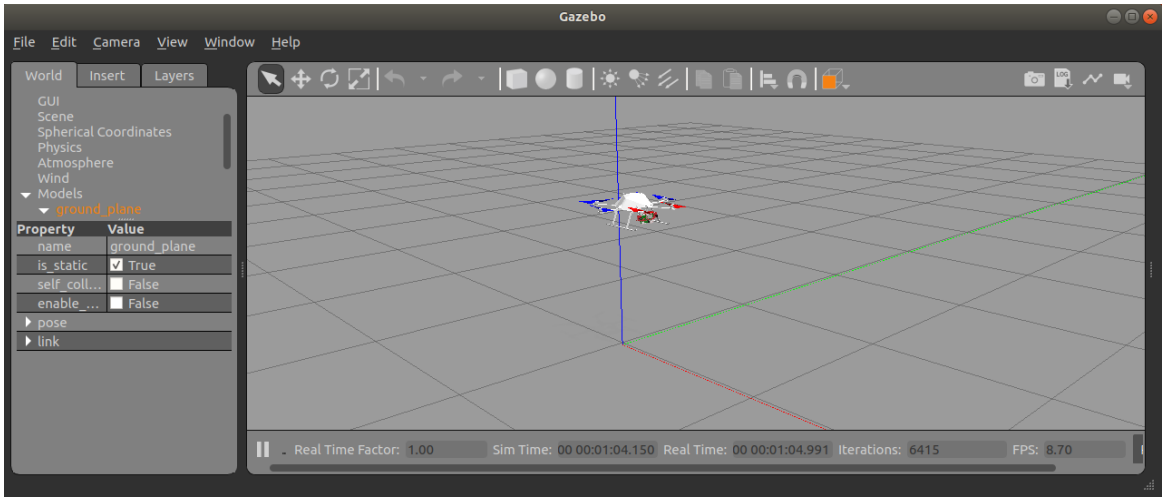
Commande d'un drone



Dans le dossier **scripts** du package **drone_control** fourni se trouve un programme où sont déjà présentes :

- La souscription (entrée) à l'odométrie **/firefly/odometry_sensor1/odometry** qui permet d'accéder à la position, la vitesse et l'orientation (sous forme de quaternion) du drone.
- La publication (sortie) de la commande sur **/firefly/roll_pitch_yawrate_thrust** en angles de roulis (roll) et tangage (pitch), vitesse de rotation en lacet (yaw_rate) et poussée (thrust). Ce message est spécifique au simulateur utilisé (rotors), de type **mav_msgs/RollPitchYawrateThrust.**

1. Tester le fichier **drone_control.launch** présent dans le dossier **launch** du package **drone_control**, qui permet de lancer le simulateur du drone ainsi que le noeud drone_control.py.
2. Dans le callback de l'odométrie, récupérer la position, la vitesse du drone, ainsi que l'angle de lacet (à partir du quaternion).

Dans le cas considéré ici, le modèle dynamique du drone peut se résumer à un double intégrateur :

$$\begin{cases} \dot{p} = v \\ \dot{v} = u \end{cases} \qquad u = \frac{\mathcal{T}}{m} R e_3 - g e_3 \tag{1}$$

avec et où $p = [p_x, p_y, p_z]^T$ est la position et $v = [v_x, v_y, v_z]^T$ la vitesse en repère inertiel, $\mathcal{T}$ la poussée, m la masse du drone, R la matrice de rotation permettant de passer du repère inertiel au repère lié au drone, e_3 l'axe vertical du repère inertiel (celui sur lequel la gravité s'applique).
En supposant les angles de roulis ϕ et tangage θ comme petits (ce qui implique pour un angle α : $\cos\alpha \approx 1$ et $\sin\alpha \approx \alpha$), le produit Re_3 se simplifie alors comme :

$$Re_3 \approx \begin{bmatrix} \theta \cdot \cos\psi + \phi \cdot \sin\psi \\ \theta \cdot \sin\psi - \phi \cdot \cos\psi \\ 1 \end{bmatrix} \tag{2}$$

Afin de réaliser un premier contrôleur, on conservera nulle la vitesse de rotation en lacet (yaw rate), ce qui a pour effet de conserver l'angle ψ nul, le produit Re_3 se simplifie alors encore comme :

$$Re_3(\psi = 0) \approx \begin{bmatrix} \theta \\ -\phi \\ 1 \end{bmatrix} \tag{3}$$

ce qui nous donne :

$$u = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \theta \frac{\mathcal{T}}{m} \\ -\phi \frac{\mathcal{T}}{m} \\ \frac{\mathcal{T}}{m} - g \end{bmatrix} \tag{4}$$

3. Concevoir un correcteur pour chacune des composantes de translation x, y, z à partir du modèle dynamique (1), dans le cas où l'on conserve $\psi = 0$. Celui-ci permet d'obtenir les composantes u_x, u_y, u_z et il faut alors d'utiliser l'équation (4) pour retrouver $\mathcal{T}$ puis θ et ϕ.
4. Le modèle dynamique en lacet (yaw) est simplement $\dot{\psi} = u_\psi$ où u_ψ est le « yaw rate » attendu dans le message publié. Proposer un correcteur adapté pour l'angle de lacet.
   Attention, lorsque l'angle de lacet varie, les composantes x et y de la vitesse mesurée (qui est en repère engin) récupérée dans le callback de l'odométrie doit être transformée en repère inertiel via la matrice de rotation en lacet
   $$C_\psi = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}$$
   Avec l'hypothèse petits angles, la composante en z est inchangée.
   Le correcteur en position défini précédemment reste applicable, par contre il faut maintenant utiliser la relation (2) pour retrouver $\mathcal{T}$ puis θ et ϕ.
   Pour toutes ces transformations, utiliser la valeur mesurée de l'angle ψ (récupérée dans le callback odométrie).