Rapport du Chef d'oeuvre

Systèmes Interactifs du Drone Paparazzi

par

Doscot - Mansouri - Poupard

27 Février 2005

S. Conversy

Table des matières

Chapitre 1

Introduction

1.1 Avant-Propos

Les évolutions techniques ainsi que la nécessité d'un moyen de transport rapide, sûr et performant, sont parmi les facteurs susceptibles de changer le monde actuel de l'aviation.

Ces même évolutions techniques, nous amèneront peut être un jour à nous poser la question de l'utilité du pilote à bord d'un avion. Cette question a été déjà posée par les militaires qui, vu la nature de leurs activités, sont souvent amenés à agir dans des environnements hostiles et inadaptés aux pilotes humains. L'utilisation des véhicules aériens sans pilote, ou plus communément appelés drones, reste aujourd'hui étroitement liée au domaine militaire où elle est facilement justifiable par la sécurité du pilote et la possibilité de créer des aéronefs miniaturisés.

Les avantages reconnus par les militaires sont aussi valables dans le domaine civil où plusieurs applications sont envisageables. Les drones pourraient être utiles pour effectuer des missions de surveillance de frontières, d'inspection d'infrastructures, de mesures atmosphériques ou même du transport de fret ou de passagers.

Plusieurs projets ont été réalisés et ont abouti à la construction de différents types de drones capables de voler de façon autonome ou assistée. On note qu'aujourd'hui il existe trois catégories principales de drones :

- Les HALE (High Altitude Long Endurance) : parmi lesquels on trouve le drone géostationnaire de la NASA, HELIOS. (voir fig.1.1)
- Les MALE (Medium Altitude Long Endurance) : comme par exemple le drone de l'USAir Force Global Hawk. (voir fig.1.2)
- Les minis drones.(voir fig.1.3)

Le projet Paparazzi s'inscrit dans la dernière catégorie. Il a été lancé en début 2003 par Pascal Brisset (Log/ENAC) et Antoine Drouin (CENA) et avait pour objectif le développement d'un mini drone à bas coût pour participer aux différentes compétitions en France ou à l'étranger.

FIG. 1.1 – Le drone HELIOS

FIG. 1.2 – Le drone Global Hawk

FIG. 1.3 – Le drone Twinstar du projet Paparazzi

Le système obtenu permet de faire voler de façon autonome un avion modèle réduit. L'équipe possède plusieurs drones équipés de microprocesseurs, de capteurs et d'une caméra embarquée. Une station sol assure également le contrôle et le suivi du drone en vol. Elle est constituée d'une radio commande, d'une antenne, d'un modem et d'un ordinateur. Cette station offre plusieurs fonctionnalités :

- Un éditeur de mission qui permet de programmer les plans de vol.
- Une interface de suivi qui permet d'afficher les paramètres de vol envoyés par le drone et le déroulement du plan de vol.
- Un outil de rejeu et d'analyse.
- Un outil de simulation.
- Un outil de visualisation de la vidéo transmise par le drone qui permet de reconstituer la carte du terrain.

1.2 Présentation du sujet

Depuis peu, le projet Paparazzi est devenu une activité interdépartementale à l'ENAC et compte un certain nombre d'intervenants aux compétences diverses. Plusieurs projets scolaires en rapport avec Paparazzi ont été attribués aux IENAC et aux étudiants en MASTER.

Notre chef d'oeuvre fait partie de ces projets et s'intéresse à l'interface de suivi de la station sol. Il entre dans le cadre des évolutions futures du système.

Nous étions donc chargés de concevoir et réaliser une partie de l'interface de la station sol qui permet d'assurer le service de gestion des appareils en vol. La principale fonctionnalité de cette interface était de permettre le suivi de plusieurs appareils simultanément. La seule contrainte énoncée au départ était d'utiliser des outils libres fonctionnant sur une plateforme Linux.

Etant donné que notre chef d'oeuvre ne consiste pas en un projet indépendant et autonome mais bien à réaliser une interface venant s'ajouter à un programme déjà écrit, il a fallu commencer par étudier le fonctionnement du programme existant.

Chapitre 2

Démarche de conception

2.1 Appropriation de l'existant

La première étape fut d'avoir une démarche utilisateur. Nous avons en effet assisté au cours des premiers entretiens à des simulations nous permettant de nous rendre compte des fonctionnalités existantes et des possibilités offertes par le programme, sans jamais nous intéresser au code. Nous avons pu constater par exemple que l'interface actuelle affichait des informations telles que :

– la trajectoire et la position de l'avion (voir fig.2.1)

FIG. 2.1 – La carte

– un horizon artificiel (instrument permettant de juger de l'attitude[1] d'un appareil) ainsi qu'une multitude d'informations (aussi diverses que la "qualité" de la connexion, des informations de debug, le niveau de la batterie, etc...) (voir fig.2.2)
– un suivi de la mission (voir fig.2.3)

[1] l'attitude caractérise la position relative de l'appareil (s'il est en virage, etc...)

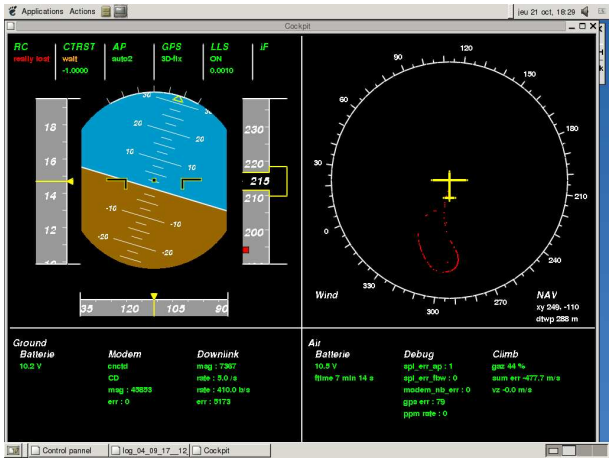

FIG. 2.2 – Le "cockpit"

FIG. 2.3 – La fenêtre de visualisation des missions

– un gestionnaire de fenêtres (voir fig.2.4)

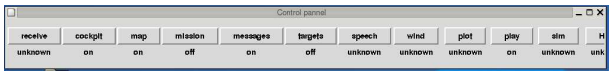

FIG. 2.4 – Le gestionnaire de fenêtres

Après l'installation (non triviale !) d'une Debian[2] et du programme sur un de nos postes, nous avons pu effectuer par nous même des rejeux de missions réelles et essayer le simulateur afin d'affiner notre compréhension et notre connaissance du programme. En parallèle, nous avons évidemment fait appel à Internet pour approfondir nos connaissances sur les drones et pour avoir une vision plus générale du projet :

– Portail du projet Paparazzi : http ://www.nongnu.org/paparazzi/
– Mémoire sur les Drones civils, Perspectives et Réalités, Pascal Brisset (2004) : http ://www.recherche.enac.fr/ brisset/enac/brisset_sapin.pdf

L'étape suivante fut de quitter la démarche utilisateur pour une démarche plus centrée développeur. Pour cela, nous avons commencé par identifier les parties du système pouvant être traitées de manière indépendante afin de ne conserver que les parties liées à l'interface.

D'ores et déjà, le système peut trivialement être divisé en deux parties physiquement indépendantes : la station sol et l'aéronef.

L'aéronef contient deux microprocesseurs (appelés MCU1 et MCU0) et des récepteurs. Un microprocesseur (en l'occurrence MCU0) se charge de traiter les informations reçues des récepteurs liés à la stabilisation et à la navigation (récepteur GPS, capteurs IR...) et échange ses données avec l'autre microprocesseur (MCU1). MCU1 reçoit quant à lui les ordres de la radiocommande et, en accord avec MCU0, agit sur les servocommandes permettant de piloter le drone.

En parallèle, MCU0 envoie des messages au sol sur le résultat de ses calculs (attitude visée, position souhaitée, etc..). Des informations sur l'état de l'aéronef (tels que le niveau de la batterie, le mode du GPS, etc..) sont également transmises au sol.

Au niveau de la station sol, ces messages, une fois réceptionnés, peuvent être écoutés par tous les programmes le souhaitant. Cela est possible car les messages sont en réalité envoyés sur un bus logiciel Ivy[3], et les programmes tiers se contentent d'écouter les messages transitant sur le bus. Il est à noter que le modem sol peut également envoyer des messages sur le bus logiciel pour donner des informations sur son état.

(voir fig.2.5)

[2]Debian est une des distributions du système d'exploitation Linux

[3]protocole développé par le CENA permettant de diffuser des informations sous forme de messages textuels à d'autres applications

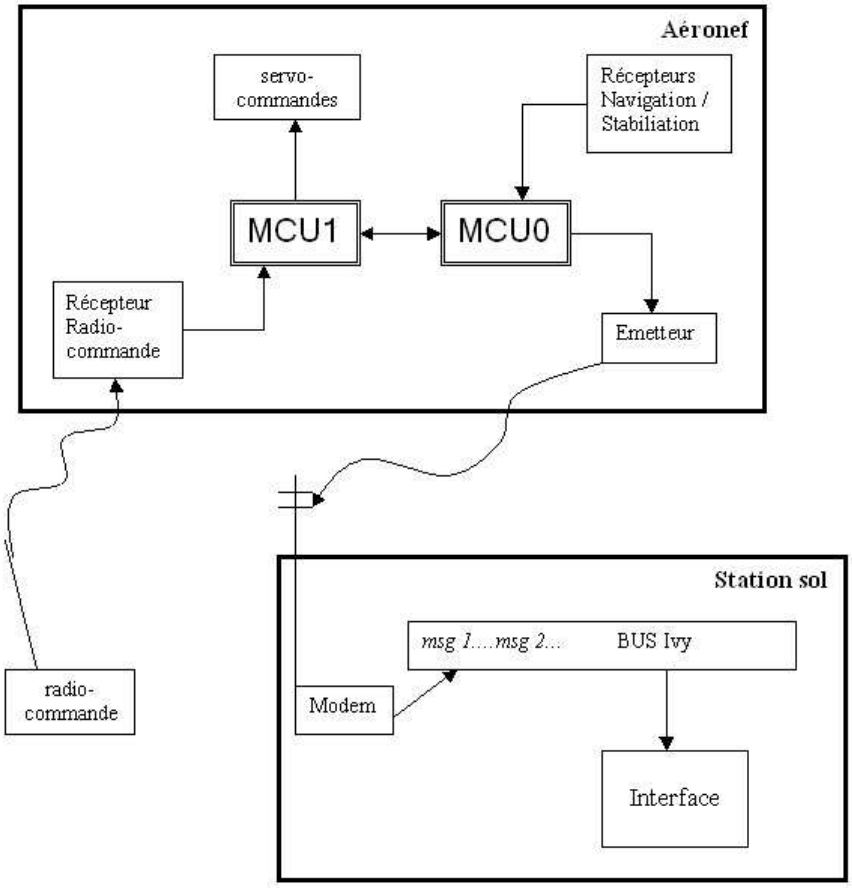

FIG. 2.5 – Schéma de fonctionnement

Au final, l'interface se contente "d'écouter" les messages transitant sur le bus Ivy et de faire les modifications d'affichage correspondantes. S'il est vrai qu'en cours de mission seuls les messages récupérés sur le bus Ivy sont à l'origine d'une mise à jour de l'affichage, l'interface peut toutefois avoir besoin d'informations "statiques" (au sens : n'évoluant pas en cours de mission). Ces informations, comme le plan de vol ou les références du terrain, sont disponibles dans des fichiers XML, dit fichiers de configuration. Il est donc nécessaire que l'interface puisse également accéder à ce type de fichiers.

2.2 Analyse des messages

Au vu de cette étude, l'étape suivante consistait logiquement à comprendre, analyser, classer les messages récupérés sur le bus Ivy. Nous avons pour cela procédé en collaboration avec Pascal Brisset et Antoine Drouin.

Un premier entretien (le 14/10), ainsi que l'échange de quelques mails, ont permis de comprendre la syntaxe utilisée ainsi que l'information exprimée à travers chaque message.

Par exemple,

```
<message id="GPS">
  <field id="mode" type="uint8" unit="byte_mask" />
  <field id="east" type="int32" unit="cm" />
  <field id="north" type="int32" unit="cm" />
  <field id="course" type="float" unit="rad" format="%.2f" />
  <field id="alt" type="float" unit="m" format="%.0f" />
  <field id="speed" type="float" unit="m/s" format="%.1f" />
  <field id="climb" type="float" unit="m/s" format="%.1f" />
  <field id="tow" type="float" unit="s" format="%.1f" />
 </message>
```

signifie que nous avons à faire à un message nommé "GPS" dont les champs nous renseignent sur : le mode du GPS, la position (east/north), la route suivie (course), l'altitude, la vitesse, le taux de montée (tous ces paramètres de vol tels que calculés par le GPS), et une information de temps (tow).

Le travail fut simplifié par le fait que la syntaxe de tous les messages est définie dans un unique fichier XML (messages.xml dans le répertoire de configuration /conf, ce fichier est disponible en annexe). Ce type d'archivage présente néanmoins un défaut : il est difficile de déterminer à quelles parties du système se référe chacun des messages. Certains, comme le message "GPS", renseignent sur un aéronef ; d'autres, comme "Modem_status", sur la station sol (ici l'état du modem sol...alors que par exemple des informations sur le modem embarqué sont disponibles dans un message nommé "Debug").

Le but de ce premier travail fut de déterminer toutes les informations potentiellement affichables ; qui diffère des messages potentiellement générés. En effet, il fallait impérativement s'affranchir de la syntaxe des messages, celle-ci étant amenée à évoluer. A l'avenir il faudra au moins ajouter l'identifiant de l'aéronef à l'origine du message pour permettre une gestion multi-drones.

L'étape suivante consistait à analyser les messages : il fallait en effet déterminer l'importance des différentes informations, toutes ne pouvant être affichées en permanence. Pour cela nous avons d'abord décidé de définir une classification, ce qui a aboutit aux définitions suivantes (par priorité décroissante) :

- Est considérée critique, une information dont l'absence met en péril le déroulement de la mission (exemple : le niveau de la batterie)
- Est considérée importante, une information qui permet un meilleur déroulement de la mission (exemple : la distance au prochain waypoint)
- Est considérée secondaire, une information utile mais dont l'absence n'a pas d'incidence sur l'efficacité de la mission (exemple : l'attitude désirée)
- Est considérée négligeable, une information qui n'a pas à être affichée (exemple : la version du boot)

Dans un premier temps nous avons essayé de déterminer par nous-même l'importance de chaque information, et ce à partir de nos propres connaissances aéronautiques, de l'utilisation du programme existant, et du rejeu de missions réelles. Il est évident qu'une démonstration "réelle" (et non sur le simulateur) nous aurait permis d'améliorer grandement notre classification, mais des contraintes de disponibilité nous ont empêché de le faire à ce stade.

Par conséquent, afin de valider notre modèle, nous avons décidé d'une réunion avec Pascal Brisset et Antoine Drouin (le 21/10), pour qu'ils puissent y apporter les modifications nécessaires en fonction de leurs expériences. En dehors de quelques corrections, l'ajout principal de cette séance fut de reconsidérer "dynamiquement" certaines informations : utiles (voir critiques) pour certaines phases de vol alors que sans intérêt pour d'autres.

Ce travail a abouti à la classification des informations, disponible en annexe.

2.3 Réalisation des modèles de taches

Mercredi 3 novembre, nous avons pu assister à une démonstration de vol sur le terrain du club d'aéromodélisme de Muret. Cette démonstration nous a permis d'évaluer, pour la première fois en condition réelle,- les tâches réalisées par les deux utilisateurs.

Toutes les séances de préparation, de vol, et d'analyse des logs ont été filmées. Après édition de la vidéo résultante, une analyse a été réalisée et a

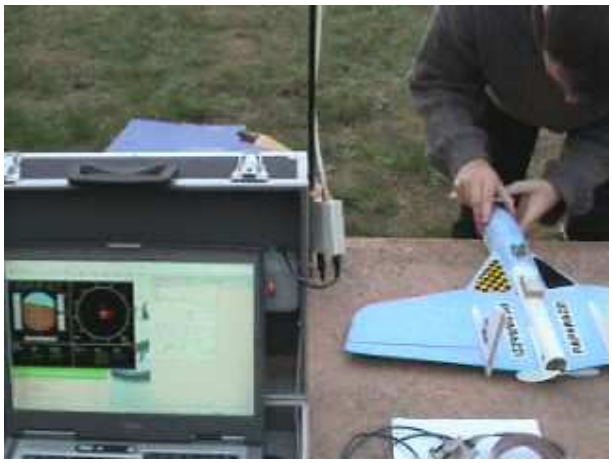

FIG. 2.6 – Le station sol et le microjet

donné lieu au rapport fourni en annexe. Un modèle de tâches a également pu être réalisé en se basant sur la vidéo (modèle lui aussi disponible en annexe).

A l'heure actuelle, il n'est possible de faire voler qu'un drone à la fois ; le modèle de tâche proposé ne correspond donc pas exactement à celui demandé (qui doit prendre en compte une gestion multi drones) mais a néanmoins l'avantage d'être la conséquence d'une utilisation réelle. Nous avons toutefois extrapolé ce premier modèle afin d'en réaliser un second prenant en compte la gestion simultanée de plusieurs aéronefs (ce nouveau modèle de tâches est également fourni en annexe).

2.4 Analyse d'un système existant

A ce stade, nous avons déterminé les spécifications du système, et effectué l'analyse des besoins et des tâches, en collaboration avec les utilisateurs.

En parallèle, dans l'idée de fournir rapidement des maquettes basse fidélité, nous avons cherché à récolter des informations sur les interfaces existantes. En raison du faible nombre de projets sur les drones (en dehors des projets militaires dont les informations sont classées), des recherches sur Internet se sont montrées infructueuses.

Il existe par contre à l'ENAC un système permettant de gérer plusieurs aéronefs, et nous l'avons donc analysé. En effet, dans le cadre des simulations de contrôle aérien utilisées pour former les ICNA[4], une personne se charge de gérer tous les avions du secteur afin de simuler le comportement des pilotes en relation avec les contrôleurs.

[4] élèves contrôleurs

Une visite au "Nouveau Bloc Simulateur" de l'ENAC nous a permis d'apprécier le système actuel au cours d'une séance de simulation et d'avoir un entretien avec les personnes l'utilisant.

Le système en question est composé de deux écrans :

1. l'un affiche une image radar du secteur,
2. l'autre affiche des informations précises sur chacun des vols (cap, vitesse, prochain waypoint...).

L'image radar permet d'avoir une vision globale de la situation : une étiquette rappelant les paramètres importants est associée à chaque avion et la position extrapolée à 2 minutes est également disponible. Il est également possible de sélectionner les avions à afficher (par niveau de vol par exemple). La personne qui joue le rôle des pilotes observe cet écran la plupart du temps pour évaluer la situation.

Le deuxième écran affiche des informations précises sur chacun des vols sous la forme de strips où apparaît la vitesse, le niveau de vol, le cap, le plan de vol, etc. La personne qui joue le rôle des pilotes observe cet écran la plupart du temps lorsqu'un contrôleur s'adresse à lui. Il peut ainsi accéder aux informations détaillées associées à l'avion "appelé" par le contrôleur, et agir en conséquence des ordres donnés : un clic, par exemple, sur le champ "FL" (niveau de vol) d'un appareil fait apparaître un menu déroulant où il est possible de sélectionner un nouveau niveau.

Les interactions se font à l'aide d'une souris, d'un clavier numérique et de deux écrans tactiles : l'un pour gérer l'affichage de l'image radar, l'autre pour gérer les communications.

Ce système se révèle assez satisfaisant d'après les utilisateurs (notamment faible nombre d'actions pour agir sur un vol). Ils lui reprochent toutefois un temps d'apprentissage élevé et un faible niveau d'interaction entre les deux écrans (il faut parfois du temps - quelques secondes - pour corréler le "strip/pilote" d'un vol avec sa trace sur l'image radar).

Bien que cette visite nous ait permis une première approche d'un système gérant plusieurs aéronefs, il faut la pondérer par les différences fondamentales entre le système observé et celui à développer. En effet, au bloc simulateur :

- les utilisateurs sont en condition d'utilisation optimale : deux écrans dans un environnement lumineux idéal, salle climatisée, aucun stress (il s'agit juste d'une simulation), etc.
- les utilisateurs "se contentent" d'effectuer les actions demandées par les contrôleurs : pas de vérification à effectuer
- le scénario joué est complètement connu et déterminé (pas d'imprévu)

Cette visite nous a permis toutefois d'avoir un feedback sur certaines interactions pouvant nous être utiles et d'influencer par conséquent la conception des maquettes basse fidélité.

Chapitre 3

Réalisation des maquettes basse fidélité

3.1 Description des maquettes

L'analyse des besoins et des tâches a donc permis un maquettage basse fidélité de l'interface. Dans un premier temps, nous avons réalisé trois maquettes basse fidélité afin de proposer différentes idées et choix de conception.

Ces maquettes sont disponibles en annexe.

3.1.1 Maquette n°1

Les informations critiques sont contenues dans des strips disposés verticalement sur la gauche de l'écran. Une carte sur la partie droite de l'écran permet d'avoir une vision globale de la situation. Les alarmes sont listées (i.e. que l'historique reste disponible) en bas à gauche.

Les strips contiennent deux boutons « cockpit » et « mission » qui permettent d'afficher ces informations successivement aux deux emplacements inférieurs droits (i.e. qu'un clic sur un bouton modifie le dernier emplacement non rafraîchi). Dans cette interface, il est également possible de faire un « drag and drop » de ces boutons directement sur l'emplacement désiré.

L'idée principale étant de pouvoir sélectionner avec précision les informations utiles, en laissant le choix de la disposition à l'utilisateur. Une autre idée est de pouvoir afficher des informations concernant différents appareils en parallèle (par exemple, il est ici possible de suivre deux missions en parallèle).

3.1.2 Maquette n°2

Les informations critiques sont disposées dans des strips disposés verticalement sous forme d'onglet à gauche de l'écran. Une gestion générale se

contente d'afficher une carte ainsi qu'une zone d'alarme affichant les alertes sous forme d'historique.

Un clic sur un strip affiche les données importantes de l'avion, la zone alarme, ainsi que une zone où l'on peut visualiser au choix la carte, le cockpit et la mission ou les trois en même temps.

Cette maquette a été conçue dans le souci d'une gestion des alarmes, d'une visibilité permanente des informations importantes ou critiques (pas de pop up d'alerte ou de clic droit affichant un menu ou des informations) et d'un accès clair aux informations désirées (en connaissance des informations qui seront masquées par cette nouvelle visualisation)

3.1.3 Maquette n°3

La première zone est constituée d'une palette de strips disposée horizontalement en haut de l'écran. Chaque strip correspond à un avion et résume ses paramètres de vol critiques.

En dessous de cette palette, une deuxième palette représente l'évolution des missions en cours, cela était destiné à permettre à l'utilisateur d'avoir une idée globale sur l'évolution de la flotte ; l'utilisateur a le choix de l'afficher ou non.

La principale partie de l'interface est dédiée à la carte, sur laquelle on peut voir les différentes étiquettes associées aux avions en vol. La partie cockpit est globalement similaire aux autres maquettes et reprend les informations disponibles sur l'interface actuelle. Seuls les strips est les étiquettes sont interactifs. Un clic sur le strip ou sur l'étiquette permet d'afficher le cockpit de l'avion concerné.

Les alarmes sont affichées dans une zone dédiée qui contient une description brève du problème et permet d'accéder rapidement aux paramètres de l'avion qui a généré l'alarme.

3.2 Validation des maquettes

Ces maquettes ainsi que leurs diverses interactions possibles ont été présentées aux utilisateurs. Nous leur avons proposé différents scénarii (disponibles en annexe) dans le but de valider certains éléments de conception.

A travers ces scénarii, nous avons essayé de prendre en compte l'ensemble des activités des utilisateurs telles que :

- Accéder aux informations importantes des différents aéronefs.
- Gérer des décollages (ou atterrissages) successifs.
- Gérer des avaries (niveau de batterie faible).
- Gérer des évènements imprévus (pas d'actualisation des données, le retour automatique d'un avion qui se serait trop éloigné).
- Visualiser l'évolution des missions.

Ceci a donnée pour chaque maquette (disponibles en annexe) l'évaluation suivante :

3.2.1 Maquette n°1

- Avantages :
  - Des strips contenant toutes les informations les plus critiques sont affichés en permanence ce qui permet de suivre tous les avions.
  - Une zone affichant les messages d'alarme à un emplacement fixe et qui permet également d'avoir un historique.
  - Un procédé de drag and drop permet d'afficher les missions et le cockpit d'appareil différents, ce qui permet de suivre l'évolution en parallèle de deux missions ou de deux cockpit différent.
- Inconvénients :
  - Risque de confondre les missions et les cockpits des appareils si le cockpit d'un avion est affiché à côté de la mission d'un autre.
  - Des clics droits faisant apparaître des données peuvent cacher des informations importantes que les utilisateurs voudraient suivre.
  - Une représentation de la carte de type radar ne permet pas de voir la trajectoire laissée par les avions.

3.2.2 Maquette n°2

- Avantages :
  - Des strips ayant les mêmes caractéristiques que ceux de la maquette n°1.
  - Une zone d'alarme ayant les mêmes caractéristiques que ceux de la maquette n°1.
  - Chaque zone d'information a un emplacement fixe ce qui évite une réorganisation des fenêtres déstabilisante et un masquage non voulue des informations
  - La visualisation dans une zone fixe du cockpit et des données jugées moins critique concernant un seul avion. La sélection de l'avion souhaité se fait par clic gauche sur le strip qui lui est associé.
  - Les caractères des informations qui n'ont pas été rafraîchies depuis un temps déterminé passent en orangé.
- Inconvénients :
  - La carte n'est pas affichée en permanence.

3.2.3 Maquette n°3

- Avantages :
  - Des strips ayant les mêmes caractéristiques que ceux de la maquette n°1.

  - Une distinction par l'affectation dans 2 zones précises de l'interface entre les données concernant le sol et celles concernant l'avion.
  - La visualisation dans une zone fixe du cockpit et des données jugées moins critique concernant un seul avion. La sélection de l'avion souhaité se fait par clic gauche sur le strip qui lui est associé.
- Inconvénients :
  - Une réorganisation de l'écran déstabilisante pour faire apparaître d'autres informations.
  - Des pops up d'alarmes susceptibles de masquer des informations.

3.3 Création et validation d'une première maquette de synthèse

Suite à cette première séance de validation, une maquette de synthèse a été réalisée dans le souci de conserver les avantages des maquettes précédentes. Au cours d'une séance de brainstorming, nous avons décidé des éléments graphiques suivants :

- Des strips de tous les avions en vol contenant les informations jugées utiles par les utilisateurs.
- Une zone fixe de l'interface affichant les messages d'alarmes et les conservant sous forme d'historique avec une scroll barre.
- L'affectation à une zone fixe de l'écran des informations concernant la station au sol distincte des données concernant les avions.
- Une zone fixe de l'écran affichant le cockpit, la mission et les données jugées moins critiques ainsi que toutes les informations du strip concernant un seul avion.
- Une carte affichée en permanence permettant la visualisation de la position de l'avion, de sa trajectoire ainsi qu'une étiquette associée indiquant les données de navigation les plus importante.

Et, comme moyens d'interactions :

- Un clic gauche sur le strip ou l'étiquette associée à l'avion fait apparaître dans la zone avion les données le concernant.
- Le strip de l'avion dont les données sont affichées dans la zone avion est mis en valeur.
- Dès qu'un message d'alarme est signalé, la zone alarme clignote en rouge et un son lui est associé si le message est critique.
- Un clic sur la zone d'alarme stoppe le clignotement : l'utilisateur a pris connaissance de l'alerte.
- Les alarmes toujours actives restent en rouge sur l'écran, les autres passent en grisé.
- Si un avion est concerné par une alarme, son strip sera entouré de rouge. Si l'alarme concerne la station sol c'est la zone associée qui sera encadrée. Les champs des données concernées par l'alarme passeront

également en rouge.
- Les champs qui n'ont pas été rafraîchis depuis un certain temps passent en grisé.

Puis suite à une nouvelle réunion avec les utilisateurs, de nouveaux besoins sont apparus et donc de nouveaux éléments graphiques et moyens d'interactions ont été imaginés.

- Les éléments graphiques :
  - La possibilité de passer la carte en mode plein écran. En effet, il n'est pas toujours nécessaire de surveiller les données d'un avion précis ainsi que la station sol. Dans ce mode, les strips sont conservés et la carte vient alors recouvrir la zone avion, sol et alarme. La carte étant plus grande et donc les contraintes d'encombrement plus faible, nous pouvons rajouter des informations sur les étiquettes.
  - S'il y a une alarme en mode carte plein écran la zone alarme réapparaît sur la carte.
  - Un nouveau besoin exprimé fut de visualiser la vidéo d'une caméra embarquée sur le drone. Ceci a conduit à remplacer, par un clic sur une icône, la « zone cockpit » par la vidéo.
- Les interactions :
  - Un clic droit sur l'étiquette d'un avion fait apparaître un menu. On peut y choisir de faire disparaître l'avion de la carte (ainsi que sa trajectoire et son étiquette) ou de régler la longueur de sa trace.
  - Si l'avion a été effacé volontairement de la carte une icône le signale sur son strip ; un simple clic sur celui-ci suffit alors à le faire réapparaître.

Les utilisateurs ont également précisé les informations manquantes ou inutiles sur les strips et les étiquettes.

Un nouveau scénario a été mis au point, en collaboration avec les utilisateurs, pour tester cette nouvelle maquette :

1. On lance le 1er avion sans le régler en mode auto1.
2. L'avion se met en attente au WP 1.
3. On lance le 2e sans le régler en mode auto1.
4. Le décollage automatique est interrompu (cause : assiette incliné et/ou trajectoire sur la carte incorrecte) : on repasse en manuel.
5. On règle l'appareil 2 (à l'aide des messages dans les settings).
6. Une fois réglé, on le passe en auto1 et il reprend la mission (i.e mise en attente autour du WP 1 à 50m).
7. Déroulement du reste de la mission qui consiste en un suivi de WP.
8. Après avoir passé le WP 2, la batterie au sol devient faible.
9. 10 secondes plus tard, la batterie de l'avion1 devient faible ; On décide de le ramener soit en manuel soit en auto1 en passant tous les blocs de la mission.

10. L'avion1 atterrit sans problème.
11. L'avion 2 continue sa mission en passant par le WP 3 mais on perd le lien descendant avec celui-ci.
12. atterrissage de l'avion 2 en mode auto1.

Après avoir joué ce scénario les utilisateurs ont été satisfaits des éléments graphiques, des informations affichées et des moyens d'interactions présentés. Cette maquette présentait néanmoins quelques défauts et nous avons donc réitéré le cycle de conception, qui a abouti à la réalisation d'une nouvelle maquette.

Chapitre 4

Réalisation de la maquette finale

4.1 Présentation de la maquette

Deux inconvénients majeurs ont été détectés sur la dernière maquette de synthèse :

- Une corrélation insuffisante entre le cockpit et le strip correspondant.
- L'impossibilité de gérer un nombre de strips supérieur à quatre.

Nous avons examiné ces problèmes et apporté de nouvelles solutions (les QOC correspondant sont disponibles en annexe).

4.1.1 Problème de l'insuffisance de corrélation

L'encadrement du strip indiquant la corrélation avec le cockpit affiché a été jugé insuffisante. Quand la distance entre les deux augmente, il devient moins évident de faire le lien.

Solution trouvée :

Après avoir réfléchi sur des effets de texture ou de connecteurs (voir QOC), nous avons finalement adopté un système d'onglet. Le strip étant affiché dans l'onglet et le cockpit dans son contenu. Nous avons cependant dû réorganiser l'interface. Les strips (donc les onglets) étant susceptibles de s'étaler sur toute la largeur de l'interface, il fallait qu'il en soit de même pour le cockpit.

(voir figure 4.1)

4.1.2 Réorganisation des éléments graphiques de l'interface.

La carte a été placée en bas à gauche de l'interface avec la mission haut dessus en correspondance verticale. Les utilisateurs auront ainsi une meilleure surveillance du bon déroulement de la mission sur la carte. Le reste des

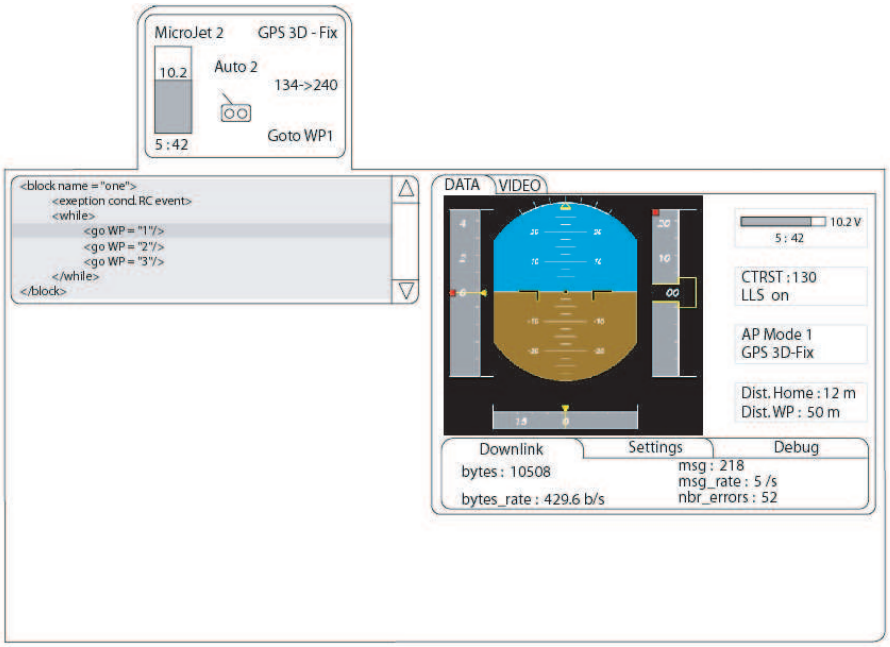

FIG. 4.1 – Le cockpit

informations du cockpit a été placé à droite sous les strips. Les éléments graphiques "Ground" et "ALRT" ont conservé leur place initiale.

(voir figure 4.2)

4.1.3 Problème du nombre de strips limité

Au vu du nombre important d'informations critiques à représenter, la taille d'un strip est très coûteuse en place sur l'interface. Cette contrainte limite à quatre le nombre maximal de strips que l'interface peut afficher.

Solution trouvée :

La solution retenue fut une diminution de la taille des strips en fonction du nombre de drones en vol. Seul le strip sélectionné conserve la taille et les informations initialement prévues. Trois modes d'affichage ont été décidés :

- un affichage maximal du strip pour une gestion d'un nombre de drones inférieur à 4. Les strips conservent la taille et le nombre d'informations prévues par la précédente maquette.
- Un affichage moyen pour un nombre de drones compris entre quatre et six.
- Un affichage minimal pour un nombre de drones compris entre sept et dix.

Ceci s'accompagne bien évidemment d'une diminution du nombre d'in-

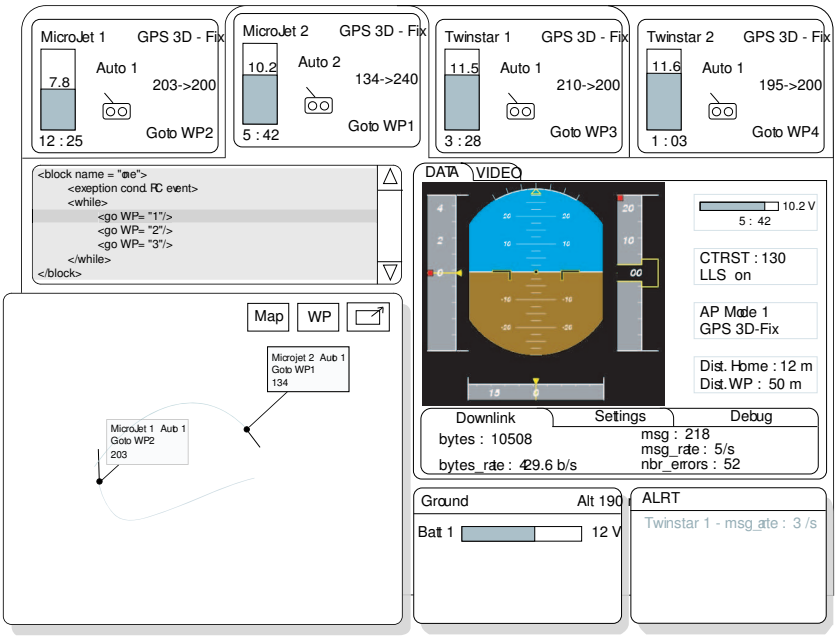

FIG. 4.2 – L'interface

formations à représenter ainsi que leur réorganisation dans le strip afin de gagner de la place. Il a fallu pour cela hiérarchiser à nouveau à l'aide des utilisateurs les données contenues dans le strip.

4.1.4 Hiérarchisation des données du strip

Comme décrit plus haut, le mode d'affichage maximal conserve toutes les données prévues par la maquette précédente. Le mode moyen conserve l'identifiant de l'appareil, le niveau de batterie, la durée de vol, le mode de pilotage et la liaison radio. Toutes les données de navigation, jugées moins importante, ont été supprimées. Le mode minimal ne conserve que le niveau de batterie et le temps de vol.

4.1.5 Réorganisation des données dans le strip

Pour une meilleure cohérence des strips avec les différents modes d'affichage, les données les plus importantes ont été placées de la gauche vers la droite, correspondant au sens de lecture. Elles sont disposées suivant trois colonnes, ainsi lors du passage du mode maximal au mode moyen la colonne de droite est supprimée. Pour le moyen vers le minimal, c'est au tour de la colonne du milieu de disparaître. Les informations conservent ainsi la même disposition relative et on obtient une meilleure cohérence entre les différents modes.

Comme la représentation des données dans le strip ne prête pas à confusion, il a été décidé de supprimer le nom du champ correspondant pour gagner de la place. Dans cette optique de gagner de la place dans le mode moyen et minimal, la jauge de la batterie a été placé de manière verticale. (voir fig 4.3 à 4.5)

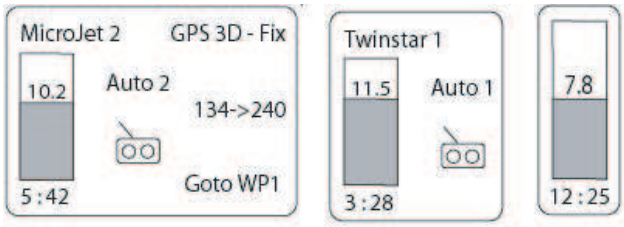

FIG. 4.3 – Les différents modes du strip

4.1.6 Différentiation entre les éléments graphiques commun et spécifique

Du fait de leur fonction, les strips et le cockpit sont spécifiques à chaque drone ; tandis que la carte, le ground et l'alarme leur sont communs. Il a été jugé nécessaire de souligner cette différence. En effet les éléments spécifiques ont été regroupés sous forme d'onglet. Les données du cockpit sont donc modifiées lorsque l'on change de drone, contrairement aux données des éléments graphiques communs. Ces éléments de natures différentes sont mis en évidence en leur associant une ombre portée, ce qui leur donne une profondeur différente.

4.1.7 Insertion de l'image de la vidéo embarquée

Ce besoin des utilisateurs est intervenu en cours de projet. Il a donc fallu trouver un moyen d'insérer cet élément graphique qui, de par sa fonction, est de taille très importante.

Solution trouvée :

Comme l'image vidéo est spécifique à chaque avion et que les utilisateurs n'ont pas besoin de la voir en permanence, il était plus judicieux de le mettre dans le cockpit. Sa taille importante demandait obligatoirement que sa visualisation soit interchangeable avec d'autres éléments graphiques du cockpit. Au vu de la place importante que prend la vidéo la seule place possible était la partie droite du cockpit. Cette position avait en outre l'avantage de la placer en correspondance avec la carte, facilitant ainsi la tâche d'acquisition d'image des utilisateurs. Ce choix est également justifié par le fait que les utilisateurs ne peuvent pas faire les tâches d'acquisition vidéo et de surveillance des données du cockpit en parallèle. Chacune d'entre elle demandant trop

d'effort cognitif, elles se font de manière séquentielle. La solution la plus naturelle pour montrer le caractère interchangeable de la vidéo et de la partie droite du cockpit, était de les placer dans des onglets.

4.1.8 Interactions possibles avec l'interface

- Interactions avec le cockpit et les strips.
  Comme décrit plus haut, le cockpit et les strips sont reliés par le système d'onglets. Dès que l'on clique sur un strip, l'interface affiche le cockpit correspondant. Dans les modes moyen et minimal, seul le strip sélectionné est affiché en mode maximal.
  De la même manière on fait apparaître l'image vidéo en cliquant sur l'onglet « vidéo », et la partie droite du cockpit correpondant aux données en cliquant sur l'onglet « cockpit ».
  Un autre système d'onglets gère l'affichage des messages de downlink, settings et debug.
- Interactions avec la carte.
  Un clic sur l'avion ou l'étiquette associée à cette avion ouvre l'onglet du cockpit correspondant.
  Un clic droit sur l'avion ou l'étiquette fait apparaître un menu offrant les options suivantes :
  - masquer étiquette
  - masquer la trace du drone
  - regler la trace du drone avec un slider

  Trois boutons situés en haut à droite de la carte offrent les options d'affichage suivantes :
  - passage de la carte en mode plein écran
  - afficher/masquer les waypoints
  - afficher/masquer le fond de carte

  Il est à noter qu'une icone apparait sur le strip d'un appareil s'il est masqué sur la carte. Un clic sur cette icone permet alors de ré-afficher l'appareil en question.

Une description plus complète des interactions est fournie en annexe.

4.2 Traitement des alarmes

Dans la suite, une alarme n'implique pas forcément la notification de l'utilisateur, mais signifie l'occurrence d'un évènement non attendu qui peut modifier l'état du système.

La nouvelle présentation des données suppose un traitement efficace des alarmes générées et surtout une notification facilement perceptible par l'utilisateur sans pour autant perturber l'avancement de sa tache principale.

En effet, la nouvelle disposition des données sur l'écran autorise le fait de cacher certains paramètres concernant des drones en vol. Ceci dépend

également du nombre d'appareils actifs.

Il est évident que la tache de l'utilisateur sera d'autant plus délicate et difficile que le nombre de drones est grand. Le système a donc le devoir d'aider l'utilisateur à résoudre les incidents et ne doit surtout pas le paniquer.

Il est à noter que les drones utilisés actuellement sont de plus en plus sûrs, ce qui nous a encouragé à proposer un système de pilotage par « alarmes » : tant que le système fonctionne normalement, l'utilisateur n'a pas besoin de voir les informations concernant tous les appareils ; il n'est averti de l'état du système que si un incident survient.

Le nombre d'alarmes à générer augmentera d'autant plus que le nombre de paramètres non visibles est important. Il est donc nécessaire de classer les alarmes suivant l'importance des incidents qui les ont provoquées ; et ce, dans le but de les montrer de façons différentes à l'utilisateur.

4.2.1 Les alarmes indépendantes de la configuration du système

Les évènements suivants sont susceptibles de générer une telle alarme :
- Une chute significative du niveau d'une des batteries.
- Le silence radio d'un avion pendant une durée jugée trop longue.
- Un disfonctionnement du modem.
- Un message erroné provenant du calculateur de bord d'un avion.

Ces alarmes ne dépendent pas de la façon dont sont disposées les informations.

La génération d'une telle alarme signifie qu'un incident risque de mettre en péril le système. L'utilisateur doit par conséquent être impérativement averti d'une manière efficace, perceptible et facilement compréhensible pour qu'il puisse agir dans les meilleurs délais. Seul le rétablissement du paramètre générant l'alarme permet le retour du système à son état initial.

En plus de son apparition dans l'historique des alarmes, une telle alarme doit modifier l'état du système pour indiquer une situation dangereuse. Nous avons jugé inutile de modifier ostensiblement l'état du « strip » concernant l'avion en difficulté car cela risque de perturber l'utilisateur et de dissiper son attention au cas où il y aurait plusieurs avions en difficulté.

La solution que nous proposons est d'attirer l'attention de l'utilisateur vers la zone d'affichage des alarmes où il trouvera la description de l'alarme et pourra atteindre directement l'élément concerné. Un affichage centralisé assure l'accès de l'utilisateur à toutes les alarmes et évite qu'il n'en oublie.

Une alarme prend alors la forme suivante :

```
<Nom de l'appareil> <paramètre concerné> : <valeur du paramètre>
```

Nous avons décidé que la meilleure façon d'attirer l'attention de l'utilisateur est de faire vibrer la fenêtre dédiée à l'affichage des alarmes à chaque

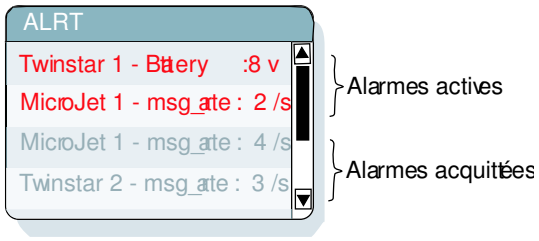

FIG. 4.4 – La fenêtre des alarmes

fois qu'une alarme critique survient. Les vibrations sont périodiques et ne s'arrêtent que si l'utilisateur acquitte toutes les alarmes.

Une telle alarme sera accompagnée d'un son pour avertir l'utilisateur s'il n'est pas en face de son écran ; néanmoins, l'utilisateur doit avoir le choix d'activer ou de désactiver ce son qui sera réitéré automatiquement si l'utilisateur reste inactif. (Cf. Notifications sonores)

En acquittant l'alarme (nous verrons comment l'utilisateur pourra le faire dans la partie suivante), les informations sur l'avion concerné passent en premier plan et les données qui ont généré l'alarme sont mises en valeurs. Le strip de l'avion sera aussi marqué d'une couleur différente (a priori le rouge), et cette distinction ne disparaît qu'avec la disparition de l'origine de l'alarme.

Pour permettre à l'utilisateur de voir toutes les alarmes non acquittées, ces dernières sont toujours visibles dans la fenêtre des alarmes. En d'autres termes, si l'utilisateur ne voit plus de texte rouge dans la fenêtre des alarmes, c'est qu'il les a toutes acquittées.

D'autres solutions pourraient être envisagées ; plus de détails concernant la notification des alarmes sont disponibles en annexe dans les QOC.

4.2.2 Les alarmes provoquées par les informations non visibles

Ces alarmes sont générées quand un paramètre non visible passe en dessous d'une valeur critique. L'importance d'une telle alarme dépend de la criticité du paramètre qui l'a provoquée. En effet, tant que le mode du GPS, par exemple, est en 3D, l'utilisateur n'a pas besoin de voir cette information, mais quand cette information est cachée et que le mode du GPS passe en 2D, ce qui signifie que les informations sur la position et l'altitude de l'avion sont erronées, l'utilisateur doit être averti.

Pour avertir l'utilisateur, nous avons choisi de lui indiquer le nouvel état du système au lieu d'attirer son attention vers la zone d'alarmes. Plus précisément, l'alarme n'est pas transmise à l'utilisateur (car son intervention ne changera pas l'état du système), mais nous affichons les informations altérées

en tant que "non fiables". Dans l'exemple du mode du GPS, les informations concernant la position et l'altitude de l'avion (affichées sur la carte ou dans le cockpit) seront accompagnées d'un signe qui montre qu'elles sont erronées.

Pour que ce traitement soit possible, il faut faire le lien entre les différents paramètres de vol pris en charge par l'interface.

4.2.3 Les alarmes de non rafraîchissement

Ces alarmes sont provoquées par l'absence d'un message durant une période déterminée. Cette absence peut être causée par une mauvaise transmission.

Il ne s'agit pas ici d'une vraie alarme mais plutôt d'un problème de fiabilité des données. Comme solution à ce problème, nous avons proposé de diminuer la saturation de la couleur de l'information affichée en fonction de son taux de rafraîchissement ; c'est-à-dire que plus l'information est vielle moins sa couleur est saturée. Cela donnera une idée à l'utilisateur sur le taux de rafraîchissement des paramètres de vol.

4.2.4 Acquittement des alarmes

Une alarme transmise à l'utilisateur doit être acquittée dans le but de nous assurer que ce dernier a bien pris connaissance du disfonctionnement du système et de son origine. Si l'alarme n'est pas acquittée, elle reste visible dans la zone d'affichage des alarmes, et dans certains cas, cette fenêtre continue de vibrer périodiquement.

Nous avons voulu concevoir un système de traitement et de notification d'alarmes de façon à ce que l'utilisateur puisse commencer la résolution du problème dès qu'il perçoit un changement dans l'état du système. La perception de ce changement doit être le premier élément de la solution. En effet, la notification doit apporter à l'utilisateur toutes les informations nécessaires sur le disfonctionnement et lui permettre d'atteindre l'avion en difficulté le plus rapidement possible.

Pour cela, nous avons imaginé l'interaction suivante pour acquitter une alarme :

Les informations sur l'alarme apparaissent dans la fenêtre d'affichage des alarmes. Ces informations donnent une idée sur l'avion en difficulté, le paramètre de vol qui a provoqué le disfonctionnement et aussi sa valeur.

Si l'utilisateur juge que ces informations sont suffisantes et qu'il n'a pas besoin d'autres éléments pour traiter le problème, il peut acquitter l'alarme avec un clic droit sur la ligne la représentant. S'il considère qu'il a besoin d'autres informations pour résoudre le problème (reprendre l'avion en mode de guidage manuel, par exemple), il acquitte l'alarme avec un clic gauche ; cette action lui permet en même temps d'afficher le cockpit de l'avion concerné.

Cette interaction offre à l'utilisateur un gain de temps considérable et lui laisse la possibilité de décider de l'importance de l'incident.

4.2.5 Notifications sonores

Les utilisateurs ont parfois tendance à quitter l'écran des yeux pour chercher l'avion dans le ciel. Il est déjà arrivé par exemple qu'un avion se soit craché parce que les utilisateurs n'avaient pas vu que le niveau de la batterie était descendu trop bas.

Pour cette raison, nous avons décidé de rajouter des notifications sonores pour les alarmes critiques.

La notification est en fait un message explicite décrivant brièvement l'alarme. Cette notification est désactivée dès que la présence de l'utilisateur devant son écran est détectée (mouvement de la souris ou touche pressée).

4.3 Choix du langage de programmation

Deux choix étaient possibles :

1. une programmation en java, le langage objet de référence
2. une programmation en PERL, le langage utilisé pour réaliser l'interface actuelle.

Les utilisateurs étant familiers avec ces deux langages, ils nous ont laissés le choix.

Nous avons évalué rapidement ces deux langages (suite, entre autres, à une discussion avec S. Conversy et J-L. Vinot)

La programmation en java présente les caractéristiques suivantes :

Avantages :

- Langage de référence, transportable, stable et facile à utiliser pour générer des interfaces.
- Langage familier des concepteurs du projet donc ne nécessitant pas de formation particulière.

Inconvénients :

- L'interface existante étant réalisée en PERL, aucune réutilisation du code associé n'est possible, il faut tout reprendre à zéro.

Quant à la programmation en PERL, elle présente les caractéristiques suivantes :

Avantages :

- Langage familier des utilisateurs qui l'ont utilisé pour coder l'interface actuelle.
- Possibilité de réutiliser le code précédent.
- Possibilité de générer les composants interactifs de l'interface à partir d'une vue simplifié de celle-ci dessinée avec Adobe illustrator (ou à l'aide de générateur tel que Glade).

– Possibilité d'utiliser le simulateur déjà existant pour tester les interfaces, sans avoir à l'adapter.

Inconvénients :

– Langage nécessitant une auto-formation.
– Langage réputé pour être peu clair syntaxiquement.

Après concertation il a été jugé favorable de programmer en PERL, ce langage présentant des avantages plus intéressants que Java pour des inconvénients mineurs.

PERL a donc été retenu pour programmer l'interface.

Chapitre 5

Evolution du projet

5.1 Le projet Paparazzi

Le projet Paparazzi est devenue une activité officielle au sein de l'ENAC, ce qui lui permet d'avoir une structure lui donnant plus de facilité (comme une salle réservée). Néanmoins étant donné l'absence d'un des deux membres principaux du projet, l'évolution s'est ralentie. Les essais en vol par exemple sont plus difficiles à organiser (nous avions vu dans le modèle de tâches qu'ils nécessitaient deux personnes au sol).

Les principales modifications apportées concernent la gestion multi-drones :

- La forme des messages a été modifiée pour intégrer l'identifiant de l'appareil (comme cela avait évoqué précédemment)
- Des tests sur simulateur ont été effectués pour constater l'évolution de plusieurs appareils simultanément.

5.2 Nouveaux objectifs

Etant donné la charge de travail complémentaire (i.e. ne concernant pas la chef d'oeuvre), nous n'avons pas pu tenir le planning initial et commencer la phase de développement (codage de l'interface) début février.

Nous avons cependant poursuivi le travail de conception, et, suite à des réunions de type « brainstorming » (avec Stéphane Conversy et Jean-Luc Vinot) ou de type « validation » (avec Antoine Drouin), nous avons pu faire évoluer la maquette présentée lors de la soutenance intermédiaire. Parallèlement, et comme prévu dans le planning, nous avons rencontré J-L Vinot pour une initiation au Perl et ainsi être prêts à commencer le codage de l'interface.

Le développement de l'interface sous Perl peut ainsi débuter fin février en ayant bouclé la phase de conception. Nous comptons réaliser d'ici la soutenance une interface opérationnelle qui, faute de temps, ne pourra cependant pas suivre de tests robustes sur simulateur et être testée en condition réelle

d'utilisation.

Toutefois, les utilisateurs disposeront de la structure d'une interface issue de plusieurs cycles de conception et pourront, étant donné leurs compétences - en Perl notamment -, l'intégrer totalement à leur projet en effectuant les tests adéquats.

Chapitre 6

Conclusion

Tout au long du chef d'oeuvre, nous avons essayé de profiter au maximum de la disponibilité des utilisateurs et de notre maître de projet en organisant des réunions périodiques et en assistant aux séances de démonstration, en situation réelle ou sur simulateur, afin de comprendre leurs besoins réels et leurs exigences.

La démarche itérative que nous avons adoptée pour arriver à la maquette papier actuelle était la seule manière de nous assurer que nous avions fait les bons choix. Comme appris durant l'année, nous avons placé les utilisateurs au centre de la conception et réitéré plusieurs fois le cycle de conception.

Une telle démarche a pour avantage d'aboutir à un résultat exploitable et d'éviter le rejet de l'interface par les utilisateurs. Ce fut la première fois que nous adoptons ce type de démarche dans le cadre du développement d'un logiciel, ce qui en fait un des apports les plus intéressants que nous avons reçu au cours de ce projet. Nous avons toutefois eu du mal à évaluer la durée de la phase de conception au cours de ce premier essai. Malgré le retard, nous avons décidé de privilégier la conception sur le codage.

Comme nous l'avons remarqué plus haut, nous avons gardé un contact étroit avec les deux utilisateurs qui nous ont aidé à surmonter les quelques difficultés que nous avons rencontrées.

Il va sans dire que ce sujet était pour nous une occasion de découvrir un nouvel aspect de l'aviation civile et surtout de participer à un projet innovateur et prometteur.

Pour finir, nous tenons à remercier les deux collaborateurs du projet Paparazzi pour leur disponibilité et leur coopération.

Chapitre 7

Glossaire

1. Cap : Sens dans lequel est dirigé l'axe longitudinal d'un aéronef, généralement exprimé en degrés par rapport au nord (vrai, magnétique, au compas ou du canevas).
2. Cockpit : dans notre cas il fait référence à certains composants de l'interface qui simule le fonctionnement du PFD (primary fligth display)
3. Ivy : protocole développé par le CENA permettant de diffuser des informations sous forme de messages textuels à d'autres applications
4. Log :un fichier contenant tous les message reçus par le modem.
5. Mission : une suite d'actions linéaires ou répétitives
6. Niveau de vol : Surfaces isobares, liées à une pression de référence spécifiée : 1013,2 mb (29,92 pouces de mercure), et séparées par un intervalle de pression spécifié
7. Phase de vol : étape d'une mission (ex. décollage, atterrissage)
8. Plan de vol : comporte une description du déroulement d'un vol (Ex. les waypoints, les coordonnées de référence, les altitudes)
9. Secteur : périmètre sous la surveillance d'un contrôleur aérien
10. Strip : terme utilisé par les contrôleurs aériens pour désigner une bande de papier résumant les principales informations concernant un avion : identifiant, cap, vitesse...
11. Version du boot : version de l'autopilote
12. Waypoint :Dans un système de navigation de zone, point non balisé, prédéterminé par ordinateur par lequel passe la route à suivre

Bibliographie

[1] Site du projet Paparazzi : http ://www.nongnu.org/paparazzi/

[2] Mémoires sur les drones civils, perspectives et réalités par Pascal Brisset, 2004

[3] Perl en Action, exemples de solution pour les programmeurs en Perl, par Tom Christiansen et Nathan Torkington aux Editions O'Reilly

[4] Document Perl, version du 3 septembre 2003, par Paul Gaborit, Publication du CENA

[5] Manuel Tk-Zinc, Publication du CENA

Chapitre 8

Annexes

- – Plan de Développement
  - Dossier de Spécifications
  - Rapport des réunions
- – Analyse des messages
  - fichier messages.xml
  - Liste des alarmes classées par ordre de sévérité
- – Analyse de la vidéo
  - Modèles de tâches
  - Rapport d'ergonomie
- – Maquettes basse fidélité
  - Scénarii
  - Description complète de la maquette basse fidélité finale
  - Rapport de brainstorming et QOC de justification
  - Compte-rendu des tests d'évaluation