B2-4

Logiciel plateforme PMD

Sommaire

1. Généralités

1.1 Objet

Le présent document décrit le logiciel « PMD-Connector » de pilotage de la plateforme dynamique du simulateur de vol NOVAFLY. Ce document donne des points de repère à un développeur cherchant à intervenir sur le code de PMD-Connector.

1.2 Diffusion

Propriétaires du système NOVAFLY

1.3 Acronymes

NOVAFLY	Nom du système SEVPro5-didactisé
SEV	Nom du sous ensemble châssis supérieur (Système d'Entrainement au Vol)
PMD	Nom du Sous ensemble plateforme (Plateforme Mobile Dynamique)

1.4 Fichiers référencés

Le logiciel est fourni en code source à l'emplacement suivant :
Bureau du Poste Instructeur\NOVAFLY-DATA-PI\6. Logiciels NOVAFLY\PMDConnector.4.1-src

2. Introduction

Le logiciel PMDConnector est en charge :

- De lire des données depuis le logiciel de simulation de vol (FSX ou P3D)
- De piloter les moteurs de la plateforme mobile afin de reproduire les mouvements de l'avion simulé à l'utilisateur.

3. Environnement de développement

Apporter des modifications au logiciel nécessite un poste de développement comportant les composants suivants :

- Visual Studio 2010 Express Edition ou supérieure,
- Microsoft Flight Simulator 2010 SDK. (fourni avec Microsoft Flight Simulator édition Professionnelle, Deluxe ou Gold ou avec Prepar3D),
- Framework .NET 4.0.

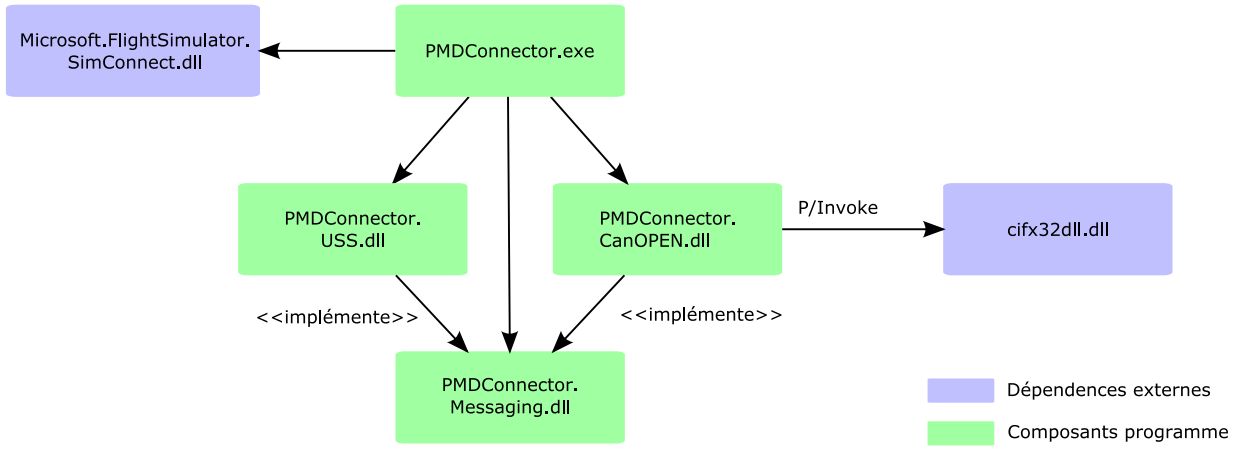
Les composants de l'application sont compilés pour une plateforme x86 (32-bits).

4. Conception du logiciel

4.1 Composants logiciels

Ce logiciel est composé :

- d'un programme .NET exécutable : PMDConnector. Il est exécuté au démarrage du logiciel de simulation de vol.
- De 3 DLL .NET :
  - **PMDConnector.Messaging**, qui contient le modèle abstrait des bus de communication.
  - **PMDConnector.CanOPEN**, qui contient l'implémentation du modèle de communication sur un bus CAN avec une carte maître Hilscher cifX CO-50 (documentation B3-5).
  - **PMDConnector.USS**, qui contient l'implémentation du modèle de communication sur un bus USS/RS485.



La dépendance externe **Microsoft.FlightSimulator.SimConnect.dll** est fournie avec le code du programme, dans le répertoire *Libs* à la racine.
La dépendance **cifx32dll.dll** est installée dans le répertoire système lors de l'installation du driver de la carte Hilscher.
Le code supporte deux bus différents pour la communication avec les moteurs : USS/RS485 et CANopen. Les nouveaux modèles de plateforme utilisent le second. Le premier sert à maintenir la compatibilité avec les modèles plus anciens.

4.2 Tests unitaires automatisés

L'environnement de développement utilise NUnit 2.6.2 pour les tests unitaires automatisés.
Les tests unitaires automatisés sont décrits dans les projets suivants de la solution :

- **NewtonRaphson.Tests**
- **PMDConnector.CanOPEN.Tests**
- **PMDConnector.Tests**

Un projet NUnit est disponible à la racine de la solution (**Tests.nunit**). Il doit être ouvert avec le programme **Libs\NUnit-2.6.2\bin\nunit-x86**.

4.3 Fichiers de configuration

Le programme PMDConnector nécessite la présence de deux fichiers de configuration :

- Un fichier de configuration **config.xml**, servant à spécifier un petit nombre de paramètres de la simulation. La plupart de ces paramètres sont historiques et ne servent plus. Ce fichier est lu par désérialisation XML (fournie par le framework .NET) de la classe **PMDConnector.Configuration**.
- Un fichier de description de courbes **curves.xml**, qui décrit des courbes de valeurs en fonction du temps. Ces courbes sont utilisées pour créer des sensations aux moments où l'avion quitte le sol et touche le sol. Les classes impliquées dans le décodage et l'utilisation de ces courbes sont **PMDConnector.CurveConfigParsing**, et toutes les classes du « namespace » **PMDConnector.Curves**.

Ces deux fichiers doivent se trouver dans le même répertoire, mais pas forcément le même répertoire que l'exécutable. **PMDConnector** peut être exécuté :

- Soit sans argument de ligne de commande, auquel cas les fichiers seront recherchés dans le répertoire "**.**" (répertoire de lancement du programme)
- Soit avec le chemin vers "config.xml" passé comme seul argument de ligne de commande (exemple : **PMDConnector C:\Temp\config.xml**), auquel cas les fichiers seront recherchés dans le répertoire spécifié.

4.4 Modèle de la plateforme PMD

Le modèle de représentation de la plateforme PMD est un ensemble de 3 moteurs :

- 3 moteurs de mouvement, situés sous la plateforme, un à l'avant, deux à l'arrière. Leurs noms respectifs dans le code sont **Front**, **RearLeft** et **RearRight**.

La position d'un moteur représente la position angulaire de la manivelle fixée dessus :

- 0° représente une position face au capteur de référence.
- Un angle positif représente le fait que la manivelle a quitté son point de référence en montant. La position la plus haute utilisée dans PMDConnector pour les moteurs de mouvement est +85°.
- A l'inverse, un angle négatif représente le fait que la manivelle a quitté son point de référence en descendant. La position la plus basse d'un moteur de mouvement est -85°.

Les configurations des moteurs sont injectées au démarrage du programme dans la méthode **Program.MakePlatform**.

4.5 Représentation des états du système

Hormis les équations dynamiques, l'état du système est représenté par 3 machines à états :

- une machine à états représentant l'état de la connexion à FSX,
- une machine à états représentant l'état de la plateforme,
- une machine à états représentant l'état d'un moteur. (3 exemplaires)

4.5.1 Connexion à FSX

Cette machine est implémentée par les classes **SEVProConnectorStateMachine, TryingToConnectState, WaitingForOpenState**, **ConnectedState** et **FinalState** dans le namespace **PMDConnector.StateMachine**.

Elle a en charge l'interaction avec l'objet SimConnect servant à récupérer les données depuis Flight Simulator.

Lorsque la machine est dans l'état "Connected", elle échange des données avec FlightSimulator et les envoie en début de flot de calcul.

4.5.2 État de la plateforme

Cette machine est implémentée par 3 classes :

- L'interface **PMDConnector.Policy.PlatformStateMachineContext**, qui décrit les événements de l'extérieur auxquels réagit la machine.
- La classe abstraite **PMDConnector.Policy.PlatformStateMachine**, implémentant la politique de haut niveau de la plateforme.
- La classe concrète **PMDConnector.StateMachine.Production.Platform.ProductionPlatformStateMachine**, qui implémente les interactions nécessaires vis-à-vis de la plateforme réelle.

Elle a en charge de réagir aux états de simulation "En simulation" ou "Hors simulation" et d'instruire les moteurs :

- En simulation, de reproduire les résultats sortis du flot de calcul, et donc de reproduire les mouvements de l'avion simulé.
- Hors simulation, de descendre en état de repos (aussi appelé position basse), puis d'arrêter les moteurs.

4.5.3 État d'un moteur

Cette machine est implémentée par 3 classes :

- L'interface **PMDConnector.Policy.EngineStateMachineContext**, qui décrit les événements de l'extérieur auxquels réagit la machine.
- La classe abstraite **PMDConnector.Policy.EngineStateMachine**, implémentant la politique de haut niveau d'un moteur.
- La classe concrète PMDConnector.StateMachine.Production.ProductionEngineStateMachine, qui implémente les interactions nécessaires vis-à-vis d'un moteur réel.

Elle a en charge de réagir aux états du moteur :

- Si le moteur est déconnecté ("Offline"),
- Si le moteur s'est bloqué en signalant une erreur,
- Si le moteur a déjà recherché ou non une position de référence.

Lorsque le moteur est dans l'état "FollowOrders/Application des consignes", le moteur s'affaire à suivre les instructions suivantes :

- Soit se déplacer vers un point,
- Soit s'arrêter,
- Soit revenir en position de repos.

Ces trois ordres sont représentés par l'énumération suivante :
PMDConnector.StateMachine.Production.PlatformOrderActions.

4.6 Flot des commandes envoyées aux moteurs

Le flot des commandes est le suivant (cf. principe de pilotage des moteurs, B2-3, §2) :

- FSX émet des mesures de simulation. Ces mesures sont représentées par la structure **PMDConnector.RequestedDataStructure**.
- Ces mesures de simulation sont interprétées avec les bonnes unités, puis envoyés aux formules de calcul des consignes de plateforme. Ces mesures sont représentées par le type **PMDConnector.MeasuresFromSimulation**.
- Ces consignes de plateforme sont ensuite converties en consignes de moteur en tenant compte du modèle structurel de la plateforme. Ce résultat est représenté par le type **PMDConnector.EngineCommand.**
- Ces consigne de moteurs (comprenant des consignes d'angle physique pour les manivelles), sont enfin converties en ordres en points-codeur pour le motoréducteur, en fonction de la résolution codeur de chaque moteur (1024 points, en l'occurrence) et du rapport de réduction du moteur à la manivelle. (100 pour les moteurs de mouvement) Le résultat de cette conversion est représenté par le type **PMDConnector.StateMachine.Production.PlatformOrder**.

On distingue les *consignes de plateforme*, qui représentent la position désirée de la plateforme à un instant *t*, et les *consignes de moteur*, qui représentent les consignes angulaires et de vitesse envoyées aux moteurs.

4.7 Formules utilisées

4.7.1 Géométrie de la plateforme PMD

Une formule statique représentant la géométrie de la plateforme mobile. Le logiciel connaît une fonction :

$$f : angle \times angle \times angle \rightarrow angle \times angle \times hauteur$$

Qui associe:

- aux trois consignes angulaires données aux moteurs
- l'angle de tangage, l'angle de roulis et le déplacement vertical de la machine.

Cette formule décrit l'effet qu'auront les consignes données aux moteurs sur l'orientation donnée à la plateforme. Ce qui intéresse le logiciel est d'exécuter la fonction inverse.

Le logiciel utilise la méthode de Newton-Raphson pour trouver l'antécédent d'une consigne (angle de tangage, angle de roulis, déplacement vertical) à travers cette fonction.

Cette formule est décrite dans la classe **PMDConnector.PMDSolver**.

4.7.2 Position angulaire de la plateforme

La formule de consigne de position angulaire de la plateforme en fonction de la simulation est décrite dans **PMDConnector.Form1.SolveProblem**.

Cette méthode utilise les méthodes de **PMDConnector.DynamicEquations**.