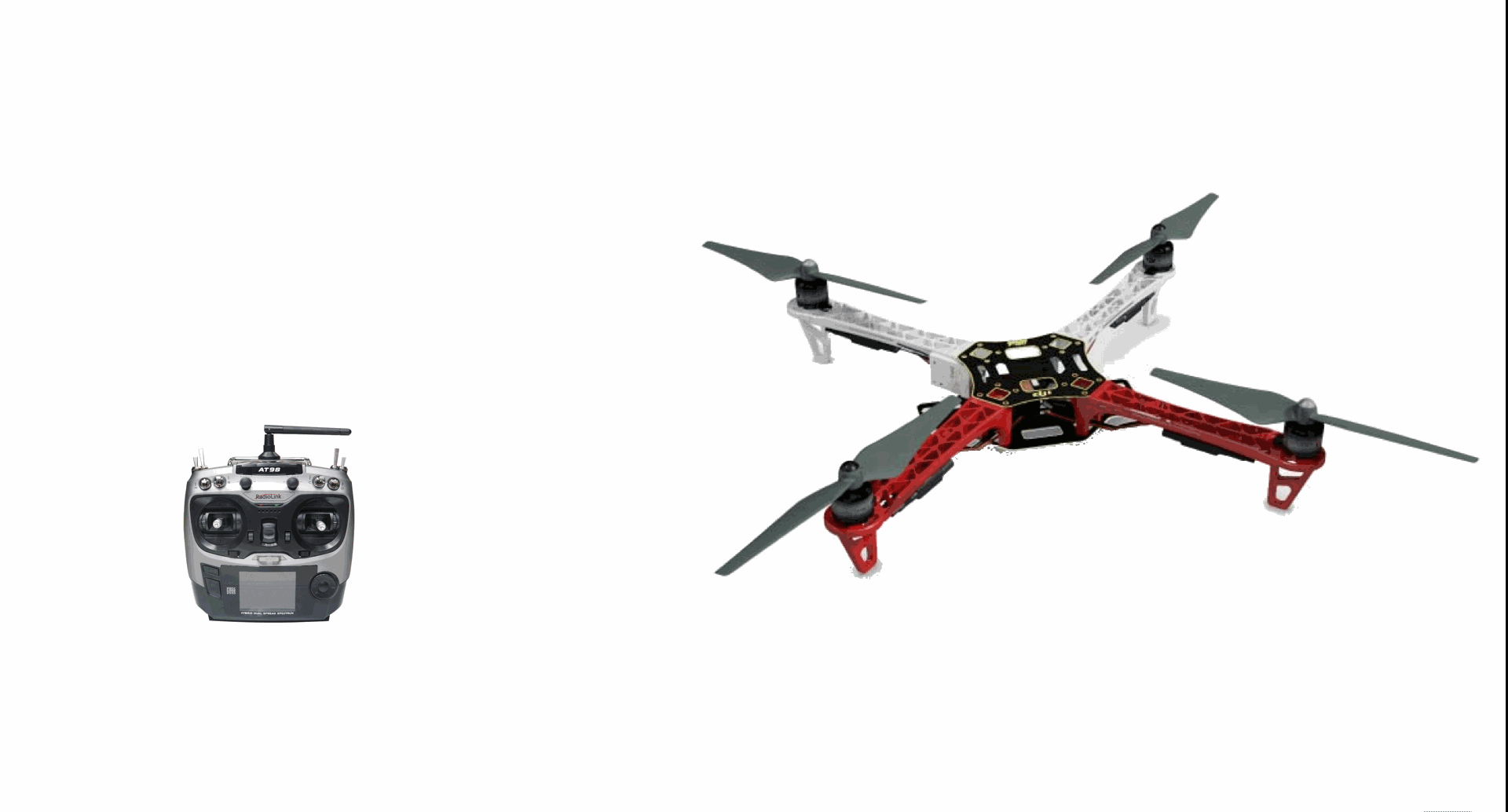
Johan Lael

Groupe 8

MEZATIO KENFACK GUISELAINE  
YEPSI ANGE YVANA  
AYENG OWONO GLADYS  
DJIENTCHEU DJOKM MEGANE  
KAPNANG KETCHEUZEU JOHAN LAEL  
SOP COLMAR

MIA SOUTENANCE

DOMAINE MIA



# NOTION

Le domaine MIA est concerné sur l’innovation(électronique) et la simulation de vol de notre drone. C’est-à-dire que, avec ce domaine on voudrait remplacer l’être humaine commandant le drone via une télécommande afin de rendre le drone autonome. Apres l’innovation il est question de simuler le vol du drone autonome afin d’étudier sa réaction en temps réel.

# OBJECTIFS

L'objectif ici est d’écrire un programme intelligent qui va remplacer la télécommande et l’humain qui actionne la télécommande et pouvoir le simuler afin d’assurer son efficacité en temps réel.

# DESCRIPTION

Donc notre programme devra

1. Demander à l’utilisateur de saisir les adresses qu’il souhaite visiter
2. Convertir les adresses en coordonnées GPS
3. Faire décoller le drone
4. Visiter une à une les adresses
5. Revenir se poser sur le site départ

# PLAN D’ACTION

## LOGICIELS A INSTALLER

Pour commencer nous avons eu à télécharger trois logiciels

## PYTHON

Python est le langage de programmation que nous avons utilisé pour programmer.

## ECLIPSE/PyDev

Eclipse nous permettra de développer en python et PyDev crée l’environnement de développement dans le quel on écrira notre programme.

## MISSION PLANER

Mission Planner est une application de station au sol complète adaptée au contrôleur de vol que nous utilisons dans notre drone. Mission planer nous permettra de simuler le vol de notre drone et de le surveiller en vol.

Mission planner peut être utilisé pour de différentes choses comme

* Charger le logiciel dans le contrôleur de vol.
* Paramétrer le drone pour des performances optimales.
* Planifier, sauvegarder et charger des missions autonomes.
* Télécharger et analyser les journaux de mission créés par le contrôleur de vol.
* S’interfacer avec un simulateur de vol PC pour créer un simulateur complet de drones.

Mais dans notre cas il sera utilisé pour

* Simuler le vol de notre drone
* Surveiller l’état du drone pendant son fonctionnement
* Conduire le drone en FPV

## PROCESSUS

Des ressources nous ont été donné et pour arriver a notre code finale nous avons ;

## Téléchargé la bibliothèqueque utilisé

En ouvrant l’invite de commande et en exécutant le code pip install dronekit\_still future geocoder dans le dossier scripts, nous avons téléchargé et installer la bibliothèque que nous allons utiliser.

## Créé un nouveau projet dans Eclipse

Nous avons ensuite créé un nouveau projet dans Eclipse et un nouveau dossier ou nous avons mis tous nos ressources c’est-à-dire ; hello\_drone, simple\_goto, quadcopter\_simulator.

## Exécuté les programmes

Les programmes hello\_drone et simple\_goto sont les programmes clefs de notre processus. Nous les avons exécutés, corrigé les erreurs et nous avons mis à jour les programmes.

## Testé et modifié

Avec ces programmes nous avons lancé la simulation afin de visualiser le fonctionnement. Ensuite nous avons introduit le géocodage et écrit le programme de la façon tel que expliqué ci dessous.

## PROGRAMMATION

Notre programme, comme dit plus haut aura différentes fonctions dans un ordre chronologique. Mais il y aura quelques fonctions secondaires très importantes comme

* Pouvoir débuter son trajet d’un point de départ souhaité (c’est-à-dire qu’il devrait pouvoir aller au point de départ avant de commencer son trajet)
* Remettre le control du drone a la télécommande en cas de besoin
* Pouvoir retourner au point de départ en cas de besoin (lorsqu’il est hors de portée par exemple)

Nous programmons avec le langage python. Avant de commencer nous allons créer l’environnement, installer le géocoder et lancer un nouveau projet PyDev. Le programme devra

## Demander à l’utilisateur de saisir les adresses qu’il souhaite visiter

La première fonction de notre programme sera de demander les adresses à l’utilisateur. Ceci se fera à l’aide de la fonction **input () ;**

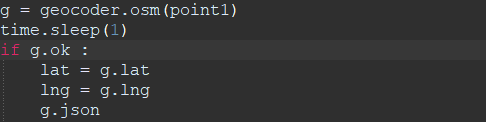
Adresses= Input (‘Entrer les adresses s’il vous plait : ’)

Print (‘les adresses sont’, Adresses)



## Convertir les adresses en coordonnées GPS

Convertir les adresses en coordonnées GPS est référé comme le Géocodage. Ceci se fera dans les étapes suivantes ;



* La fonction input () récupère l’adresse saisie par l’utilisateur
* On demande ensuite à OpenStreetMap (une alternative open source à Google Maps) de nous donner les coordonnées GPS correspondant à cette saisie
* On vérifie la validité du résultat (Open Street Map a-t-il réussi à géocoder l’adresse ?)
* En fonction de ce résultat, on fait au choix
  + Si adresse valide, on affiche les coordonnées et on met à jour le booléen pour sortir de la boucle
* Sinon, on affiche un petit message d’erreur

## Faire décoller le drone

Le drone décollera utilisant la fonction takeoff

print ("Taking off!")

vehicle.simple\_takeoff (





## Visiter les destinations

Voici le centre du programme ici nous allons utiliser des fonctions qui nous permettra d’aller au adresses (peut-être se poser et redécoller) et aller a d’autre destinations ou rentrer au point de départ. Ici les fonctions utilisées sont simples comme la fonction Goto ;

print ("Going towards first point ")

point1 = LocationGlobalRelative (-35.361354, 149.165218, 20)

vehicle.simple\_goto(point1)



Ici point2 = LocationGlobalRelative est mis en commentaire parceque point2 est déjà importé du code qu’on a écrit qui demande a l’utilisateur d’entrer ses addresses a visiter.

## Revenir se poser au point de départ

Ici la fonction return to launch est utilisé car les coordonnées du point de départ sont gardées.

print ("Returning to Launch")

vehicle.mode = VehicleMode("RTL").



## Fonctions secondaires

* Le drone pourra aller se placer sur un point de départ en utilisant la fonction init\_vehicle

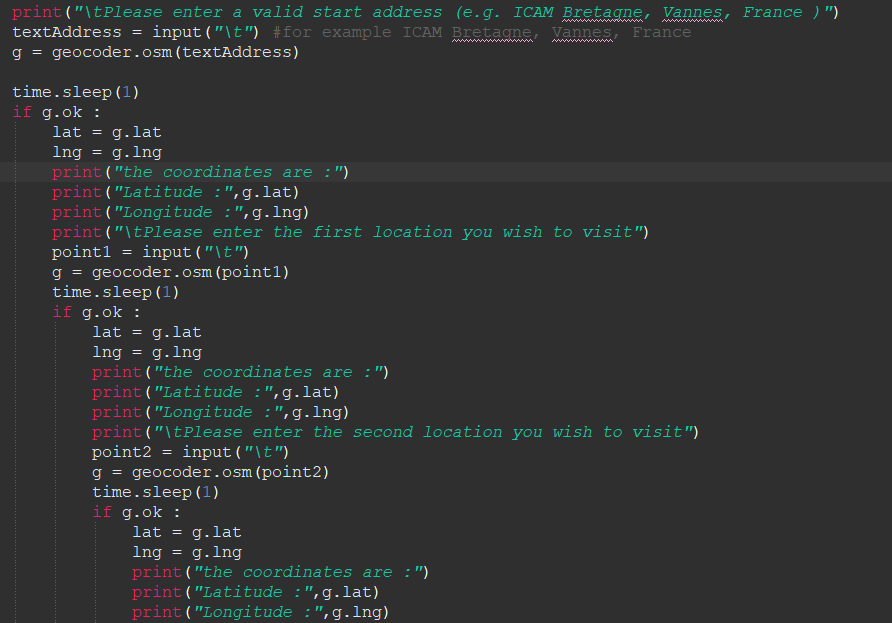
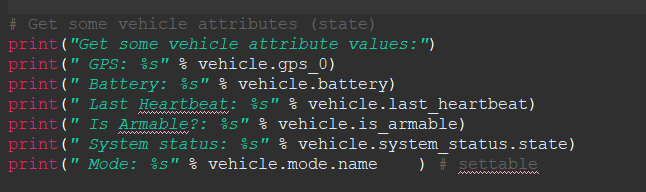
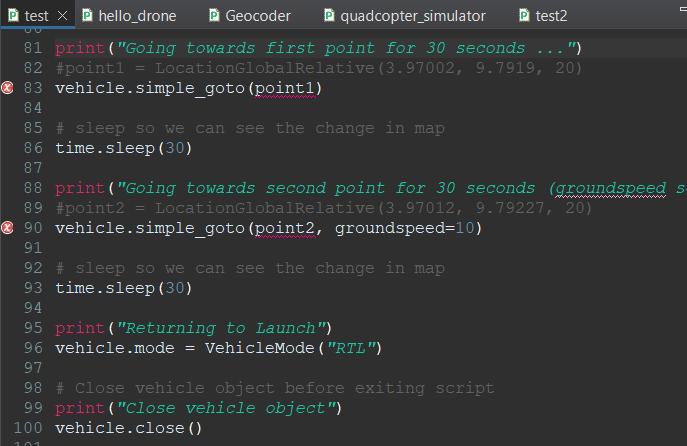
vehicle = init\_vehicle(47.6897121 , -2.7478361)

* Arrivé aux destinations, le drone pourra se poser pour un moment avant de continuer en utilisant la fonction time.sleep

time.sleep(30) .

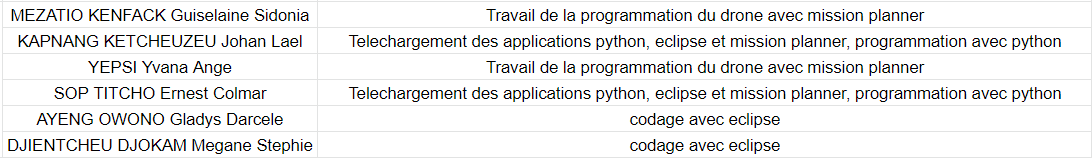
## SIMULATION

Nous allons simuler le vol d’un drone avec mission planner qui nous aidera dans la simulation. Tout les codes y compris ce qu’on a écrit sont importés dans Test.py (qui est le fichier ouvert dans Eclipse ou on a importé simple\_goto) et puis le code est lancer avec mission planner.

1. Tout d’abord le code importe hello\_drone qui dans elle-même est importé notre code (appelle Geocoder) qui demande a l’utilisateur d’entrer les addresses ; de départ, du premier point et du deuxième.
2. Le code hello\_drone en lui-même recupere des information de l’etat du drone. 
3. Ensuite le code simple\_goto envoi le drone vers le point1 et le point2 defini dans le code Geocoder et par la suite renvoi le drone au point de depart.

## DEROULEMENT

Pendant la première phase de travail sur ce domaine, le travaille a été reparti tel que ;



Durant la deuxième phase du travail il était de commenter les ressources (codes) qui nous ont été donnés afin de comprendre ce que les codes fessaient pour pouvoir écrire les notre. Ceci étant dit, les ressources ; simple\_goto, quadcopter\_simulator, hello\_drone, sur chaque ligne de code a été ajouté un commentaire.

## CONCLUSION

Ainsi donc notre drone est programmé pour visiter des destinations de façon autonome et avec les coordonnées de départ enregistré, le drone pourra il retourner en cas de besoin.