PRESENTATION FINLAE P22
COMMANDE EN POSITION D'UN DRONE

Tuteur : Monsieur Komi Midzodzi PEKPE

Binômes : Lijie YAO et Lirui ZHANG

SOMMAIRE

- L'introduction
- Ce que nous avons fait
  - L'étude sur le drone
  - L'étude d'environnement de travail
  - L'etude de logiciel et d’interface
  - Commande et contrôle effectué sur le drone (Avec Pyparrot)
  - Recueillir les données renvoyées par le capteur (Avec pyparrot)
  - L'étude des commandes ROS (Avec bebop_autonomy)
  - ROStopic list et les explications de données
  - La relation entre les nœuds et les topics
  - Créez notre fichier python basé sur bebop_autonomy
  - Pulisher et Subscriber
  - Récupérer et Analyse de données
  - Limiter le mouvement et réguler la vitesse
  - L’interface
- Démonstration

L’INTRODUCTION

- Le but de notre projet est étude de faisabilité de quelques fonctions qui peut utilité pour des TP ou les recherches en tant que support.
- Dans la première moitié de nos études, nous avons mené des recherches et réalisé le contrôle en position du drone par une interface.
- Pour la seconde moitié du travail, nous avons utilisé la mécanisme de ROS pour contrôler le drone et réguler la vitesse.

L'ÉTUDE SUR LE DRONE

WIFI

La communication entre le drone et l'ordinateur ou le téléphone portable

FlightFree Pro

App utilisé par téléphone

radioélectrique

La communication entre le drone et la manette

ROS

Le système d'exploitation pour le drone

L'ÉTUDE SUR LE DRONE

- Une caméra horizontale qui prendre des photos et des vidéos
- Une caméra verticale qui permet le maintien d'un point fixe.
- Un capteur ultrason qui analyse l'altitude de vol jusqu'à 5 mètres.
- Un gyroscope 3 axes qui permet de calculer l'angle d'inclinaison de l'appareil.
- Un magnétomètre 3 axes qui donne la possibilité de définir la position du drone, à l'image d'une boussole.
- Un GPS et un GLONASS pour la géolocalisation du drone et aide à mesurer la vitesse de drone pour plus de stabilité à haute altitude.
- Un capteur de pression qui permet de mesurer la pression et de calculer l'altitude de vol lorsque celle-ci dépasse les 5 mètres.
- Un accéléromètre qui permet de mesurer l'orientation du drone sur 3 axes et sa vitesse linéaire.

https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjzh-KTmanfAhUszlUKHelSB34QjRx6BAgBEAU&url=https%3A%2F%2Fwww.lesnumeriques.com%2Fdrone%2Fparrot-bebop-2-p29821%2Ftest.html&psig=AOvVaw18v9aMGfyQKEi_2To7tlBi&ust=1545216225108124

L'ÉTUDE DE TRAVAIL D'ENVIRONNEMENT

C'est quoi

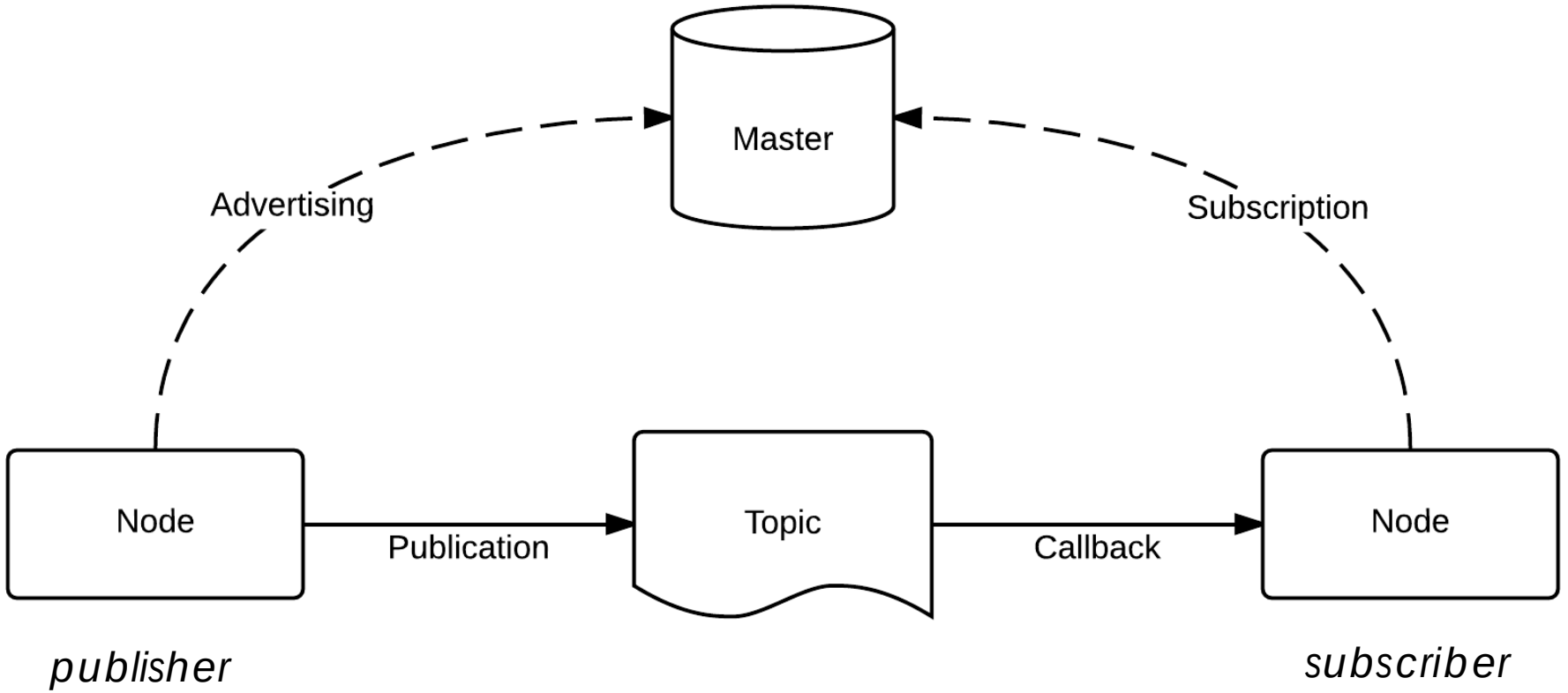
Un méta-système d'exploitation qui peut fonctionner sur un ou plusieurs ordinateurs et qui fournit plusieurs fonctionnalités.

Pourquoi

Pour faire contrôler le drone.
Il y a un ROS Kinetic dans le drone.

https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjn5tPHmanfAhVKzlUKHXaiBsoQjRx6BAgBEAU&url=https%3A%2F%2Fwww.udoo.org%2Fros-on-udoo-neo-dual-quad%2F&psig=AOvVaw1thkG6L89-k9Kr0FDhm_z6&ust=1545216332155304

L'ÉTUDE DE TRAVAIL D'ENVIRONNEMENT



https://fr.wikipedia.org/wiki/Robot_Operating_System#/media/File:ROS-master-node-topic.png

L'TUDE DE LOGICIEL ET D'INTERFACE

Parrot
For Developers

SDK Parrot

Connecter, piloter, recevoir des flux, enregistrer et télécharger des médias (photo et vidéo), envoyer et lire des plans de vol sur pilote automatique et mettre à jour notre drone.

Bebop Autonomy

Un drive développé sur SDK Parrot, utilisé pour lancer le pilote, envoyer de commandes à Bebop, lecture de Bebop, configuration de Bebop et du pilote

ANACONDA

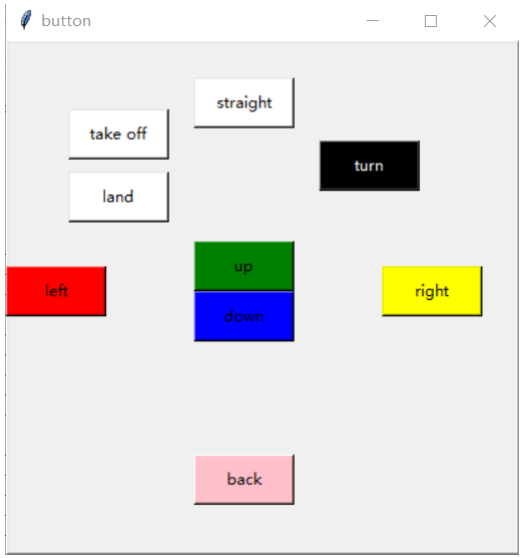
python

Python Pyparrot

Une interface de python pour Bebop basé sur Anaconda. Utilisé pour communiquer et contrôler le drone, utilisé également pour récupérer des données des capteurs.

COMMANDE ET CONTROLE EFFECTUE SUR LE DRONE (AVEC PYPARROT)

- Connecter et déconnecter le service de wifi
- Décollage et atterrissage
- Capteurs
- Caméra vidéo
- Vol et rotation
- Faire certaines limites(altitude, vitesse etc.)
- Pause ou sommeil
- Créer une interface pour contrôler (appuyer chaque bouton et déterminer le valeur de variable dans le terminal pour réaliser les différents fonctions)



RECUEILLIR LES DONNEES RENVOYEES PAR LE CAPTEUR (AVEC PYPARROT)

- Les limitations avec pyparrot

  - Données de retour manquantes pour la vitesse de rotation.

  - Chaque fois que nous voulons obtenir les données du capteur, nous devons appeler *Bebop.sensors.sensors_dict* séparément. Nous ne pouvons pas obtenir les informations en temps réel ni permanent.

  - Par conséquent, nous ne pouvons qu'abandonner cette méthode et réutiliser Bebop_autonomy.

L'ÉTUDE DES COMMANDES ROS (AVEC BEBOP_AUTONOMY)

- Utilisant de topic et de message

  - rostopic list

    Répertorier tous les sujets en cours de publication et d'abonnement.

  Nous prêtons attention aux sujets suivants:

    /bebop/cmd_vel

    /bebop/odom

    /bebop/states/ardrone3/PilotingState/AltitudeChanged

    /bebop/states/ardrone3/PilotingState/SpeedChanged

L'ÉTUDE DES COMMANDES ROS (AVEC BEBOP_AUTONOMY)

- rostopic echo [topic]

  Afficher les données publiées sur le sujet

L'ÉTUDE DES COMMANDES ROS (AVEC BEBOP_AUTONOMY)

- rostopic hz [topic]

  Afficher la fréquence de publication du sujet

Nous pouvons voir que la fréquence moyenne de données renvoyées par le capteur est d'environ 5hz.

L'ÉTUDE DES COMMANDES ROS (AVEC BEBOP_AUTONOMY)

Nous configurons principalement les paramètres suivants:

- PilotingSettingsMaxAltitudeCurrent
- PilotngSettingsMaxDistanceValue
- PilotingSettingsNoFlyOverMaxDistancesShouldnotflyover
- SpeedSettingsMaxRotationSpeedCurrent
- SpeedSettingsMaxVerticalSpeedCurrent

ROSTOPIC LIST ET LES EXPLICATIONS DE DONNÉES

- Les messages de type **geometry_msgs/Twist** sont publié à **cmd_vel** topic

```
linear.x  (+)  Traduire en avant
linear.x  (-)  Traduire en arrière
linear.y  (+)  Traduire à gauche
linear.y  (-)  Traduire à droite
linear.z  (+)  Monter
linear.y  (-)  Descendre
angular.z (+)  Tourner dans le sens antihoraire
angular.z (-)  Le sens des aiguilles d'une montre
```

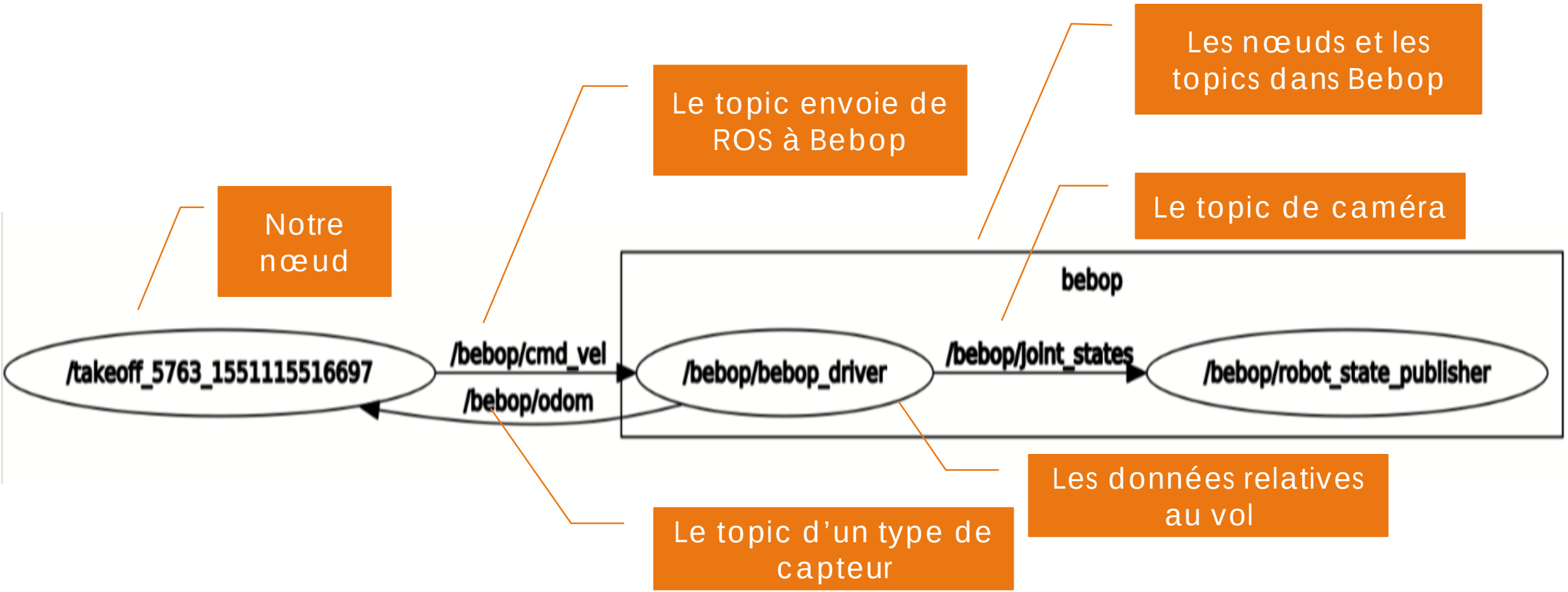
- Les valeurs du topic Odometry qui sont récupérées appartiennent à l'intervalle [-1;1]. C'est la pourcentage, avec le changement comme ci-dessous:

```
roll_degree      = linear.y  * max_tilt_angle
pitch_degree     = linear.x  * max_tilt_angle
ver_vel_m_per_s  = linear.z  * max_vert_speed
rot_vel_deg_per_s = angular.z * max_rot_speed
```

```
'PilotingSettingsMaxTiltCurrent': 20.0
'SpeedSettingsMaxVerticalSpeedCurrent': 1.0
'SpeedSettingsMaxRotationSpeedCurrent': 100.0
```

degrée
m/s
degrée/s

LA RELATION ENTRE LES NŒUDS ET LES TOPICS



CRÉER NOTRE FICHIER PYTHON BASE SUR BEBOP_AUTONOMY

- Utiliser rospy pour implémenter Pulisher et Subscriber
  - rospy.info('%f',[paramètre])
  - rospy.Publisher([topic],[type],queue_siez=?)
  - def callback
  - rospy.Subscriber([topic],[paramètre],[callback])
  - rospy.init_node([nom],anonymous=True)
  - rospy.spin()
  - sleep()

PULISHER ET SUBSCRIBER

```
def turn(angular_speed):
     pub = rospy.Publisher("bebop/cmd_vel", Twist, queue_size=1, latch=True)
     move_cmd = Twist()
     move_cmd.angular.z = angular_speed
     pub.publish(move_cmd)
```

```
def callback_AttitudeChanged(msg):
    ang_yaw = msg.yaw
    rospy.loginfo('real angle of rad between -3.14 to +3.14 is %f', ang_yaw)
    input_to_file_ang_yaw([ang_yaw])
```

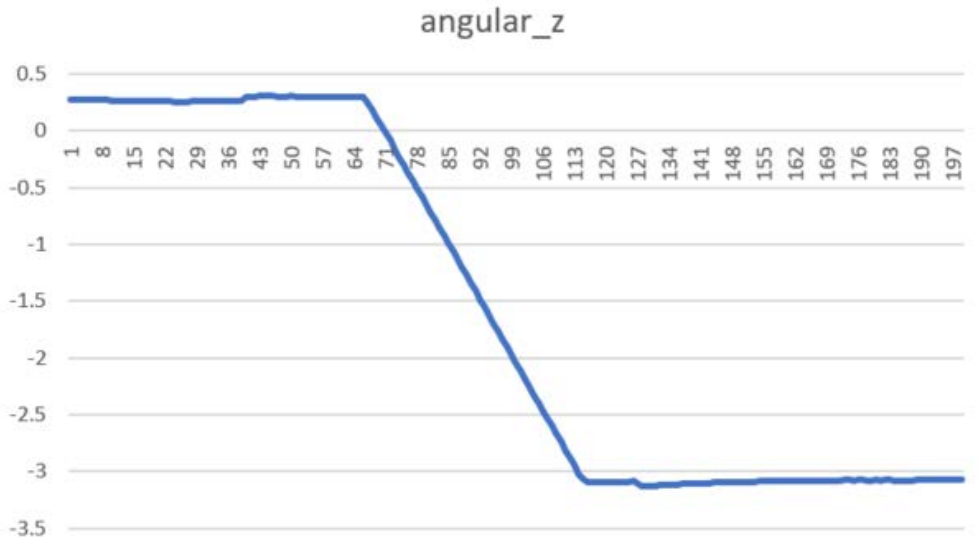
```
def get_vitesse_rotation():
    str_input = entry2.get()
    if (str_input.replace('.', '', 1)).replace('-', '', 1).isdigit():
        float_input = float(str_input)
        if (-1.0<=float_input<=1.0):
            while not rospy.is_shutdown():
                turn(float_input)
        else:
            print('%s faut dans l\'intervalle de [-1.0,1.0]' % str_input)
    else:
        print('%s n\'est pas un chiffre !' % str_input)
```

```
def listener():
    rospy.Subscriber('/bebop/states/ardrone3/PilotingState/AttitudeChanged',
                     Ardrone3PilotingStateAttitudeChanged, callback_AttitudeChanged)
```

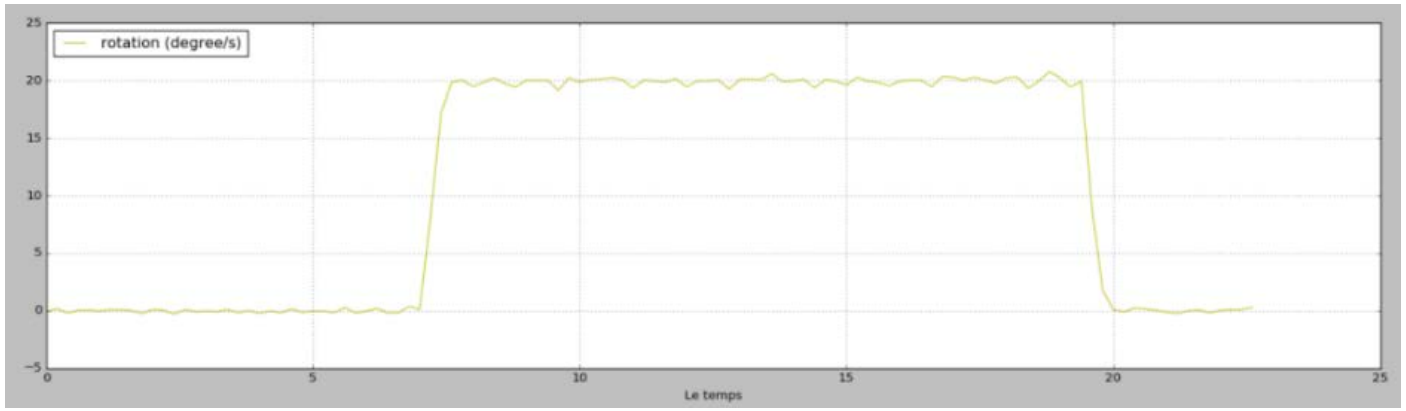
```
Boutton_trun=Button(win,text="valider la vitesse de rotation",command=get_vitesse_rotation)
Boutton_trun.pack()
entry2=Tkinter.Entry(win,width=50,bg="white",fg="black")
entry2.pack()
```

RÉCUPÉRER ET ANALYSE DE DONNEES

- rostopic echo /bebop/states/ardrone3/PilotingState/AttitudeChanged/yaw



- Par calcul:
  deg = [angular(t)-angular(t-1)]*57.32/0.2



LIMITER LE MOUVEMENT ET RÉGULER LA VITESSE

- Limiter le mouvement
- la hauteur de vol maximale à 150m
- Les limites du déplacement horizontal ne peut pas être mis inférieur à 10m
- La vitesse de rotation maximale égale à 200 degré/s
- Faire une validation

```
rosrun dynamic_reconfigure dynparam set /bebop/bebop_driver
PilotingSettingsMaxAltitudeCurrent 2.0 (in m))
```

```
rosrun dynamic_reconfigure dynparam set /bebop/bebop_driver
PilotingSettingsMaxDistanceValue 5.0 (in m)
```

```
PilotingSettingsNoFlyOverMaxDistanceShouldnotflyover = 1
```

LIMITER LE MOUVEMENT ET RÉGULER LA VITESSE

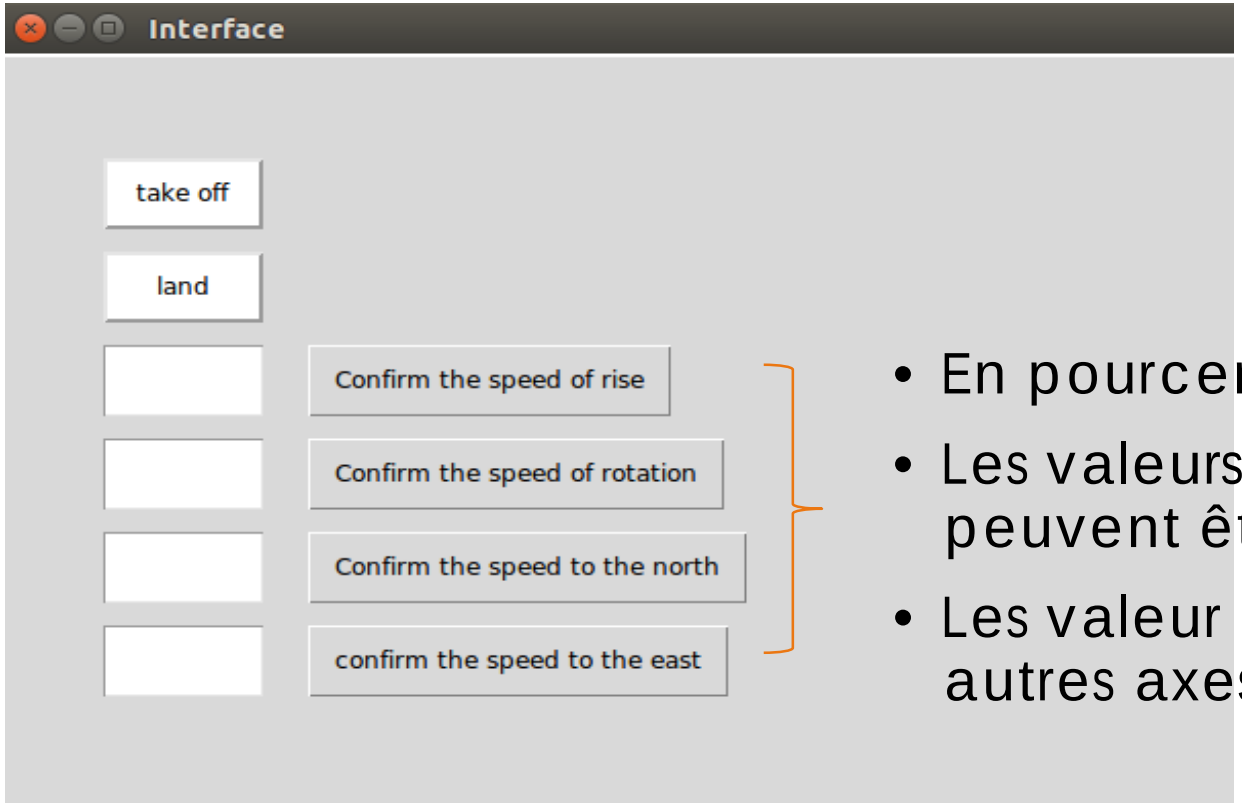
- Réguler la vitesse

- Lire les données viennent le clavier
  - Utilisation d'une boucle
  - Utilisation d'une rupture

```
def get_vitesse_verticale():
    str_input = entry1.get()
    if (str_input.replace('.', '', 1)).replace('-', '', 1).isdigit():
        float_input = float(str_input)
        if (-1.0<=float_input<=1.0):
            i = 0
            while i < 50:
                up(float_input)
                i += 1
                sleep(0.2)
        else:
            print('%s faut dans l\'intervalle de [-1.0,1.0]' % str_input)
    else:
        print('%s n\'est pas un chiffre !' % str_input)
```

```
def get_vitesse_go_north():
    str_input = entry3.get()
    if (str_input.replace('.', '', 1)).replace('-', '', 1).isdigit():
        float_input = float(str_input)
        if (-1.0<=float_input<=1.0):
            #i = 0
            #while i < 10:
            while not rospy.is_shutdown():
                go_north(float_input)
                #i += 1
                #sleep(0.2)
        else:
            print('%s faut dans l\'intervalle de [-1.0,1.0]' % str_input)
    else:
        print('%s n\'est pas un chiffre !' % str_input)
```

L'INTERFACE



- En pourcentage, dans une intervalle [-1,1]
- Les valeurs maximales de la vitesse sur l'axe Z peuvent être changé
- Les valeur maximale de la Vitesse sur les deux autres axes ne peuvent pas être changé

DÉMONSTRATION

Merci de votre attention!