Création d'un Airframe pour PX4

Iwan Sidaropoulos

Résumé

Ce document est destiné aux développeurs souhaitant adapter PX4 à des plateformes personnalisées, il détaille le processus de création et d'intégration d'un airframe pour PX4, une plateforme open-source dédiée aux drones et véhicules autonomes. Il explore l'architecture modulaire de PX4, les différents modes de simulation, et les méthodes de création et d'intégration d'airframe au sein de PX4.

Table des matières

1 Présentation de PX4

PX4 est une plateforme open-source de pilotage automatique conçue pour les drones et autres véhicules autonomes. Développé à l'origine à l'ETH Zurich, il est aujourd'hui maintenu par la fondation Dronecode et une large communauté de chercheurs, ingénieurs et passionnés. Il fournit une solution complète pour la navigation, le contrôle de vol et la gestion des capteurs.

L'architecture de PX4 est modulaire, permettant une grande flexibilité dans l'intégration de nouveaux capteurs, contrôleurs et algorithmes de vol. Il repose sur un firmware optimisé fonctionnant sur diverses plateformes matérielles comme les cartes Pixhawk et d'autres systèmes embarqués. Il utilise le protocole MAVLink pour la communication entre le contrôleur de vol, les stations au sol et d'autres périphériques.

PX4 prend en charge plusieurs modes de vol, allant du pilotage manuel à l'automatisation complète via des missions programmées. Il s'intègre également avec des simulateurs comme Gazebo et jMAVSim, facilitant ainsi le développement et le test des algorithmes avant leur déploiement sur un appareil réel.

Grâce à sa flexibilité et à son ouverture, PX4 est utilisé dans de nombreux domaines, allant des applications industrielles aux projets de recherche et aux drones de loisir. Son écosystème dynamique, soutenu par une large communauté, en fait une référence majeure dans le domaine des systèmes autonomes.

2 Architecture PX4

Avant d'aborder la création d'un airframe, il est essentiel de comprendre l'architecture de PX4. Cette compréhension permet d'identifier où et comment un airframe s'intègre dans l'ensemble du système.

2.1 Architecture système

PX4 repose sur une architecture modulaire qui peut être configurée de différentes manières selon les besoins du système. Les deux architectures principales utilisées sont :

— Une architecture basée uniquement sur un **Flight Controller**, qui gère l'ensemble des calculs de vol et l'interface avec les capteurs et actionneurs.

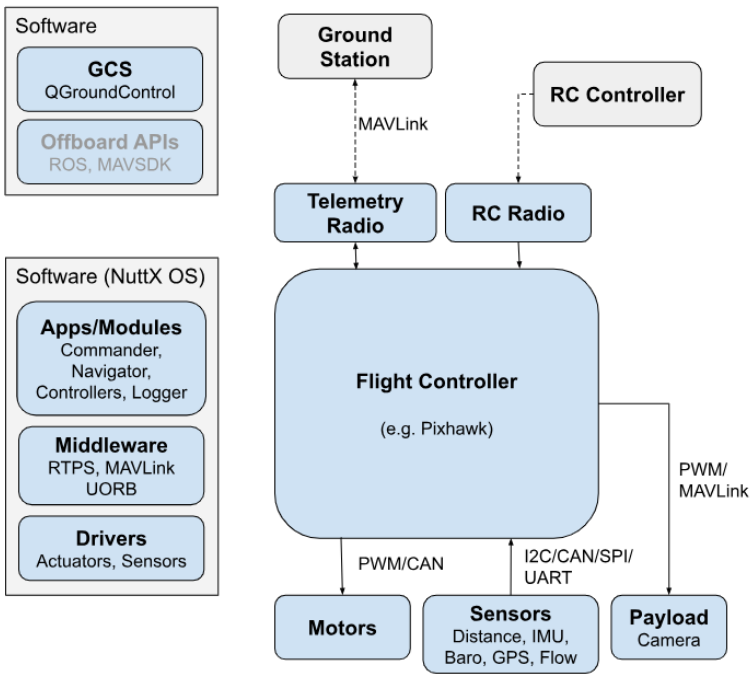

FIGURE 1 – Architecture système avec uniquement un Flight Controller.

— Une architecture combinant un **Flight Controller** et un **Companion Computer**, où ce dernier apporte des capacités de calcul supplémentaires, principalement pour des applications de vision par ordinateur.

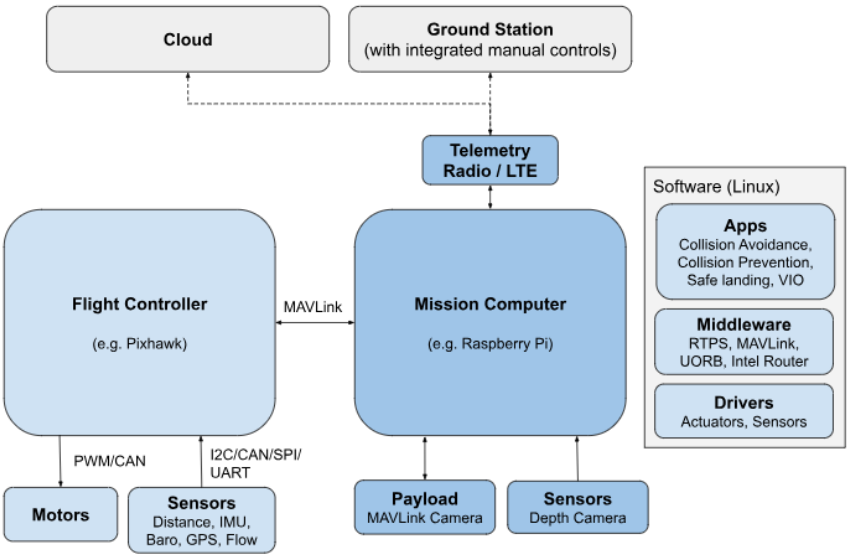

FIGURE 2 – Architecture système avec Flight Controller et Companion Computer.

Les principales composantes visibles dans ces architectures sont :

— **Station au sol (GCS)** : Interface de contrôle du drone, permettant la supervision, la planification de missions et le contrôle manuel via une liaison radio de télémétrie. Un exemple de GCS largement utilisé est **QGroundControl**.

— **Cloud** : Dans certaines configurations, le drone peut être connecté au cloud via le Companion Computer pour l'enregistrement de données, la gestion à distance ou des traitements avancés.

— **Télémétrie et communication** : Les communications entre le drone et la station au sol sont assurées via un lien radio. Le Companion Computer peut également gérer les communications réseau et router les messages MAVLink vers la station au sol ou le cloud.

— **Capteurs** : Incluent GPS, compas, baromètres, lidars, caméras, capteurs de flux optique et autres dispositifs permettant la navigation et l'évitement d'obstacles.

— **Actionneurs** : Englobent les moteurs (via ESCs) et autres dispositifs permettant la stabilisation et le mouvement du drone.

— **Software** : Le Flight Controller exécute le **PX4 flight stack**, comprenant les drivers, les modules de communication, les contrôleurs et les estimateurs d'état. Sur un Companion Computer, un système Linux peut exécuter des logiciels additionnels (ex. MAVSDK, ROS, MAVLink Router) pour gérer des fonctionnalités avancées.

2.2 Architecture Logicielle

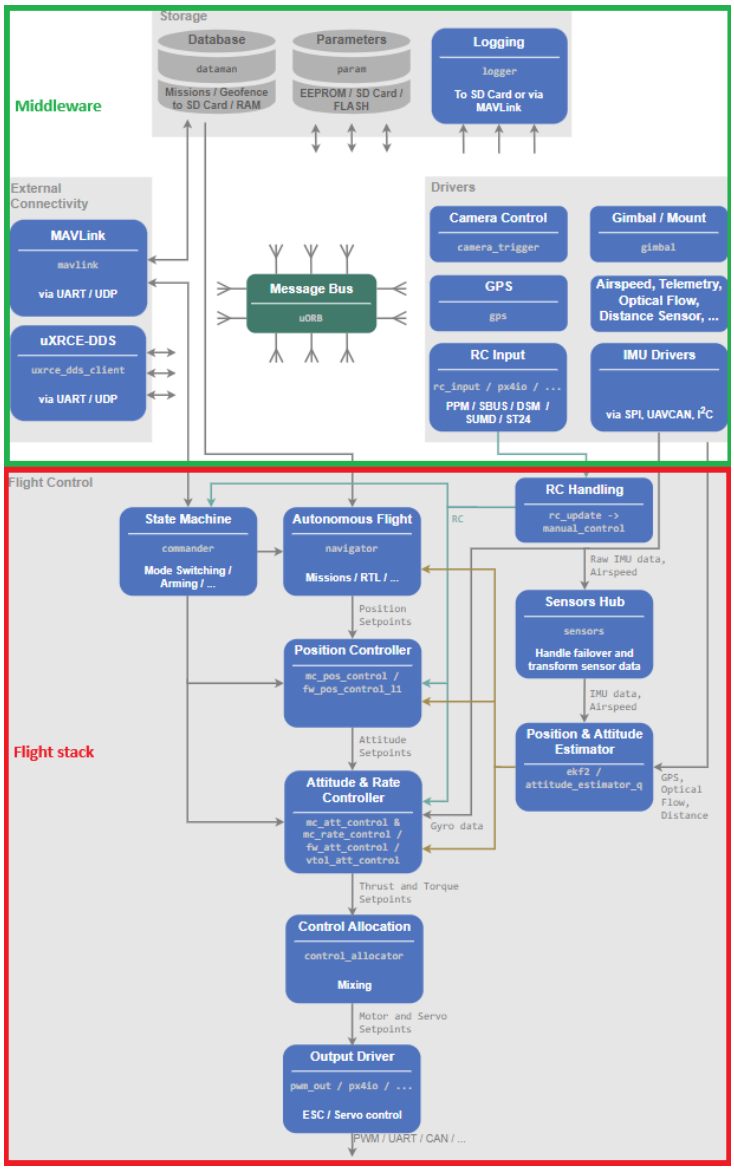

FIGURE 3 – Architecture logicielle de PX4.

PX4 est conçu avec une architecture logicielle modulaire et extensible qui permet de gérer les différents composants du système de vol de manière efficace. Cette architecture repose sur un bus de messages centralisé qui facilite la communication entre les divers modules du firmware. Grâce à cette organisation, il est possible d'intégrer facilement de nouveaux capteurs, algorithmes de contrôle ou interfaces de communication.

2.2.1 Message Bus (uORB)

Le bus de messages uORB est le cœur du système PX4. Il permet aux différents modules de communiquer en publiant et en s'abonnant à des messages spécifiques. Ce mécanisme de messagerie assure un découplage entre les composants, rendant le système plus modulaire et évolutif.

2.2.2 Storage

Le stockage des données dans PX4 est géré par :

— Une base de données qui enregistre les missions et les zones de géorepérage.
— Un système de paramètres stocké en EEPROM, sur carte SD ou en mémoire flash.
— Un module de journalisation (Logging) permettant d'enregistrer des données en vol pour l'analyse et le diagnostic.

2.2.3 Drivers

Les drivers assurent l'interface entre PX4 et le matériel (capteurs et actionneurs). Parmi les principaux drivers :

— Contrôle caméra : Gestion des déclencheurs pour la capture d'images.
— GPS : Acquisition des données de positionnement.
— RC Input : Lecture des commandes envoyées par une radiocommande.
— IMU Drivers : Interface avec les capteurs inertiels (gyroscopes, accéléromètres).

2.2.4 Flight Control

Le contrôle de vol dans PX4 repose sur plusieurs modules qui assurent la gestion de la navigation, du positionnement et du contrôle d'attitude. La figure 4 illustre l'organisation générale de cette *stack* de contrôle.

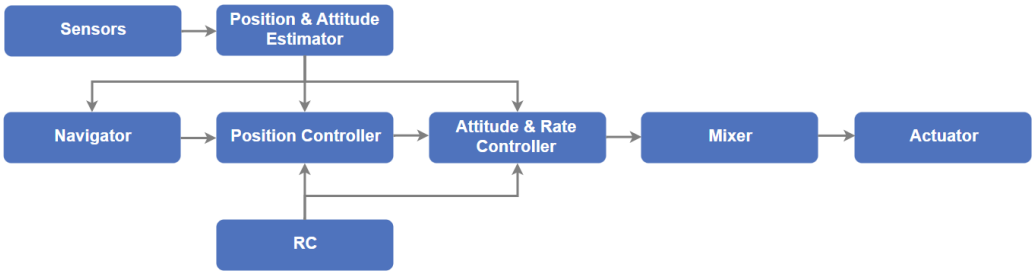

FIGURE 4 – Flight stack.

Les principaux composants sont :

— ***Navigator*** : Responsable des missions autonomes et de la planification de trajectoire. Lorsqu'un vol autonome est activé, ce module génère des *position setpoints*, qui correspondent aux points à atteindre dans l'espace (latitude, longitude, altitude). En mode manuel, ces *setpoints* sont plutôt déterminés par les commandes de l'utilisateur via la radiocommande.

— ***Position & Attitude Estimator*** : Fusionne les données des capteurs pour estimer en continu la position, la vitesse et l'attitude du drone. Ces informations sont essentielles pour que les contrôleurs puissent prendre des décisions précises.

— ***Position Controller*** : Transforme les *position setpoints* en *attitude setpoints*. Concrètement, il détermine l'angle d'inclinaison (roulis, tangage) et l'orientation (lacet) nécessaires pour atteindre la position cible, tout en générant une commande de poussée verticale.

— ***Attitude Controller*** : Convertit les *attitude setpoints* en *angular velocity setpoints* (vitesses angulaires). Il calcule les vitesses angulaires idéales autour des axes de roulis, tangage et lacet pour atteindre l'attitude désirée, en tenant compte des dynamiques du drone et des éventuelles compensations nécessaires.

— ***Rate Controller*** : Prend en entrée les *angular velocity setpoints* fournis par l'*Attitude Controller* et génère les commandes de *thrust et torque* nécessaires pour ajuster dynamiquement la vitesse angulaire du drone. Il stabilise l'appareil en appliquant les couples correctifs à chaque instant.

— ***RC Handling*** : Gère les commandes envoyées par l'utilisateur via la radiocommande. En mode manuel, ces commandes influencent directement les *setpoints* du *Position Controller* ou de l'*Attitude Controller* selon le mode de vol actif.

Actuator Control Pipeline

Le *Rate Controller*, le *Control Allocation*, ainsi que les *Output Drivers* forment l'*Actuator Control Pipeline*, qui est responsable de la conversion des commandes en signaux moteurs et actionneurs. La figure 5 illustre cette chaîne de traitement.

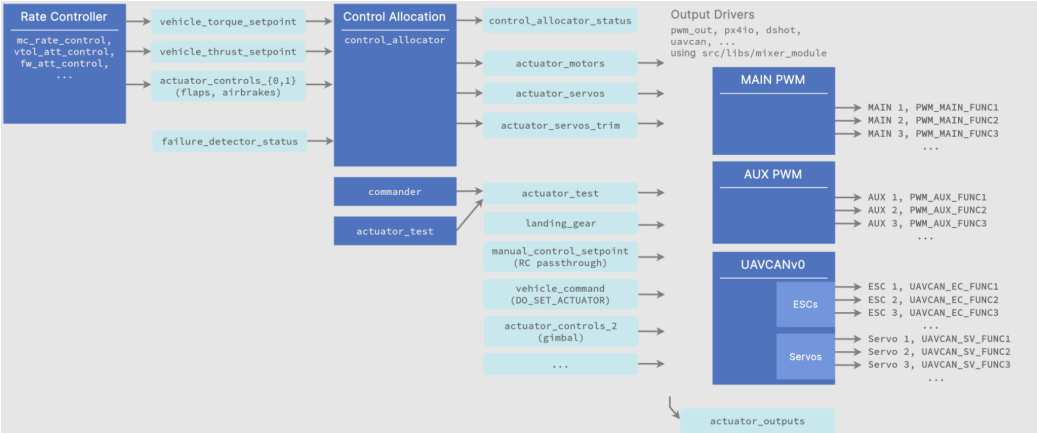

FIGURE 5 – Actuator Control Pipeline.

— ***Rate Controller*** : Stabilise l'appareil en ajustant la vitesse de rotation autour des axes de roulis, tangage et lacet. Il reçoit en entrée les *angular velocity setpoints* et génère les commandes de *thrust et torque*, qui seront ensuite traitées par le *Control Allocation*.
— ***Control Allocation*** (ou ***Mixing***) : Répartit les commandes de *thrust* et *torque* entre les différents actionneurs en fonction de la configuration de l'appareil (*quadricoptère*, *avion*, *VTOL*, etc.).
— ***Output Drivers*** : Génèrent les signaux finaux destinés aux moteurs et actionneurs via différents protocoles (*PWM*, *DShot*, *UAVCAN*, etc.).

On remarquera que certains modules sont implémentés sous plusieurs programmes distincts en fonction du type d'appareil. Par exemple, le contrôleur d'attitude existe sous différentes variantes : `airship_att_control`, `fw_att_control`, `mc_att_control` ou encore `vtol_att_control`, respectivement dédiés aux dirigeables, avions à voilure fixe, multicoptères et appareils hybrides VTOL. À l'inverse, d'autres modules sont implémentés sous un seul programme modulable selon la configuration du véhicule. C'est le cas de `control_allocator`, qui ajuste dynamiquement la répartition des commandes en fonction du type d'appareil et de ses actionneurs disponibles.

2.2.5 External Connectivity

PX4 utilise plusieurs protocoles pour communiquer avec des systèmes externes.

— **uXRCE-DDS** : Protocole optimisé permettant la communication

avec des architectures distribuées, notamment dans des contextes robotiques et multi-drones.

— **MAVLink** : Protocole de communication principal utilisé par PX4 pour échanger des données avec les stations au sol et d'autres systèmes via *UART* ou *UDP*. Il est au cœur des interactions avec **QGroundControl** ainsi que la majorité des simulateurs.

QGroundControl

QGroundControl (QGC) est la station au sol principale utilisée avec PX4. Elle permet de surveiller l'état du drone, d'envoyer des commandes, de planifier des missions et d'accéder aux logs de vol.

Les principales fonctionnalités de QGroundControl incluent :

— La configuration et le calibrage des capteurs et actionneurs.
— La planification et le suivi des missions en temps réel.
— La visualisation des données de télémétrie.
— L'affichage des alertes et statuts critiques du drone.

3 Simulation

3.1 Simulateurs

PX4 est compatible avec plusieurs simulateurs permettant de tester les algorithmes de vol avant leur déploiement sur un appareil réel. Ces simulateurs diffèrent en termes de précision, d'interfaces et de cas d'usage. La majorité utilise le protocole **MAVLink** pour échanger des données avec PX4.

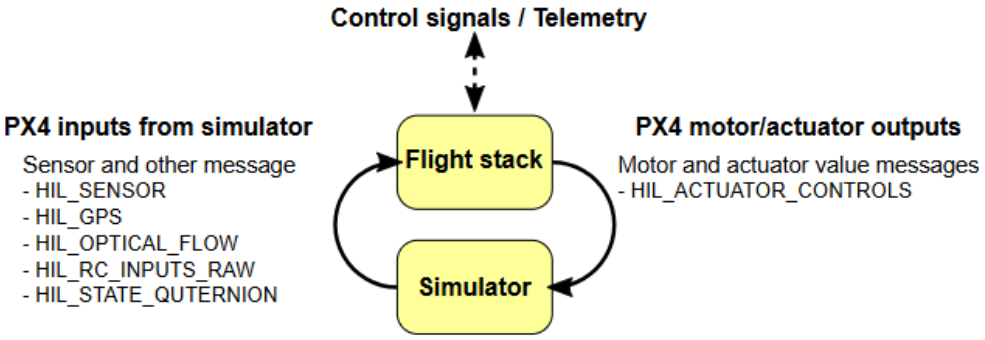

FIGURE 6 – API MAVLink.

Parmi ces simulateurs, le plus couramment utilisé est **jMAVSim**. Il s'agit d'un simulateur léger et rapide, principalement conçu pour tester la stabilité de vol des multicoptères. Grâce à son intégration native avec **MAVLink**, il permet une interaction fluide avec PX4.

Le seul simulateur qui ne repose pas sur **MAVLink** pour communiquer avec PX4 est **Gazebo**, anciennement **Gazebo Classic**. Ce simulateur 3D avancé offre une modélisation physique plus réaliste et est particulièrement adapté aux tests impliquant des interactions avec l'environnement et l'utilisation de capteurs complexes. Contrairement aux autres simulateurs, Gazebo utilise un plugin spécifique pour échanger des données avec PX4.

FIGURE 7 – Simulation sur Gazebo.

3.2 Software In The Loop (SITL)

Le mode *Software-In-The-Loop* (SITL) permet d'exécuter PX4 sur un ordinateur sans nécessiter de matériel physique. Il est utilisé pour tester les algorithmes de contrôle et valider le comportement du système avant de passer à une implémentation réelle.

Dans ce mode, le firmware PX4 tourne en simulation et communique avec un simulateur externe qui modélise la dynamique du véhicule. La communication entre PX4 et le simulateur se fait généralement via le protocole **MAVLink**, permettant d'interagir avec le drone simulé comme s'il s'agissait d'un appareil réel.

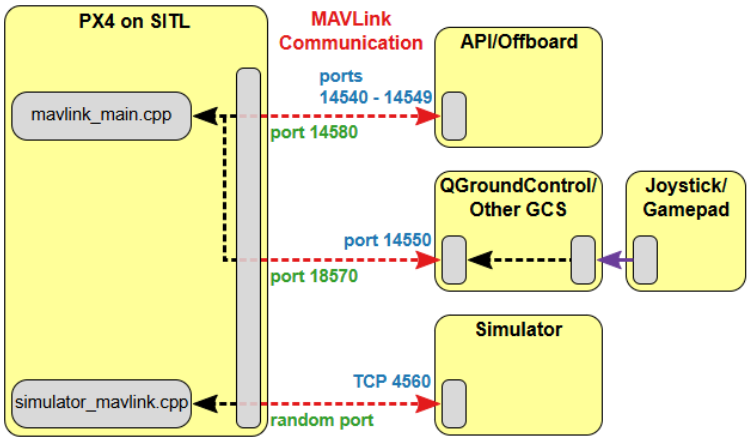

FIGURE 8 – Environnement de simulation SITL (pour simulateur utilisant MAVLink).

PX4 propose une large gamme de modèles de véhicules pouvant être simulés en SITL. Ces modèles sont définis dans des fichiers de configuration situés dans le répertoire :

`PX4-Autopilot/ROMFS/px4fmu_common/init.d-posix/airframes`

Chaque fichier de ce dossier correspond à un modèle spécifique et est conçu pour un simulateur donné. Par exemple, on y trouve des configurations adaptées à **Gazebo Classic**, **jMAVSim**, **SITL-In-Hardware (SIH)**, **JSBSim** ou encore **FlightGear**.

Les fichiers suivent une convention de nommage qui inclut un identifiant unique et le simulateur cible. Par exemple :

— `10017_jmavsim_iris` : Modèle Iris pour le simulateur jMAVSim.
— `10015_gazebo-classic_iris` : Modèle Iris pour Gazebo Classic.

— `1033_jsbsim_rascal` : Modèle Rascal pour JSBSim.
— `1040_gazebo-classic_standard_vtol` : Modèle VTOL standard pour Gazebo Classic.
— `4001_gz_x500` : Modèle X500 pour le simulateur Gazebo (Ignition/GZ).

Ces fichiers contiennent les paramètres de configuration nécessaires à l'initialisation du modèle dans l'environnement de simulation, notamment les réglages des capteurs, les constantes physiques et les paramètres du contrôleur de vol.

3.3 Hardware In The Loop (HITL)

Le mode *Hardware-In-The-Loop* (HITL) permet d'exécuter le firmware PX4 sur un contrôleur de vol réel tout en utilisant un simulateur externe pour modéliser l'environnement et la dynamique du véhicule. Cette approche permet de tester le comportement du logiciel en conditions proches du réel, sans risquer d'endommager un drone physique.

Dans ce mode, le contrôleur de vol est connecté à un simulateur tel que **jMAVSim** ou **Gazebo Classic**, qui émule les capteurs et l'environnement du drone. La communication entre le contrôleur de vol et le simulateur se fait via le protocole **MAVLink**, permettant d'interagir avec le drone simulé comme s'il s'agissait d'un appareil réel.

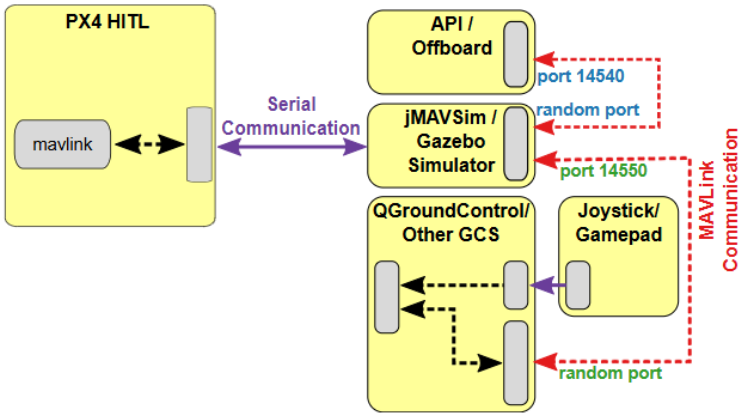

FIGURE 9 – Environnement de simulation HITL.

Les configurations des modèles compatibles avec le mode HITL sont situées dans le répertoire :

`PX4-Autopilot/ROMFS/px4fmu_common/init.d/airframes`

Actuellement, deux fichiers sont spécifiquement dédiés à la simulation HITL :

— `1001_rc_quad_x.hil` : Modèle quadricoptère compatible avec **jMAVSim** et **Gazebo Classic**.

— `1002_standard_vtol.hil` : Modèle VTOL standard compatible avec **Gazebo Classic**.

Ces fichiers contiennent les paramètres nécessaires à l'initialisation du modèle en HITL, notamment la configuration des capteurs, les constantes physiques et les paramètres du contrôleur de vol.

3.4 Simulation In Hardware (SIH)

Le mode *Simulation-In-Hardware* (SIH) est une alternative au *Hardware-In-The-Loop*. Contrairement au HITL, où un simulateur externe modélise la dynamique du véhicule, le SIH exécute cette simulation en interne. L'ordinateur de bureau est uniquement utilisé pour visualiser le véhicule en mouvement via **jMAVSim**.

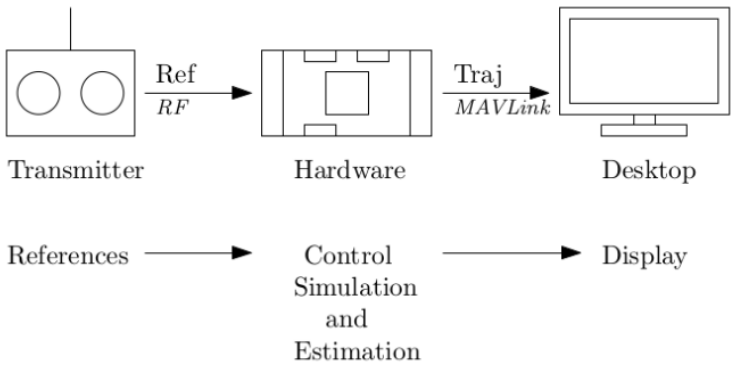

FIGURE 10 – Environnement de simulation SIH.

Par rapport au HITL, le SIH offre plusieurs avantages : il élimine les latences dues à la communication avec un simulateur externe, simplifie la configuration en supprimant la dépendance à un logiciel tiers et permet un contrôle direct du véhicule via une radiocommande.

Les fichiers de configuration SIH sont situés dans :

`PX4-Autopilot/ROMFS/px4fmu_common/init.d/airframes`

Chaque fichier correspond à un modèle spécifique et contient les paramètres nécessaires à son initialisation :

— `1100_rc_quad_x_sih.hil` : Modèle quadrirotor SIH.
— `1101_rc_plane_sih.hil` : Modèle avion à voilure fixe SIH.
— `1102_tailsitter_duo_sih.hil` : Modèle tailsitter SIH.
— `1103_standard_vtol_sih.hil` : Modèle VTOL standard SIH.

SIH as SITL est une variante de la simulation SIH qui permet d'exécuter PX4 en mode *Software-In-The-Loop* (SITL) tout en utilisant le moteur de simulation interne du SIH. Contrairement au mode SIH classique, qui tourne sur un contrôleur de vol réel, *SIH as SITL* s'exécute entièrement sur un ordinateur, comme une simulation SITL standard.

Comme en mode SIH classique, il est également possible de visualiser le véhicule dans un environnement graphique, comme **jMAVSim**, ce qui permet de suivre les mouvements du véhicule en temps réel pendant la simulation.

Les fichiers de configuration pour *SIH as SITL* sont stockés dans :

```
PX4-Autopilot/ROMFS/px4fmu_common/init.d-posix/airframes
```

Les modèles compatibles sont :
— `10040_sihsim_quadx` : Modèle quadricoptère.
— `10041_sihsim_airplane` : Modèle avion.
— `10042_sihsim_xvert` : Modèle tailsitter.
— `10043_sihsim_standard_vtol` : Modèle VTOL standard.

4 Airframes

4.1 Définition et exemples

Un **airframe** dans PX4 représente une configuration matérielle spécifique, définissant la disposition des moteurs et servos, ainsi que les paramètres de vol associés. Il permet d'adapter le firmware PX4 à un type précis de véhicule et configure automatiquement les modules essentiels du système, notamment les contrôleurs de vol, le mixage des actuateurs et l'estimation d'état.

PX4 prend en charge une grande variété d'airframes, notamment les drones multirotors, les avions, les VTOL (Vertical Take-Off and Landing), les rovers et d'autres plateformes spécialisées. L'interface graphique de QGroundControl permet de sélectionner facilement un airframe parmi les modèles supportés, comme illustré en figure 11.

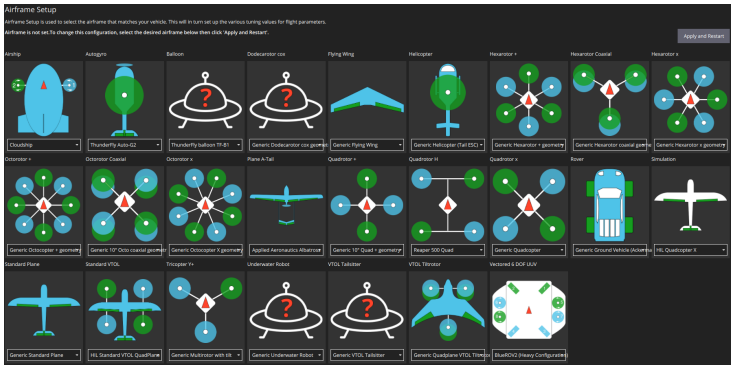

FIGURE 11 – Sélection des airframes dans QGroundControl

4.2 Composition d'un airframe

Dans la plupart des cas, un airframe est simplement représenté par un fichier de configuration placé dans :

— `PX4-Autopilot/ROMFS/px4fmu_common/init.d/airframes` pour les airframes standards.

— `PX4-Autopilot/ROMFS/px4fmu_common/init.d-posix/airframes` pour les airframes destinés à la SITL.

Le nom du fichier suit la convention `${ID}_${model_name}`, où `ID` est le `SYS_AUTOSTART_ID` utilisé pour sélectionner l'airframe, et `model_name` est le nom du modèle de l'airframe.

Prenons l'exemple de l'airframe `3000_generic_wing`, qui définit une aile volante générique :

```
#!/bin/sh
#
# @name Generic Flying Wing
#
# @type Flying Wing
# @class Plane
#
# @board bitcraze_crazyflie exclude
#

. ${R}etc/init.d/rc.fw_defaults

param set-default CA_AIRFRAME 1

param set-default CA_ROTOR_COUNT 1
param set-default CA_ROTOR0_PX 0.15
param set-default CA_SV_CS_COUNT 2
param set-default CA_SV_CS0_TYPE 5
param set-default CA_SV_CS0_TRQ_P 0.5
param set-default CA_SV_CS0_TRQ_R -0.5
param set-default CA_SV_CS1_TYPE 6
param set-default CA_SV_CS1_TRQ_P 0.5
param set-default CA_SV_CS1_TRQ_R 0.5
```

Listing 1 – Fichier 3000_generic_wing

L'airframe commence par l'instruction suivante :

```
. ${R}etc/init.d/rc.fw_defaults
```

Cette ligne permet d'initialiser la flight stack de PX4 correspondant à la classe des *flying wing* (avions), ce qui charge les modules essentiels nécessaires à un aéronef de type avion.

Ce fichier configure les paramètres initiaux, tels que :

— `CA_AIRFRAME` : Identifiant de l'airframe.
— `CA_ROTOR_COUNT` : Nombre de rotors.
— `CA_SV_CS_COUNT` : Nombre de servos et leur configuration.

Une description complète des paramètres est disponible sur la page officielle de la documentation PX4 ici.

4.3 Spécificités pour la simulation

Les fichiers d'airframes dédiés à la simulation incluent généralement des paramètres supplémentaires liés à l'environnement simulé. Leur contenu varie selon le simulateur utilisé.

4.3.1 SITL

Les fichiers SITL contiennent souvent plus de code pour gérer les interactions avec le simulateur. Exemple : `4008_gz_advanced_plane`.

```
PX4_SIMULATOR=${PX4_SIMULATOR:=gz}
PX4_GZ_WORLD=${PX4_GZ_WORLD:=default}
PX4_SIM_MODEL=${PX4_SIM_MODEL:=advanced_plane}

param set-default SIM_GZ_EN 1
param set-default SENS_EN_GPSSIM 1
param set-default SENS_EN_MAGSIM 1
param set-default SIM_GZ_EC_FUNC1 101
```

Listing 2 – Extrait du fichier 4008_gz_advanced_plane

Les lignes spécifiques à la simulation sont :
— `SIM_GZ_EN` : Active la simulation Gazebo.
— `SENS_EN_GPSSIM` : Active le GPS simulé.

4.3.2 HITL

Les fichiers HITL contiennent des paramètres particuliers, comme illustré avec `1001_rc_quad_x.hil` :

```
param set SYS_HITL 1
param set UAVCAN_ENABLE 0
param set-default CBRK_SUPPLY_CHK 894281
param set-default HIL_ACT_FUNC1 101
```

Listing 3 – Extrait du fichier 1001_rc_quad_x.hil

Les lignes spécifiques à HITL sont :
— `SYS_HITL` : Active le mode HITL.
— `HIL_ACT_FUNC1` : Définit des fonctions spécifiques à la simulation HITL.

4.3.3 SIH

Les fichiers SIH, comme `1101_rc_plane_sih.hil`, incluent des paramètres spécifiques :

```
param set SYS_HITL 2
param set SIH_T_MAX 6
param set SIH_MASS 0.3
param set SIH_IXX 0.00402
```

Listing 4 – Extrait du fichier 1101_rc_plane_sih.hil

Les paramètres propres à SIH sont :

— `SYS_HITL=2` : Mode SIH activé.
— `SIH_MASS`, `SIH_IXX` : Caractéristiques physiques du modèle.

4.3.4 SIH as SITL

Les fichiers "SIH as SITL", comme `10041_sihsim_airplane`, incluent des paramètres spécifiques :

```
PX4_SIMULATOR=${PX4_SIMULATOR:=sihsim}
PX4_SIM_MODEL=${PX4_SIM_MODEL:=airplane}
param set-default SENS_EN_GPSSIM 1
```

Listing 5 – Extrait du fichier 10041_sihsim_airplane

5 Intégration d'un Nouvel Airframe

L'ajout d'un nouvel *airframe* dans PX4 implique plusieurs étapes, notamment la création d'un fichier de configuration, son intégration dans QGroundControl, ainsi que la possibilité de le simuler sous Gazebo. Cette section détaille ces différentes étapes.

5.1 Intégration du fichier de configuration

L'intégration d'un nouvel *airframe* commence par la création d'un fichier de configuration dans l'un des répertoires suivants :

- `PX4-Autopilot/ROMFS/px4fmu_common/init.d/airframes` pour un *airframe* destiné au matériel réel (*HITL* ou vol réel).
- `PX4-Autopilot/ROMFS/px4fmu_common/init.d-posix/airframes` si l'*airframe* est destiné à une simulation *SITL* ou *SIH as SITL*.

Le fichier doit suivre la convention de nommage :

`[ID]_[nom].hil` ou `[ID]_[nom]`

où `[ID]` est un identifiant numérique unique qui ne doit pas déjà être utilisé dans les fichiers existants.

Une fois le fichier créé, il doit être déclaré dans le fichier CMake approprié :

- `PX4-Autopilot/ROMFS/px4fmu_common/init.d/airframes/CMakeLists.txt` pour une utilisation sur matériel réel.
- `PX4-Autopilot/ROMFS/px4fmu_common/init.d-posix/airframes/CMakeLists.txt` pour une utilisation en *SITL* ou *SIH as SITL*.

Il suffit d'ajouter le nom du fichier dans la liste des fichiers inclus dans le projet.

5.2 Création d'un nouveau groupe d'airframes dans QGroundControl

Si le nouvel *airframe* appartient à une nouvelle catégorie, il est nécessaire de créer un groupe correspondant dans **QGroundControl**. Les étapes sont les suivantes :

1. Ajouter une icône au format `.svg` dans le répertoire :

   `PX4-user_guide/assets/airframes/types`

2. Modifier le fichier `PX4-Autopilot/Tools/px4airframes/srcparser.py` en ajoutant dans la méthode `GetImageName(self)` la ligne suivante :

```
elif (self.name == "[Nom]"):
    return "[Nom]"
```

3. Pour mettre à jour QGroundControl, ajouter l'image `.svg` dans :

   `qgroundcontrol/src/AutoPilotPlugins/Common/Images`

4. Modifier le fichier de ressources `qgroundcontrol/qgcimages.qrc` et ajouter la ligne suivante :

```
<file alias="Airframe/[nom de l'image].svg">
    src/AutoPilotPlugins/Common/Images/[nom de l'image].svg
</file>
```

5.3 Intégration d'un modèle Gazebo

Pour simuler l'*airframe* dans Gazebo, un modèle 3D doit être ajouté dans PX4. La structure du modèle doit respecter la hiérarchie suivante :

- Un dossier portant le nom du véhicule.
- Un fichier `model.sdf` définissant la structure et les propriétés physiques du modèle.
- Un fichier `model.config` décrivant les métadonnées du modèle.
- Un sous-dossier `meshes` (si nécessaire) contenant les fichiers de modèles 3D au format `.dae`.

Ce dossier doit être placé dans :

`PX4-Autopilot/Tools/simulation/gz/models`

5.4 Intégration de l'airframe dans QGroundControl

Pour que l'*airframe* soit reconnu par QGroundControl et apparaisse dans la liste des configurations disponibles, il est nécessaire de suivre les instructions détaillées sur la documentation officielle de PX4 :

`https://docs.px4.io/main/en/dev_airframes/adding_a_new_frame.html#add-frame-to-qgroundcontrol`

Références

[1] Documentation PX4, *PX4 Autopilot User Guide*.

[2] Dépôt GitHub PX4, *PX4-Autopilot Repository*.

[3] Dépôt GitHub QGroundControl, *QGroundControl Repository*.

[4] Architecture de PX4, *PX4 Concept: Architecture*.

[5] Architecture des systèmes PX4, *PX4 Systems Architecture*.

[6] Simulation sous PX4, *PX4 Simulation Guide*.

[7] Simulation avec Gazebo (Ignition/GZ), *PX4 Gazebo Simulation*.

[8] Modèles Gazebo pour PX4, *PX4 Gazebo Models*.

[9] Développement d'un nouvel airframe, *PX4 Developer Guide: Airframes*.

[10] Ajout d'un nouvel airframe dans PX4, *Adding a New Airframe in PX4*.

[11] Middleware PX4, *PX4 Middleware Guide*.