



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

AER8500 - Informatique embarquée de l'avionique

Mini-projet de système avionique simulé

Rapport final

Arthur Van Betsbrugge 1955439

Katherine Zamudio-Turcotte 1951929

Travail présenté à:

Habacuc Honvo

Polytechnique Montréal

Le 17 avril 2023

Table des matières

1. Division des tâches durant le projet	3
2. Algorithme pour la fonction avionique	3
3. Tâches critiques	5
4. Plan de contingence	6
Commentaires sur le projet	7

1. Division des tâches durant le projet

Le tableau suivant décrit la répartition des tâches qui s'est appliquée durant le projet.

Tableau 1. Division des tâches selon l'ordre chronologique

Katherine	Arthur
Création du projet : classes C++, code d'initialisation des sockets	Communication par socket et tests d'envoi de paquets TCP
Interface-utilisateur avec PySimpleGUI	Encodage/Décodage des paquets ARINC429 côté calculateur et côté agrégateur en plus des mises à jour de l'affichage sur l'interface-utilisateur
Conception/programmation de l'algorithme avionique : changement d'altitude selon le temps, atténuation de l'angle et de la vitesse à l'approche de la cible, implémentation de la chute libre	Validation des entrées de l'utilisateur dans l'interface-utilisateur
Rédaction du rapport et ajustements aux avertissements affichés sur l'UI	Rédaction du rapport

2. Algorithme pour la fonction avionique

Notre algorithme avionique est grandement simplifié par rapport à un algorithme se basant sur des instruments de vol réguliers. Entre autres, il prend pour acquis les hypothèses suivantes :

- Le sol est plat. L'altitude est donc considérée égale à l'élévation.
- Les changements d'angle et de vitesse de l'avion peuvent se faire instantanément, mais pas les changements d'altitude.
- Une hausse de la vitesse de montée de 100 m/min entraîne une hausse de la puissance des moteurs de 10% (requis du projet).
- Lorsque la puissance moteur est maximale (100%), la vitesse de montée est maximale (800 m/min). Donc à vitesse de croisière (vitesse de montée = 0 m/min), la puissance moteur est 20%.
- Si la vitesse de montée ne change pas, mais que le pilote modifie l'angle d'attaque, la puissance moteur ne change pas.
- Lorsque l'avion est en descente, la puissance moteur reste à 5%.
- La chute libre survient lorsqu'un angle d'attaque au-dessus de l'angle critique (15°) ou en-dessous de -15° est manuellement entré par le pilote. Si l'altitude entrée ou la vitesse entrée est maximale, l'angle d'attaque est inférieur à l'angle critique.
- L'entrée d'une nouvelle altitude, d'une nouvelle vitesse ou d'un nouvel angle d'attaque par le pilote annule la chute libre et l'avion vole normalement à nouveau.
- Les 3 états possibles de l'avion sont : *AU_SOL* (altitude = 0), *VOL_DE_CROISIERE* (altitude \neq 0 et vitesse montée = 0) et *CHANGEMENT_ALT* (vitesse montée \neq 0). En chute libre, l'état est *CHANGEMENT_ALT*.

À l'état initial du système, l'état est *AU_SOL*, l'altitude est de 0 pieds, la vitesse de montée est de 0 m/min, l'angle d'attaque est de 0 degrés et la puissance des moteurs est à 0%. À partir du moment où le système est initialisé, plusieurs événements peuvent survenir.

Tableau 2. Fonctions de l'algorithme avionique

Événement	Conséquence (étapes réalisées par la fonction appelée)
Une nouvelle altitude est envoyée par le pilote (initié par l'agrégateur)	<ol style="list-style-type: none"> 1. La chute libre est annulée si elle est en cours 2. Une nouvelle cible en altitude est initialisée 3. L'état devient <i>CHANGEMENT_ALT</i> 4. La vitesse de montée, la puissance des moteurs et l'angle d'attaque sont mis à jour*
Un nouvel angle d'attaque est envoyé par le pilote (initié par l'agrégateur)	<ol style="list-style-type: none"> 1. La chute libre est annulée si elle est en cours 2. L'angle d'attaque de l'aéronef devient instantanément l'angle en entrée 3. L'état devient <i>CHANGEMENT_ALT</i> 4. La cible en altitude est retirée
Une nouvelle vitesse de montée est envoyée par le pilote (initié par l'agrégateur)	<ol style="list-style-type: none"> 1. La chute libre est annulée si elle est en cours 2. La vitesse de l'aéronef devient instantanément celle en entrée 3. L'état devient <i>CHANGEMENT_ALT</i> 4. La puissance moteur est mise à jour : $P(\%) = 20\% + \text{vitesse} / (10\text{m/min de plus par } \% \text{ de } P)$ 5. La puissance moteurs est mise à 5% si on descend 6. La cible en altitude est retirée
Une seconde vient de passer et il faut mettre à jour l'état du système avionique (initié par le calculateur)	<p>Si l'état est <i>CHANGEMENT_ALT</i>,</p> <ol style="list-style-type: none"> 1. Mettre à jour l'altitude en fonction du changement d'altitude fait dans la dernière seconde (tient seulement compte de la vitesse de montée actuelle) 2. Si on dépasse l'angle critique, amorcer une chute libre 3. Si nous sommes en chute libre, diminuer la vitesse de montée selon un taux de 9.81m/s^2 4. Sinon, <ul style="list-style-type: none"> Si l'altitude est minimale (0'), l'état devient <i>AU_SOL</i> et l'angle d'attaque ainsi que la vitesse sont remis à 0 Sinon si l'altitude cible ou maximale est atteinte ou si la vitesse = 0 ou si l'angle = 0, l'état devient <i>VOL_DE_CROISIERE</i> et l'angle ainsi que la vitesse sont remis à 0 Sinon si une altitude cible est initialisée, la vitesse de montée, la puissance des moteurs et l'angle d'attaque sont mis à jour*

*voir le calcul ci-dessous, qui décrit selon quelle courbe la vitesse et l'angle d'attaque diminuent à l'approche de la cible

Lorsqu'une cible en altitude est initialisée, une fonction dans notre système avionique fait le calcul suivant pour s'assurer que la vitesse et l'angle diminuent à l'approche de la cible :

$$\Delta \text{altitude}(pi) = | \text{cible}(pi) - \text{altitude actuelle}(pi) |$$

$$\text{direction} = 1 \text{ si } \text{cible} > \text{altitude actuelle}, -1 \text{ sinon}$$

*si $\Delta altitude > 5000'$: $vitesse\ de\ mont\acute{e}e = direction * v_{max} = direction * 800\ m/min$*

*sinon : $vitesse\ de\ mont\acute{e}e\ (m/min) = direction * 93.93\ log(\Delta altitude)$*

$puissance\ moteurs\ (\%) = puissance\ de\ base\ (\%) + vitesse/(100m/min \div 10\%) = 20 + vitesse/10$

si $puissance\ moteurs < 5\%$: $puissance\ moteurs = 5\%$

*$angle\ d'attaque\ (^{\circ}) = direction * e^{\frac{\Delta altitude}{14770}} - 1$*

3. Tâches critiques

Une des fonctions les plus critiques de notre projet est la mise à jour des valeurs dans le calculateur. Dans notre cas, nous avons supposé que ces valeurs devaient se mettre à jour à toutes les secondes. Ainsi, un rafraîchissement de l'altitude, de la vitesse de montée, de la puissance des moteurs et de l'angle d'attaque dans le calculateur s'effectue en assumant un délai de 1000 ms entre chaque rafraîchissement. Cette tâche est considérée critique, puisque des mises à jour fréquentes des valeurs d'altitude, de vitesse et de puissance des moteurs sur l'application GUI sont nécessaires pour que l'utilisateur entre de nouvelles commandes au bon moment. Par exemple, si l'utilisateur planifie modifier l'angle d'attaque manuellement une fois l'avion à 5000 pieds, si le système ne se rafraîchit pas suffisamment rapidement, le pilote pourrait voir l'altitude passer de 4950 pieds à 5050 pieds et ainsi envoyer sa commande trop tard. Si le rafraîchissement se fait trop rapidement, les valeurs d'altitude, de vitesse et de puissance moteur pourraient changer trop rapidement pour que le pilote puisse adéquatement les lire, c'est pourquoi nous avons décidé que la période ne serait pas plus courte qu'une seconde.

Pour implémenter cette fonction, nous avons utilisé une minuterie (*timer*) de la librairie POSIX pour déclencher une interruption à toutes les secondes. Le *callback* appelle la mise à jour du calculateur telle que définie dans le tableau 2. Avant de se conclure, la routine d'interruption demande au *socket* d'envoyer l'état du système avionique mis à jour vers l'agrégateur.

Une autre tâche critique que nous avons développée est la transmission des paquets ARINC429. Chaque paquet ARINC429 envoyé par l'entremise du *socket* côté transmetteur est enregistré dans la FIFO de messages du *socket* côté receveur. Autant dans l'agrégateur que dans le calculateur, le programme lit la FIFO de réception de manière périodique. La présence d'une FIFO (*mailbox*) permet d'éviter qu'un nouveau message envoyé n'écrase le précédent s'il n'a pas encore été traité. Elle permet aussi d'éviter de devoir développer des routines d'interruptions asynchrones qui pourraient nuire au déterminisme de l'application.


L'encodage et le décodage des paquets ARINC429 se fait dans des fonctions qui se retrouvent dans chaque programme pour traduire les informations dans le format désiré, que ce soit en des nombres bruts pour le calculateur, des chaînes de caractères à afficher sur l'interface-utilisateur, ou en paquet ARINC429 pour la transmission. L'encodage et le décodage des paquets doit se faire de façon exacte : une erreur de codage pourrait entraîner des comportements erronés dans le système. Nous considérons donc le codage comme étant une tâche critique également.

4. Plan de contingence

Durant la réalisation d'un projet, plusieurs imprévus peuvent survenir et ceux-ci peuvent bien entendu compromettre la qualité du travail soumis. C'est pourquoi il est important d'envisager des solutions pour ces problèmes dès le début du projet. Dans notre cas, le plan de contingence suivant a été mis sur pied.

Tableau 3. Plan de contingence pour le projet

Problème	Sévérité	Solution
Les applications ne sont pas capables de communiquer entre elles	7	Créer une application avec deux fils d'exécution (<i>threads</i>) qui se communiquent à l'aide de fichiers.
Les technologies choisies sont trop complexes à utiliser ou à mettre en commun	8	Identifier les embûches critiques et chercher de nouvelles technologies qui permettent de les mitiger. En cas de manque de temps: laisser tomber AFDX.
Notre plan de développement préliminaire a des erreurs	4	Concevoir une nouvelle architecture. En cas de manque de temps: laisser tomber AFDX.
Perte d'un membre de l'équipe	10	Discuter avec le chargé s'il y a des accommodements possibles.
L'algorithme de fonction avionique développé est devenu trop complexe ou ne fonctionne pas	5	Changer pour un algorithme plus simple, même s'il ne respecte pas toutes les exigences.
Confusion par rapport au requis	5	Demander des clarifications au chargé ou à des collègues.
Il manque du temps pour compléter le projet	6	Laisser AFDX de côté et se concentrer sur les fonctions critiques.

 : Problème rencontré lors du projet

 : Problème non rencontré lors du projet

Commentaires sur le projet

Arthur Van Betsbrugge: Généralement, l'idée derrière le projet est bonne. Nous pouvons mettre en pratique des notions que nous avons apprises dans un format assez flexible. Cette flexibilité nous a permis de choisir des outils et implémentations avec lesquels nous sommes familiers. Cependant, le document est un peu confus. Personnellement, je trouve qu'il aurait été bien d'avoir un grand tableau avec les phrases formelles, les contraintes et les formules afin de mieux centraliser l'information, puisqu'il était un peu difficile de trouver des informations qui pouvaient se trouver dans l'énoncé, la première section de contraintes, la deuxième section de contrainte, dans la section de phrases formelles, ou d'autre part. Aussi, il y a des détails qui nous ont porté à confusion dans le document. Par exemple, la puissance des moteurs est définie comme étant fournie de l'agrégateur au calculateur, alors que les valeurs que l'utilisateur peut entrer au système sont semblablement l'altitude, le taux de montée et l'angle d'attaque. Il n'était donc pas clair si c'était à l'agrégateur de calculer la conversion entre le taux de montée et la puissance des moteurs, puisque ce genre de calculs devrait normalement être la responsabilité du calculateur. En bref, le projet était assez stimulant et je garderais le format. J'essaierais juste de clarifier un peu le document d'énoncé de projet.

Katherine Zamudio-Turcotte: Je suis entièrement d'accord avec ce que mon coéquipier a écrit. J'ajouterais également que ce serait bien si le projet était juste un peu moins libre. J'aime beaucoup l'idée de laisser aux étudiants le choix des langages et des technologies qu'ils préfèrent, mais ce serait aussi super intéressant si, par exemple, on donnait des formules comme celle de la force de portance dans l'énoncé, ou bien celle du coefficient de portance en fonction de l'angle d'attaque. Cela pourrait permettre d'enseigner aux étudiants, qui sont issus du génie électrique et informatique en grande partie, les principes de l'avionique qui sont pertinents à comprendre en informatique embarquée. Aussi, plutôt que d'utiliser AFDX, il aurait été plutôt intéressant d'utiliser CAN bus ou un autre protocole plutôt simple. Un tutoriel pour utiliser un système d'exploitation temps réel comme FreeRTOS aurait aussi pu guider les étudiants vers une implémentation plus fidèle aux systèmes embarqués.