Compte rendu : Gridmission

I- Définition de la mission :

Une **mission de grid** est un type de mission de vol automatisé où un drone suit un trajet en **quadrillage** (ou grille) pour couvrir méthodiquement la zone définie. Ce type de mission est utilisé principalement pour la **cartographie**, la **surveillance** et l'**inspection**, permettant une collecte de données homogène et exhaustive. Les principales caractéristique de cette mission est de :

- **Trajectoire structurée** : Le drone suit un schéma en lignes parallèles (ou en zigzag) pour couvrir toute la surface ciblée.
- **Recouvrement optimisé** : Un taux de chevauchement entre les passages est défini pour assurer une couverture complète et précise.
- **Waypoints prédéfinis** : La mission repose sur une série de points de passage programmés.

Aussi cette mission peut être utiliser dans les applications suivantes :

- Cartographie aérienne et photogrammétrie
- Surveillance environnementale et inspection d'infrastructures
- Agriculture de précision (analyse des cultures)
- Recherche et sauvetage (scans de zones larges)

II- Les entrées et la sortie de la mission :

1- **Les entrées** : les entrées de ce code sont le **nombre d'extrémités** qui forment le polygone ainsi que les **coordonnées GPS** de ses points. Ensuite, l'utilisateur choisit la **moitié de la distance** entre chaque paire de points successifs du trajet de la grille, ainsi que **l'altitude du drone**.

2- **Les sorties** : La **première sortie** de ce code est un **algorithme** qui génère des points **ordonnés** (ou triés) formant le **trajet en grille**, puis crée une mission **waypoint** avec cette liste de points (de cette façon, on obtient la mission de grille). De plus, la **deuxième sortie** est le **calcul de la surface** de cette forme.

III- Explication du code de la mission :

Ce script Python permet d'automatiser une mission de **grille** pour un **drone Parrot**. Il commence par définir une zone de vol sous forme de **polygone GPS** (qui sont les entrées de ce code) dont les coordonnées sont validées. Ensuite, l'utilisateur spécifie la **moitié de la distance entre chaque point successif du trajet** ainsi que **l'altitude du drone**, qui doit être comprise entre **0 et 100 mètres**. Une fonction génère ensuite une **grille de points ordonnés**, représentant le trajet optimal du drone à l'intérieur du polygone, en s'assurant que tous les points restent dans la zone définie. En parallèle, une fonction de **streaming vidéo** affiche le flux en temps réel via **RTSP**. Le programme établit la connexion avec le drone, active **l'évitement d'obstacles** et le fait **décoller**. Ensuite, il suit

le **trajet en grille** en utilisant la commande `moveTo()`, avec des pauses pour garantir la stabilité. Une fois tous les points atteints, le drone **atterrit** automatiquement et se **déconnecte**. Le script calcule également la **surface couverte** en fonction du nombre de points du trajet.

IV- Explication des fonctions du script :

start_video_stream() : Démarre un flux vidéo RTSP en temps réel depuis le drone et l'affiche en redimensionné à l'écran.

validate_gps_coordinates(polygon_points) : Vérifie que les coordonnées GPS fournies sont valides (latitude entre -90 et 90, longitude entre -180 et 180).

divide_into_circles(polygon_points, circle_radius_m) : Divise une surface polygonale en une grille de cercles en fonction du rayon donné, en s'assurant qu'ils sont bien dans le polygone.

count_circles_in_polygon(polygon_points, circle_radius_m) : Compte le nombre de cercles qui peuvent être placés à l'intérieur du polygone.

sort_circle_centers(circle_centers) : Trie les centres des cercles en une trajectoire optimisée en zigzag pour un parcours plus efficace.

read_polygon() : Retourne la liste des points du polygone définissant la zone de vol.