregistrement et diffusion directe des images sur iPhone ou sur clef USB
- Résolution caméra de 1280x720 pixels @30 fps (HD)

## Caméra Verticale

- Caméra à haute vitesse. 64° diagonale de la lentille, Capteur CMOS
- Enregistrement et diffusion directe des images sur iPhone ou sur clef USB
- Résolution de 320x240 (QVGA) @60 fps
- Utilisée pour mesurer la vitesse au sol
- Permet la stabilisation, même avec un vent léger

## Autre détecteurs de l'AR.Drone

- Validation des tirs des drones ennemis
- Estimation de la distance
- Positionnement d'objets virtuels
- Calcul des marqueurs d'objets virtuels
- Distance de détection : de 30 centimètres à 5 mètres

## Altimètre à ultrason

- Fréquence d'émission : 40 kHz
- Portée de 6 mètres (19,7 pieds)
- Sensibilité vert

## V. Système d'exploitation pour robots (ROS)

**ROS** : est un système d'exploitation pour robots. De même que les systèmes d'exploitation pour PC, serveurs ou appareils autonomes, ROS est un système d'exploitation complet pour la robotique de service.

Il fournit des services proches d'un système d'exploitation (abstraction du matériel, gestion de la concurrence, des processus...) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base de données centralisée de données, système de paramétrage du robot...).

### ➤ L'organisation générale de ROS :

ROS se résume en 5 grands principes suivants :

**Peer to Peer** : Une architecture peer to peer couplée à un système de tampon (buffering) et un système de lookup (un nom de service appelé master dans ROS), permet à chacun des acteurs du robot (Système embarqué) de dialoguer en direct avec un autre acteur, de manière synchrone ou asynchrone en fonction des besoins.

**Basé sur des outils (micro kernel)** : ROS a adopté un design microkernel qui utilise un grand nombre de petits outils pour faire le build et le run des différents composants ROS.

**Multi langages** : peut être programmé en différents langages. La spécification de ROS intervient au niveau message. Les connexions peer to peer sont négociées en XML-RPC qui existe dans un grand nombre de langages. Ces messages sont décrits en IDL (Interface Definition Language).

**Léger** : ROS utilise du code (pilotes et algorithmes) issus d'autres projets open source :

Simulateur Player/stage

Librairie de traitement d'image et de vision artificielle : OpenCV

Algorithme de planification : OpenRave

**Gratuit et open source** : ROS passe des données grâce à de la communication interprocess. De ce fait, les modules n'ont pas besoin d'être liés dans un unique process, facilitant ainsi l'usage des différentes licences.

## **Les autres systèmes d'exploitation généralistes pour robots :**

**Microsoft Robotics Developer Studio** : système multiplateforme, il fournit un outil de simulation. Il n'est cependant compatible qu'avec Windows et se programme avec un langage managé .NET (C# de préférence).

**NAOQi** : système robotique réalisé pour le robot NAO par Aldebaran Robotics. Se programme en C++ ou Python. NAOQi est open source.

**URBI** : Il est open source et propose son propre langage de script (URBIScript). Il se programme en C++ également. URBI est multiplateformes.

Il existe ensuite de nombreux systèmes embarqués spécifiques à un robot en particulier.

### **➤ Les grands principes de ROS :**

**Programmer avec ROS** : ROS n'est pas dépendant d'un langage. Trois bibliothèques principales ont été définies pour ROS qui permettent de programmer respectivement ROS en Python, en Lisp ou en C++. En plus de ces trois bibliothèques, deux bibliothèques expérimentales sont proposées, qui permettent de programmer ROS en Java ou en Lua .

### **Le système de fichiers de ROS :**

Les ressources de ROS sont organisées dans une structure hiérarchique sur disque. Deux concepts importants se détachent :

**Le package** : C'est l'unité principale d'organisation logicielle de ROS. Est un répertoire qui contient les nœuds, les bibliothèques externes, des données, des fichiers de configuration et un fichier de configuration xml nommé manifest.xml.

**La stack** : Une stack est une collection de packages. Elle propose une agrégation de fonctionnalités telles que la navigation, la localisation... Une stack est un répertoire qui contient les répertoires des packages ainsi qu'un fichier de configuration nommé stack.xml.

### **➤ Les dernières distributions en date de ROS :**

**ROS Fuerte**, publiée le 23 Avril 2012

**ROS Electric Elys**, publiée le 30 août 2011

**ROS Diamondback**, publiée le 2 mars 2011

**ROS C Turtle**, publiée le 2 août 2010

**ROS Box Turtle**, publiée le 2 mars 2010

### **➤ Les notions de base de ROS :**

**Les nœuds** : Un nœud est une instance d'un exécutable et peut correspondre à un capteur, un moteur, un algorithme de traitement, de surveillance. Chaque nœud qui se lance se déclare au Master. On retrouve ici l'architecture micro kernel où chaque ressource est un nœud indépendant.



**Le Master :** Le Master est un service de déclaration et d'enregistrement des nœuds qui permet ainsi à des nœuds de se connaître et d'échanger de l'information. Le Master est implémenté via XMLRPC.

Le Master comprend une sous-partie très utilisée qui est le Parameter Server, comme son nom l'indique est une sorte de base de données centralisée dans laquelle les nœuds peuvent stocker de l'information et ainsi partager des paramètres globaux.

### **Les topics :**

L'échange de l'information s'effectue soit de manière asynchrone via un topic ou de manière synchrone via un service.

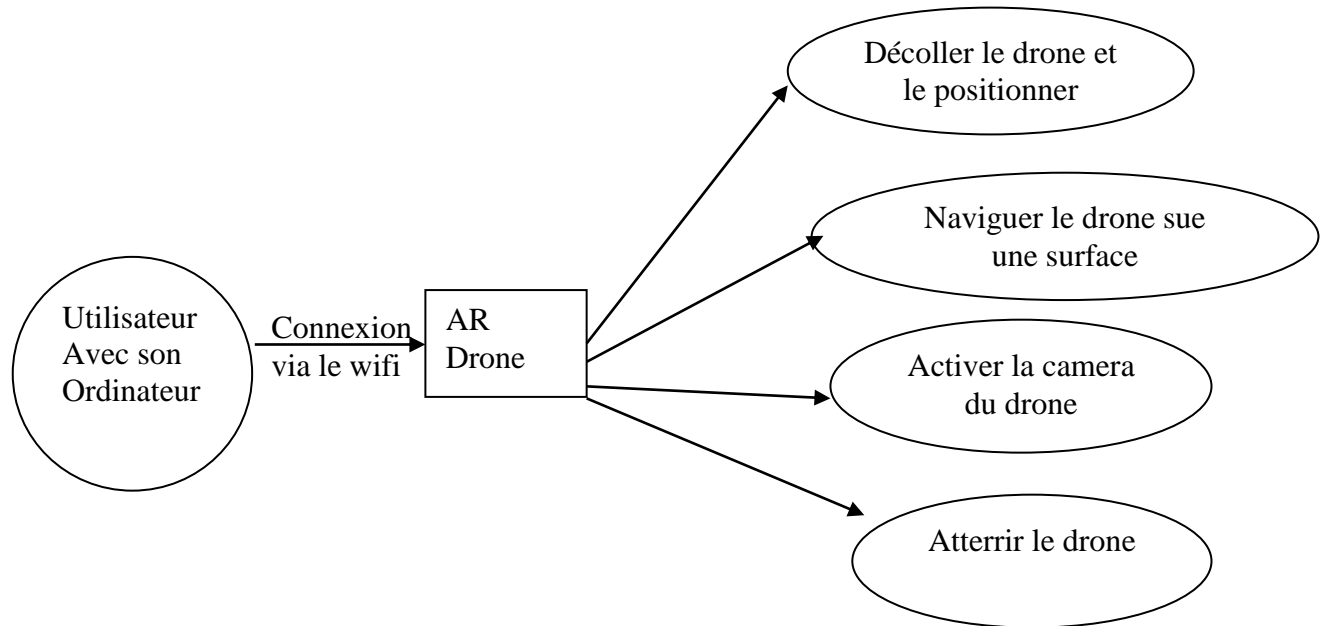
Un topic est un système de transport de l'information, en quelque sorte un bus d'information asynchrone, Les nœuds envoient ou reçoivent des messages sur des topics

**Les messages :** Un message est composé d'une combinaison de types primitifs (chaines de caractères, booléens, entiers, flottants...) et de message (le message est une structure récursive).

La description des messages est stockée dans `nom_package/msg/monMessageType.msg`. Ce fichier décrit la structure des messages.

**Les services :** Le service en revanche répond à une autre nécessité, celle d'une communication synchrone entre deux nœuds, la description des services est stockée dans `nom_package/srv/monServiceType.srv`

## VI. Diagramme de cas d'utilisation :



## VII. Simulation avec TurtleSim sous ROS :

La simulation permet ainsi de tester sans aucun coût ces nouvelles technologies, et d'anticiper les problèmes qui pourront se poser dans le futur.

Dans cette partie notre objectif est d'apprendre à utiliser ROS et la prise en main, puis de télé-opérer turtlesim avec le clavier existant et module autonome de télé-opération, par la suite nous allons présenter notre simulation par la présentation d'un scénario.

- **Pour ouvrir le turtleSim on écrit :**  
`>>roslaunch turtlesim turtlesim_node`
- **Pour faire bouger la tortue avec le clavier on utilise la commande suivante :**  
`>>roslaunch turtlesim turtle_teleop_key`

- **créer un module autonome de télé-opération :**

**Il faut créer le publisher-subscriber :**

```
>> cd ~/catkin_ws/src
>> catkin_create_pkg publisher_subscriber std_msgs roscpp message_generation
message_runtime

>> cd publisher_subscriber
```

- **Éditer le package.xml et ajouter :**  
[ `<build_depend>message_generation</build_depend>` ]  
[ `<run_depend>message_runtime</run_depend>` ]
- **Dans le beginner\_simulation/src On crée le fichier subscriber\_tsim.cpp, ainsi mettre code source.**

**Éditer le fichier CMakeLists.txt et ajouter le Publisher créé :**

```
add_executable(pubtsim src/publisher_tsim.cpp)
target_link_libraries(pubtsim ${catkin_LIBRARIES})
add_dependencies(pubtsim beginner_Simulation_generate_message_cpp)
```

Puis on créer le fichier `publisher_tsim.cpp` qui permet de télé-opérer le turtlesim dans le fichier `beginner_simulation/src`

- **Pour compiler et mettre à jour :**

```
cd catkin_ws  
catkin_make
```

- **Pour le roscore :**

```
>>cd catkin_ws  
>>catkin_make  
>>roscore
```

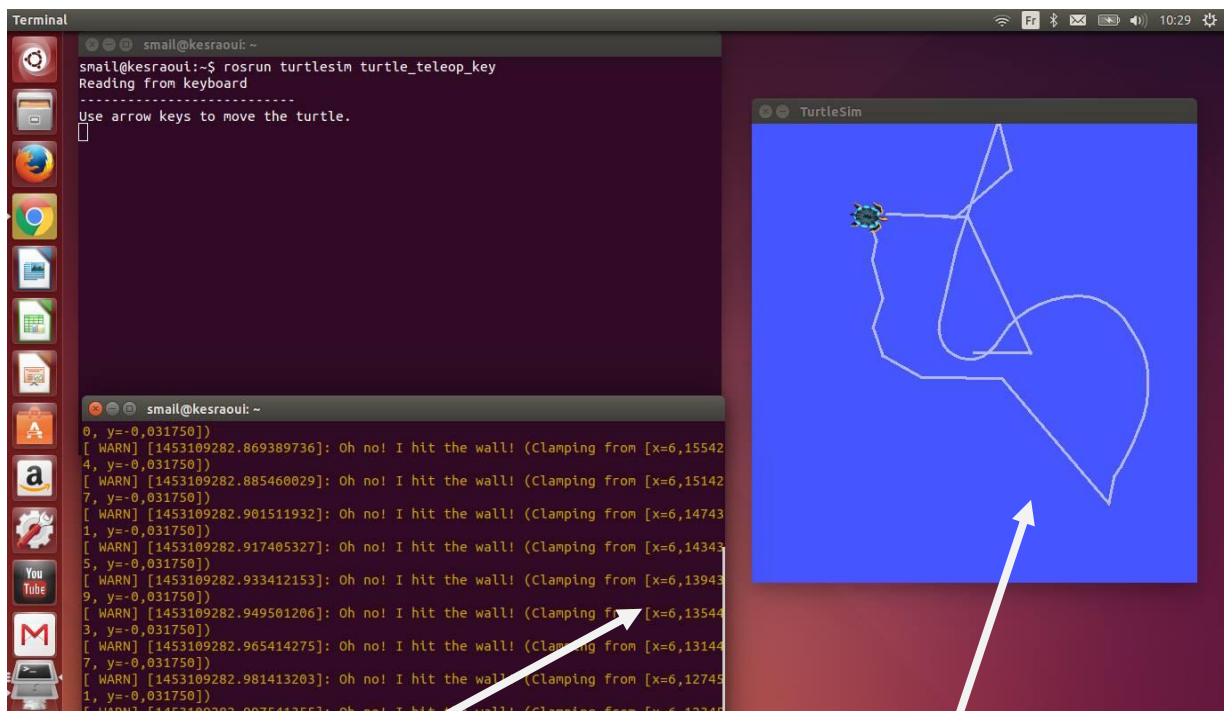
- **Appeler les ressources :**

```
source ~/catkin_ws/devel/setup.sh
```

- **Démarrer le module turtlesim :**

```
roslaunch beginner_Simulation pubtsim
```

## VIII. Le résultat de la simulation :



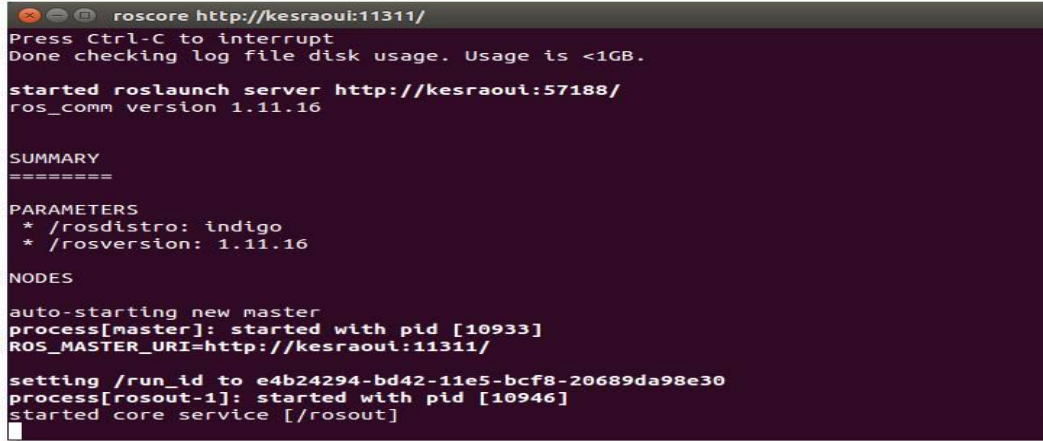
Affichage des  
positions X,Y

Surface de  
navigation du robot

## **IX. Solution proposée pour Télé opéré le drone avec ROS :**

### **1- Se connecter à l'AR-drone :**

Lancer le noyau ROS avec roscore

A terminal window with a dark purple background and white text. The text shows the output of the 'roslaunch' command. It starts with 'roscore http://kesraoui:11311/' and 'Press Ctrl-C to interrupt'. It then says 'Done checking log file disk usage. Usage is <1GB.' followed by 'started roslaunch server http://kesraoui:57188/' and 'ros\_comm version 1.11.16'. Below this is a 'SUMMARY' section with 'PARAMETERS' listing '/rostdistro: indigo' and '/rosversion: 1.11.16'. The 'NODES' section shows 'auto-starting new master', 'process[master]: started with pid [10933]', 'ROS\_MASTER\_URI=http://kesraoui:11311/', 'setting /run\_id to e4b24294-bd42-11e5-bcf8-20689da98e30', 'process[rosout-1]: started with pid [10946]', and 'started core service [/rosout]'.

```
roscore http://kesraoui:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://kesraoui:57188/
ros_comm version 1.11.16

SUMMARY
=====
PARAMETERS
* /rostdistro: indigo
* /rosversion: 1.11.16

NODES
auto-starting new master
process[master]: started with pid [10933]
ROS_MASTER_URI=http://kesraoui:11311/

setting /run_id to e4b24294-bd42-11e5-bcf8-20689da98e30
process[rosout-1]: started with pid [10946]
started core service [/rosout]
```

### **2- Se connecter au drone via le wifi du pc à l'AR-drone :**

Dans un second terminal, nous allons lancer rosrn avec des paramètres, dont il va afficher l'adresse IP de l'AR-drone.

**\$ rosrn ardrone\_autonomy ardrone\_driver**

- 3- Pour utiliser le driver ardrone\_autonomy une fois celui-ci installé avec la commande « **sudo apt-get install ros-indigo-ardrone-autonomy** » qui nous permet d'importer le package ardrone\_autonomy
- 4- Puis avec la commande **rostopic list** pour récupérer un topic à partir de la ligne de commande

## **X. Commandes de télé-opération du drone :**

### **1. Demarrage de l'AR-DRONE en ligne de commande :**

Pour lancer l'AR-drone, on utilise la commande :

**\$rostopic pub -1/ardrone/takeoff std\_msgs/Empty**


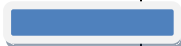



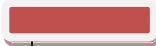
Pour arrêter l'AR-drone, On utilise la commande :

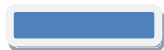
**\$ rostopic pub -1/ardrone/land std\_msgs/Empty**

Pour redémarrer le système du l'AR\_drone, on utilise la commande :

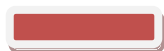
**\$rostopic pub -1/ardrone/reset std\_msgs/Empty**

## **XI. Etat d'avancement :**

	Decembre	Janvier	Mars	Avril	Mai	Juin
Etude et analyse						
Installation de l'environnement et simulation						
Rédaction du premier rapport						
Implémentation						
Tests						
Exposer final						



Les taches réalisées



Les taches en cours de réalisations

## **XII. Conclusion :**

Dans cette partie de notre travail on a réussi a ce familiariser avec le système d'exploitation ROS la ou on a simulé avec TurtlSim et de faire commander un drone a partir de notre Pc avec des commandes.

### **XIII .Les perspectives :**

- faire stabiliser le drone pendant une période de temps puis l'atterrir.
- spécifier la surface de vole
- Activer la camera du drone et gérer les réactions face aux obstacles avec toute autonomie

# *Webgraphie*

[Historique des drones]

<http://www.federation-drone.org/les-drones-dans-le-secteur-civil/histoire-du-drone/>

[ROS]

[www.ROS.org](http://www.ROS.org)

<http://www.generationrobots.com/fr/content/55-ros-robot-operating-system>

[https://github.com/AutonomyLab/ardrone\\_autonomy/tree/indigo-devel/src](https://github.com/AutonomyLab/ardrone_autonomy/tree/indigo-devel/src)