République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la recherche scientifique

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Projet de fin d'études

Pour l'obtention du titre

d'Ingénieur d'Etat en Electronique

THEME :

CONCEPTION D'UN SIMULATEUR DE VOL

MATERIEL ZLIN 142

Présenté par :

Mr. TALABOULMA Walid

Promotion Septembre 2012

Ecole Nationale Polytechnique

10, Avenue Hacen Badi, El-Harrach, Alger

ملخص

الهدف من هذا المشروع هو إعادة تشكيل لوحة القيادة لطائرة من نوع **ZLIN-142** بإيصال شريحة إلكترونية MBED مزودة بحاسوب ARM بجهاز مقلد للطيران MICROSOFT FLIGHT SIMULATOR عن طريق API-SimConnect
الغرض من ذلك التقرب قدر الإمكان من وكن الطيار و إعادة المحيط البصري لتجربة طيران حقيقية

الكلمات المفاتيح : جهاز مقلد للطيران ,Flight simulator X Mbed ARM Zlin 142
API, Sim- Connect.

Résumé

L'objectif de ce projet, est la reproduction du tableau de bord d'un aéronef model Zlin 142 en interfaçant une carte microcontroleur Mbed muni d'un processeur ARM avec le simulateur de vol de Microsoft Flight simulator X par le biais de son API SimConnect, le but étant de s'approcher le plus possible des commandes diponibles au sein du cockpit et de restituer l'environnement visuel d'une réelle experience de vol.

Mots clés : Simulateur de Vol, Zlin 142, Mbed ARM, Flight simulator X, API, Sim-Connect.

Abstract

the main subject of this project, is the reproduction of a Zlin 142 plane's dashboard by interfacing a Mbed micontroler board featured with an ARM core processor with a flying simulator (MICROSOFT Flight Simulator X) through it's native API : Sim-connect. Our goal is to emulate the cockpit and provide a visual experience of a real cruise in the air.

Keywords: Fly Simulator, Zlin 142, Mbed ARM, Flight simulator X, API, Sim- Connect.

Résumé

L'objectif de ce projet, est la reproduction du tableau de bord d'un aéronef model Zlin 142 en interfaçant une carte microcontroleur Mbed muni d'un processeur ARM avec le simulateur de vol de Microsoft Flight simulator X par le biais de son API SimConnect, le but étant de s'approcher le plus possible des commandes diponibles au sein du cockpit et de restituer l'environnement visuel d'une réelle experience de vol.

Mots clés : Simulateur de Vol, Zlin 142, Mbed ARM, Flight simulator X, API, SimConnect.

Design of a Hardware Zlin 142 Flight Simulator

Abstract

the main subject of this project, is the reproduction of a Zlin 142 plane's dashboard by interfacing a Mbed micontroler board featured with an ARM core processor with a flying simulator (MICROSOFT Flight Simulator X) through it's native API : Sim-connect. Our goal is to emulate the cockpit and provide a visual experience of a real cruise in the air.

Keywords: Fly Simulator, Zlin 142, Mbed ARM, Flight simulator X, API, Sim- Connect.

Remerciements

pour faire simple et pour qu'il n'y ai pas de jaloux, je remercie toutes les personnes qui ont participées de près ou de loin à la réussite de ce travail.

Dédicaces

Je dédie ce travail à ma famille
mes parents
ma soeur
mon frère
mes deux grand meres que dieu les garde
mon grand père paix à son ame
à mes amis : Tahar El Hac, Moncef errougi, Sofiane alias pesh, Youcef youyou le végétarien à la retraite, Lateef le Blond, Bilel Boudouma, MAncef bleu. tout le monde un milliard fi khater la7bab ... XD.

Spécial dédicace au chauffeur de Kouss, et aux femmes de ménages et leur bouteille d'esprit de sel.

Table des matières

Table des figures

Introduction et Théorie

« I believe I can Fly, I beleive I cant Touch the Sky »"

(R.kelly)

Chapitre 1

La Simulation de Vol

1.1 Introduction

On peut définir un simulateur de vol comme un système dans lequel un pilote réel est aux commandes d'un avion virtuel dont le comportement est obtenu par simulation. Dans sa version professionnelle, il se présente généralement sous la forme d'une cabine de pilotage, mobile ou non, actionnée par logiciel, ces simulateurs de vol sont largement utilisés par les compagnies aériennes, l'industrie aéronautique militaire et civile pour la formation continue des pilotes (nouveau type d'avions ou d'équipements, situations extrêmes, opérations militaires) et développer de nouveaux avions. Il permet notamment aux pilotes des avions de ligne de s'entraîner et de se former aux situations d'urgence.. Il existe aussi des simulateurs de vol en jeu vidéo, pour lesquels du matériel informatique grand public suffit.

1.2 Des pionniers à aujourd'hui

Compte-tenu de la difficulté du pilotage, on a eu recours assez rapidement à des entraîneurs qui étaient des simulateurs très simplifiés où l'apprenti pilote répétait sur des commandes fictives les manœuvres de base. Dès les débuts de l'aviation on a tenté de restituer les effets aérodynamiques des commandes sur un avion simplifié fixé au sol et placé dans le lit du vent. Un des premiers entraîneurs connu fut le « tonneau Antoinette » construit en France en plusieurs exemplaires par la société de Léon Levavasseur dès 1910 (figure 1.2.1) : il comportait un poste de pilotage monté sur rotule et actionné manuellement, il permettait aux apprentis pilotes de répéter les manoeuvres de base sur des commandes fictives tout en subissant des effets simulés tels que lacet, roulis et tangage (figure 1.2.2).

Le premier vrai simulateur de vol fut vraisemblablement le système mis au point par Edwin Link en 1929 (figure 1.2.3), un fabriquant d'orgues aux Etats-Unis. Il comportait

FIGURE 1.2.1: Le tonneau Antoinette (visible au musée de l'Air et de l'Espace, Le Bourget)

une cabine posée sur un mouvement électro-pneumatique dont les positions répondaient aux commandes du pilote. Un chariot équipé d'un stylet reproduisait sur table traçante le trajet virtuel de l'avion et l'instructeur pouvait donner des ordres à l'élève à l'aide d'un micro. Ce simulateur, très utilisé lors de la Seconde Guerre mondiale essentiellement pour le vol aux instruments, connut diverses évolutions jusque dans les années 1960.

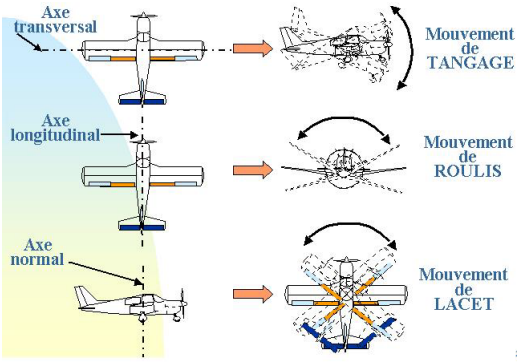


FIGURE 1.2.2: Les mouvements (rotations) d'un avion

La mise en œuvre de modèles de vol sur calculateurs, analogiques d'abord dès les années 1950 puis numériques, a donné aux simulateurs la possibilité de représenter plus fidèlement le comportement d'un aéronef en vol.

Le grand projet Whirlwind du Massachusetts Institute of Technology en 1946 fut de concevoir et mettre au point un calculateur numérique en temps réel nécessaire à un simulateur de vol militaire.

Outre la puissance de calcul souvent insuffisante, la restitution visuelle des premiers

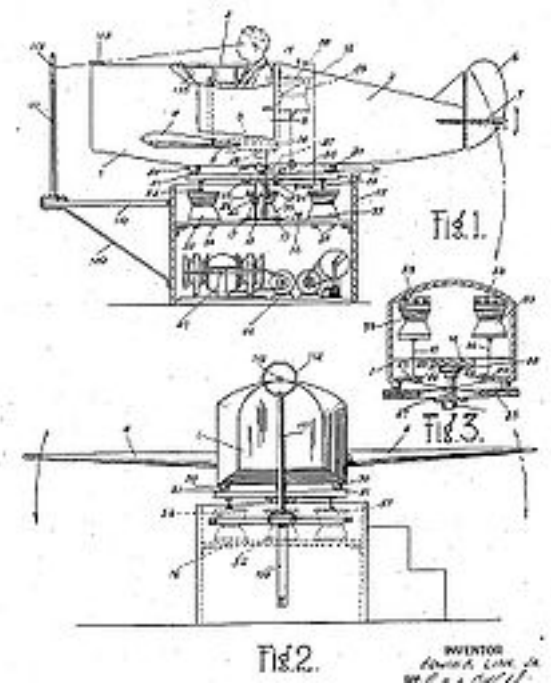


FIGURE 1.2.3: Le brevet d'Edwin Link

simulateurs, limitée aux phases de décollage et d'atterrissage, n'était rendue que par le déplacement d'une caméra vidéo survolant mécaniquement une maquette réelle de terrain de grande dimension. Ce n'est que plus tard, au milieu des années 1970, que commencèrent à apparaitre des images de synthèse encore très schématiques mais qui permettaient un rendu ponctuel très précis des feux de piste, de nuit. Au début des années 1980, la représentation en trois dimensions de surfaces avec ombrage avait une allure plus réaliste mais il a fallu attendre les années 1990 pour voir dans les simulateurs des images de synthèse 3D texturées.

1.3 Composants d'un simulateur de vol

Les deux principaux composants d'un simulateur de vol sont le cœur logiciel d'une part, et un ensemble matériel nécessaire au pilote pour piloter le simulateur et percevoir les réactions de l'avion virtuel (simulé) (figure 1.3.1).

1.3.1 Partie logicielle

Par ensemble logiciel, on entend le modèle de vol de l'avion simulé bien entendu, mais aussi les modèles de comportement des équipements (les capteurs et afficheurs ont leur propre dynamique), les modèles de l'environnement (typiquement l'atmosphère et

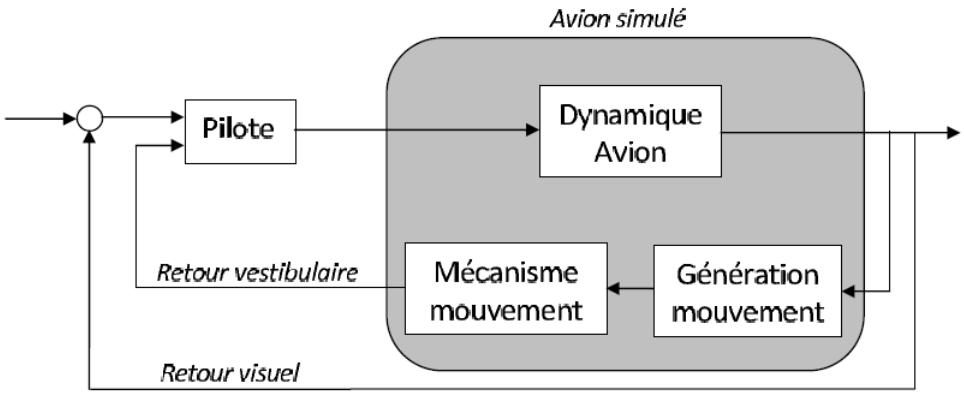


FIGURE 1.3.1: Schéma-bloc pilote + avion simulé

son évolution), les bases de données nécessaires aussi bien en vol à vue qu'en vol aux instruments, les modèles nécessaires aux scénario comme le trafic, les échanges radio, les incidents, etc.

Les solutions logicielles sont maintenant nombreuses sur le marché : l'expansion des jeux vidéo contribue largement au développement de simulateurs de plus en plus réalistes. Lorsque ces simulateurs sont ouverts pour permettre d'ajouter ses propres créations, beaucoup d'utilisateurs deviennent créateur d'avion en réalisant, à l'aide de logiciels de modélisation 3D, des modèles d'avions militaires ou d'avions de ligne. D'autres créent des versions personnelles et virtuelles basées sur de vrais avions de ligne, ou même des décorations virtuelles de compagnies fictives comme Virtual Delta, Viasa Virtual, et d'autres, que l'on peut trouver sur Internet. A noter qu'en plus du pilotage virtuel, beaucoup d'utilisateurs découvrent le trafic aérien en ligne : lorsque les pilotes et des contrôleurs du trafic aérien virtuels jouent ensemble en temps réel pour simuler un véritable trafic.

1.3.2 Partie matérielle

L'ensemble matériel comporte les dispositifs d'entrée des commandes avec éventuellement des retours d'effort, les dispositifs de visualisation des instruments et de l'environnement extérieur, les dispositifs de restitution sonore, les dispositifs de restitution de mouvement (ou proprioceptifs). Que le simulateur soit professionnel ou non, on retrouvera toujours plus ou moins cette structure. On dira que le simulateur est à base mobile si le dispositif de restitution des mouvements existe, sinon on parlera de simulateur à base fixe.

Le système de visualisation est constitué d'un générateur d'images de synthèse, qui est un calculateur spécialisé, très rapide. Il possède en mémoire une base de données (le paysage) et il calcule l'image qui serait vue par le pilote compte tenu de la position

géographique du mobile, de son orientation, de son altitude, de l'heure, des conditions atmosphériques, etc. Ce calcul s'effectue plusieurs dizaines de fois par seconde (en général 60 fois). L'image produite par le calculateur doit être présentée selon un champ visuel conforme à celui existant dans la réalité. Dans un avion de ligne, par exemple, le champ visuel est limité par les fenêtres de la cabine de pilotage, soit environ 45°et 220° dans les directions respectivement verticales et horizontales. Les systèmes de visualisation simulée peuvent tout à fait couvrir ce champ visuel. De plus, les images sont projetées à travers un système optique qui les renvoie à l'infini pour que les yeux de l'observateur puissent s'accommoder à l'infini, comme dans la réalité.

Les bruits sont, eux aussi, synthétisés et des enceintes acoustiques sont placées aux endroits adéquats autour de la cabine pour être perçus là où ils le seraient en réalité.

Pour piloter, le pilote doit percevoir les mouvements de l'avion dans l'espace : cette perception se fait par l'observation visuelle à l'extérieur du cockpit (ligne d'horizon, pente, dimension des objets au sol), à l'intérieur par la lecture des instruments de vol mais aussi grâce aux sensations des capteurs humains comme ceux de l'oreille interne qui fonctionnent comme des accéléromètres. Le simulateur idéal doit donc restituer ces deux aspects ; la solution actuelle la plus utilisée est l'hexapode ou plateforme de Stewart (figure 5).

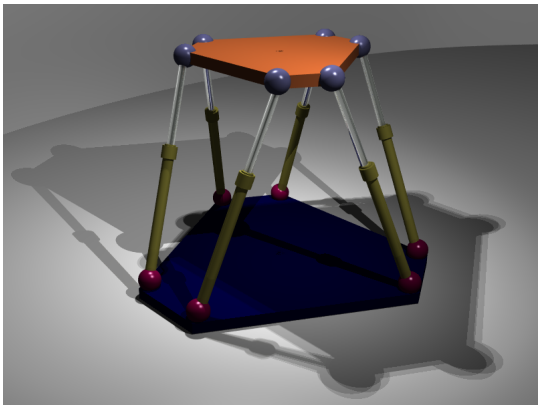


FIGURE 1.3.2: Plateforme de Stewart (6 degrés de liberté)

Les ordres envoyés aux vérins ne cherchent pas à reproduire l'attitude du mobile simulé, mais à recréer les accélérations ressenties à bord. Ces accélérations sont dues aussi bien aux actions de l'équipage sur les commandes de vol qu'à la turbulence atmosphérique, aux vibrations des moteurs et à celles résultantes de l'action des freins, etc.

1.4 Les différents types de simulateurs

On distingue deux principaux types de simulateurs, de par la complexité, l'avancement technologique et de fait le cout, et le publique visé.

1.4.1 Les simulateurs de vol professionnels

Ils sont utilisés soit pour des études, soit pour des entraînements, leur constitution dépends principalement de leur utilisation.

1.4.1.1 Simulateurs d'études

Les simulateurs de vol d'études sont utilisés pour le développement de nouveaux avions, leur utilisation peut se faire durant toutes les étapes de développement. Pour un même avion, différents simulateurs peuvent être utilisés, pour chaque étape de développement on utilise le simulateur approprié et on néglige les autres paramètres, par exemple, lors de la conception du calculateur de bord on pourra se passer du simulateur de la visualisation externe.

1.4.1.2 Simulateur de formation ou d'entraînement

Le nom de simulation de formation peut induire en erreur le lecteur, en effet ce genre de simulateur ne sert pas à l'apprentissage du pilote mais plutôt pour faire des entraînements très spécifiques destinés à un personnel déjà pilote. L'objectif principal de ce simulateur est l'acquisition de nouvelles connaissances pour les nouveaux avions ainsi que pour les procédures à suivre lors de certaines phases de vol bien précises.

1.4.1.3 Simulateur d'enquête liée aux accidents

Les enquêteurs cherchent à reconstituer l'enchaînement des faits ayant conduit à un accident grâce aux enregistrements des boîtes noires. Le simulateur permet une approche qualitative de la situation à laquelle l'équipage s'est trouvé confronté et d'en tirer des enseignements pour l'amélioration éventuelle des interfaces ou des procédures.

1.4.2 Les simulateurs de vol grand public

1.4.2.1 Simulateurs pour ordinateur personnel

Les simulateurs de vol ont été parmi les premiers types de logiciels de simulation développés pour les ordinateurs individuels. Un type populaire de simulateur de vol est le simulateur de vol de combat, comprenant les séries tel que Aces High ou Fighter Ace. Au début des années 2000, les simulateurs de vol grand public, conçus principalement pour se divertir, deviennent si réalistes qu'après les attentats du 11 septembre 2001, quelques journalistes et experts ont spéculé sur le fait qu'il était possible pour les pirates de l'air d'avoir acquis assez de connaissance dans le maniement d'un avion de ligne en utilisant Microsoft Flight Simulator. Lorsque ces simulateurs sont « ouverts » pour permettre d'ajouter ses propres créations, beaucoup d'utilisateurs deviennent « créateur d'avion » en réalisant, à l'aide de logiciels de modélisation 3D, des modèles d'avions

militaires ou d'avions de ligne. D'autres créent des versions personnelles et virtuelles basées sur de vrais avions de ligne. En plus du pilotage virtuel, beaucoup d'utilisateurs découvrent le « trafic aérien en ligne », lorsque les pilotes et des contrôleurs du trafic aérien virtuels jouent ensemble en temps réel pour simuler un véritable trafic.

Parmi les simulateurs les plus connus pour ordinateurs personnels, on peut citer :

- *Microsoft Flight Simulator*, le simulateur de vol grand public le plus connu.
- *X-Plane*, un simulateur de vol civil original pouvant être utilisé pour la formation et le seul à être certifié par la FAA.
- *FlightGear*, un simulateur de vol sous GPL , donc entièrement gratuit ; avec tous les aéroports du monde, des modèles d'avions très nombreux et beaucoup de ressources en général.

1.4.2.2 Simulateurs pour console de jeu

Beaucoup plus rares sont les simulateurs de vol disponibles pour les différentes consoles de jeu. Le plus connu est *Pilotwings* disponible pour Super Nintendo et sa suite Pilotwings 64 pour la console Nintendo 64. En raison de la difficulté à représenter un environnement complexe et les limitations de traitement d'un tel système informatique, les simulateurs de vol pour consoles tendent jusqu'à présent à être simplistes et à conserver, au niveau des sensations de pilotage, un côté « arcade », mais néanmoins, ils visent à recréer le plus fidèlement possible un vol.

1.4.2.3 Simulateur de vol spatial

L'espace étant un prolongement logique de l'espace aérien, les simulateurs de vol spatial peuvent être considérés comme un prolongement du genre. Ces deux genres de simulateurs se rejoignent parfois, car quelques simulateurs de vol comme *X-Plane* comportent des extensions pour utiliser des vaisseaux spatiaux. Cependant les simulateurs de vol spatial mettent plus l'accent sur le réalisme du rendu de la haute atmosphère et des voyages interplanétaires. Comme simulateur de vol spatial pour ordinateurs personnels, on peut citer : *Microsoft Space Simulator*

1.4.2.4 Simulateur de vol à domicile

Pour une mise en situation plus proche du réel, certains amateurs n'hésitent pas à construire eux-mêmes un poste de pilotage similaire à celui d'un avion réel. Pour cela on trouve dans le commerce des panneaux complets d'instruments fonctionnels ayant l'aspect des instruments réels. De tels simulateurs utilisent généralement un des logiciels présenté précédemment. Ces simulateurs « privés » permettent à leur développeur d'approcher et de toucher de plus près le pilotage de simulateurs professionnels.

1.5 Conclusion

Dans ce premier chapitre on a présenté beaucous de notion théoriques sur la simulation de vol, l'historique nous aide à voir l'evolution dans le domaine et à mieux imaginer ce qu'on pourrait y apporter, la schéma des composants standard d'un simulateur de vol, va nous permettre par la suite de structurer notre travail autour d'une logique bien définie, et puisque un simulateur de vol n'est rien sans ... Avion, passons au chapitre suivant pour présenter celui qui a été choisis pour donner vie à notre simulateur.

Points clés

Positionnement

- ❏ historique
- ❏ composants d'un simulateur
- ❏ les différents types de simulateurs

Chapitre 2

Le Zlin 142

2.1 Introduction

Le Choix d'un avion comme objet de ce travail devait être judicieux et bien pensé, il était clair depuis le début que prendre un avion de ligne moderne tel un *Boeing 747* aurait été impossible dans le temps qui nous était imparti, on devait se tourner vers de petits appareils certes moins impressionnant, mais tout aussi intéressant à étudier et bien plus accessible que les gros porteurs qui naissent dans les usines d'*Airbus*.

L'un des premiers aéronefs auquel on pense est sans doute le *Cessna* rendu célèbre par les studios de Hollywood, c'est aussi l'avion qui sert à entrainer les débutants et souvent les accompagne dans leur baptême de l'air. mais malgré cela on a penché pour un avion un peu moins connu mais qui a l'avantage d'être fabriqué sous licence en Algérie à la base de *Tafaraoui à Oran* sous l'appellation *Fernass 142* qui est en réalité une copie légèrement modifiée du *Zlin 142*

2.2 Histoire du constructeur aéronautique

Zlin (Zlinská Letecká Akciová Společnost) était une société tchécoslovaque célèbre pour ses avions de compétition et de voltige très fiables et aux performances particulièrement élevées. Les débuts remontent à 1933, lorsque des enthousiastes de l'aviation et des pilotes ont commencé à dessiner et fabriquer des planeurs sous la direction de Jan Kryspin dans des ateliers de la ville de Zlín. En 1934, le fabricant de chaussures Bata également implanté à Zlín créa la société de construction aéronautique Bata A.S. Zlin. Frantisek Majer, qui rejoignit l'équipe cette même année créa le premier véritable planeur de l'entreprise baptisé Z-X. Le 8 juillet 1935, la raison sociale de la société fut modifiée en Zlinská Letecká Akciová Společnost et Bata construisit 10 km plus loin à Otrokovice une nouvelle usine plus grande dans laquelle déménagea la société de construction aéronautique qui conserva le nom « Zlin ». C'est là que furent construits jusqu'en 1939 les

premiers avions à moteur comme le Z-XII ou le Z-XIII, cinq modèles au total. Au cours de l'occupation allemande, l'usine ne développa pas d'avions mais fabriqua des avions-écoles Klemm Kl 35 et des Bücker Bü 181 Bestmann, rebaptisés après la guerre en Zlin Z-181. Après la Seconde Guerre mondiale, les usines Zlin furent rebaptisées *Moravan*. Au cours des années suivantes, des modèles célèbres y furent créés comme la série des « Trener » (Trainer) Z-26 (en) et Z-126 (en) ainsi que leurs successeurs Z-326 (en), Z-526 et Z-726 (en). D'autres appareils furent construits comme l'avion-école et de tourisme Z-42 et l'avion de voltige aérienne Z-50LS.

2.3 La Naissance du Z-142

L'avion fut fabriqué en tant que successeur et remplaçant de la célèbre serie des Zlin Trener. *Moravan* a développé une nouvelle famille d'aéronefs léger, dotés d'un agencement de siège type cote à cote, parmi lesquels un trainer a deux places le Zlin Z 42, et un autre à quatre places, le Zlin Z 43.

Figure 2.3.1: Un Zlin Z-42

Le Z 42 (figure 2.3.1) a pris son premier envol le 17 Octobre 1967, décrochant la certificat de mérite le 7 Septembre 1970. Le section centrale du fuselage de l'avion est faite a base de tue en acier soudés, recouvert par des plaques en métal et des panneaux en fibre de verre. La queue est une monocoque. Le train d'atterrissage monté en tricycle est fixe, avec une roue avant mobile pour les manœuvres de garage .

Conçue pour des fins acrobatiques, l'avion fut équipé avec un ensemble complet d'injection inversé, augmentant la duré des vols la tête en bas. le Z 42 vol grâce à un moteur

six-cylindre inversé de type Walter développant une puissance de 180 hp.

Le développement du Zlin Z-142 (figure 2.3.2) se poursuit avec l'introduction de quelques modifications comme un cadre a deux places inspiré du grand frère Z 42, et un moteur Walter suralimenté refroidis à l'air, et ainsi bien plus puissant car pouvant atteindre les 210 hp. l'entrée sur la marché de ce prototype s'est faite en Décembre de l'année 1978.

FIGURE 2.3.2: Le Zlin Z-142

2.4 Tableau de chasse

Deux Z-142 ont pris part dans une mission de bombardement, engagé par les *forces de libération des Tigres Tamouls* contre les bases aériennes du Sri Lanka en 2007. En Octobre de 2008 des Zlins ont été utilisés pour mener une attaque contre des bases militaires de l'armée du Sri Lanka, et une centrale électrique proche de la ville de Colombo, Sri Lanka.

2.5 Spécifications techniques

notez que les données présentées ci-dessous peuvent varier selon les variantes et les modifications apportées par certains organismes qui détiennent des licences d'exploitation, mais dans de faible proportions.

2.5.1 Caractéristiques générales

- Équipage : 1
- Capacité : 1 passager ou un étudiant
- Longueur : 7.07 m
- Envergure : 9.11 m
- Hauteur : 2.69 m
- Surface des ailes : 13.15 m2
- Poids à vide : 600 kg
- Poids limite de décollage : 920 kg
- Propulsion : Moteur à 6 cylindres inversé 210hp

2.5.2 Performances

- Vitesse à ne jamais dépasser : 315 km/h
- Vitesse maximum : 230 km/h
- Vitesse de croisière : 200 km/h
- Portée : 650 km
- Portée étendue : 1200 km/h (réservoirs de carburants additionnel sur les ailes)
- Vitesse d'ascension : 5.0 m/s

2.6 Cockpit et Commandes de bases

Le centre de pilotage ou *cockpit* d'un Zlin 142 est un biplace de conception simple comportant les instruments de mesures de bases nécessaires au vols et différentes commandes de pilotages présentes sur la majorité des aéronefs (figure 2.6.1).

On distingue :

- deux parties latérales siège des instruments de mesure, une principale à gauche et l'autre pour le copilote ou l'étudiant, les mêmes mesures sont copiées et affichées en double, sauf cas particulier.
- Une partie centrale ou toutes les commandes sont rassemblées et partagées par les deux occupants, une sur le tableau de bords, et une entre les deux places assises.
- Des commandes jumelées similaires a ceux qu'on trouve dans une voiture d'autoécole, a savoir : les pédales pour le contrôle du gouvernail, et le manche pour les ailerons et toutes manœuvres de changement de cap et/ou d'altitude.

FIGURE 2.6.1: Cockpit d'un Zlin Z-142 (vue globale)

2.6.1 Panneau centrale

2.6.1.1 Le Haut

voir (figure 2.6.2)
- 1 : voyants lumineux et signaux d'urgence ou d'alerte (bas niveau de huile , perte d'altitude, température élevée ...)
- 2 : simple horloge de bords.
- 3 : G mètre, pour connaitre les accélération subis par l'appareil.
- 4 : radio indispensable pour les communications avec les tours de contrôle et les envois de messages d'appel au secours en cas de problème.

2.6.1.2 Commandes du Moteur

voir (figure 2.6.3)
- 1 : starter pour démarrer le moteur de l'avion, réglable sur quatre position selon les besoins en puissance.

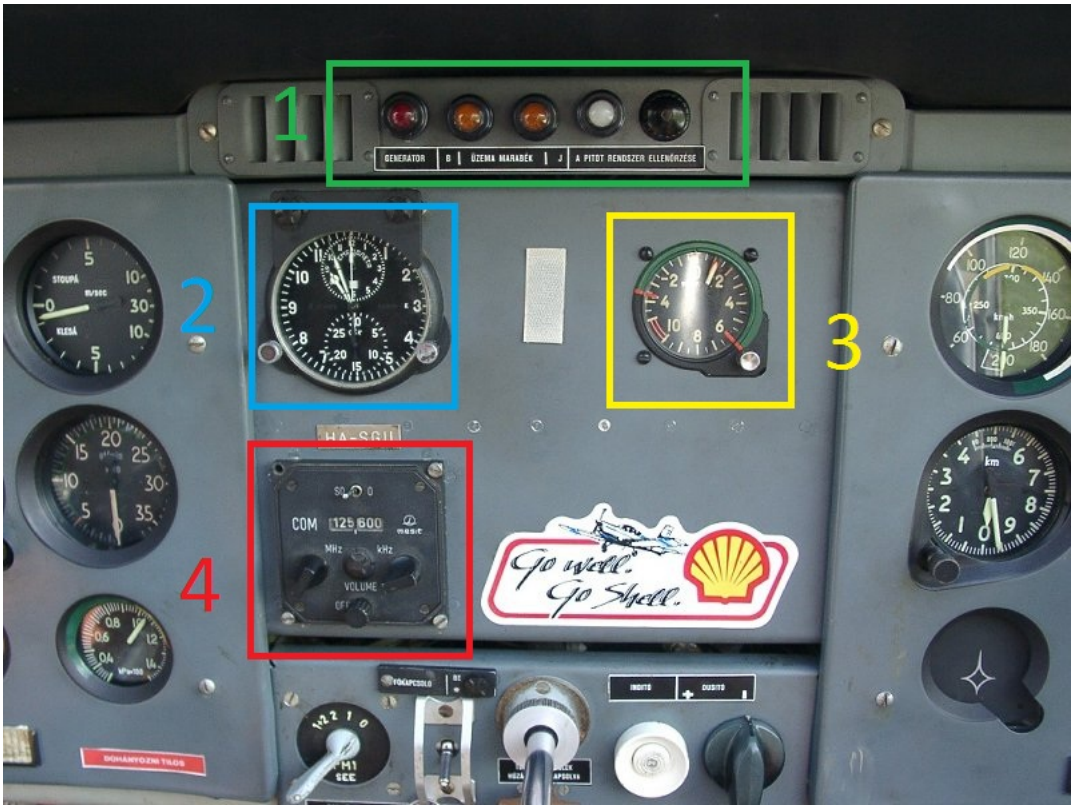


FIGURE 2.6.2: panneau central et ses instruments

– 2 : commande des gaz *(throttle),* agis sur la poussé de l'avion et affecte la vitesse de rotation de l'hélice.
– 3 : réglage de dosage pour changer les composition du mélange carburant air, selon les besoins en puissance, et les performances du moteur.

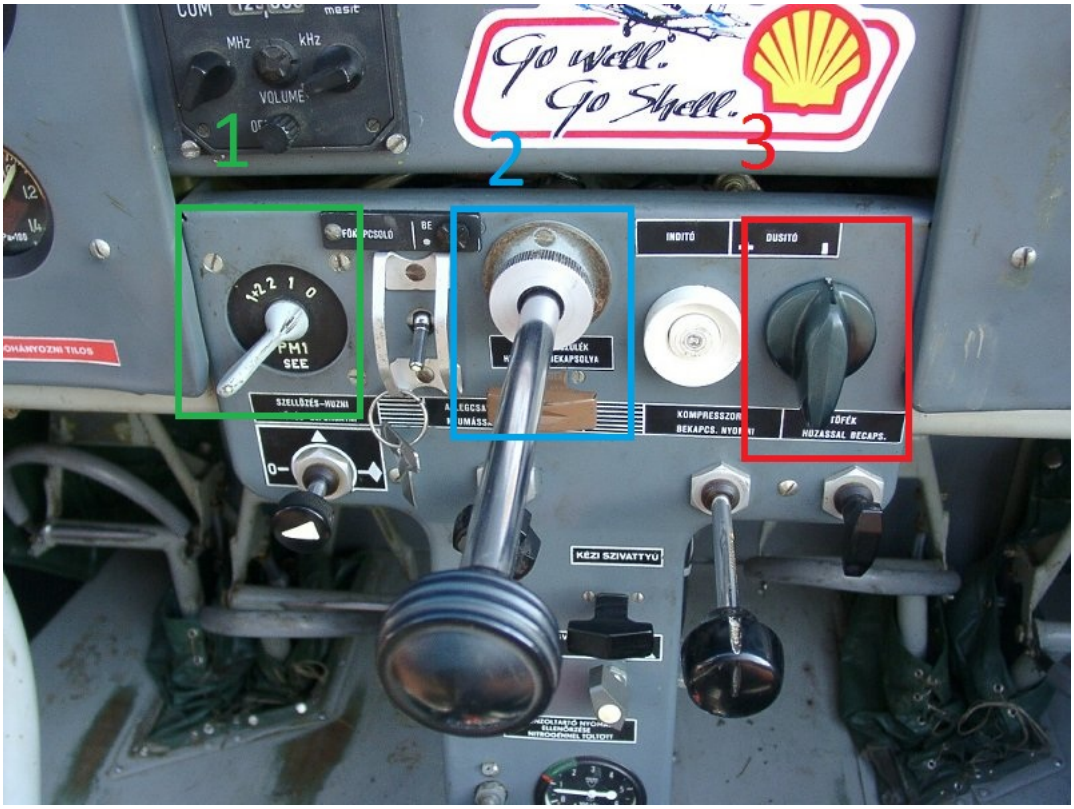


FIGURE 2.6.3: commandes du moteur d'un Zlin

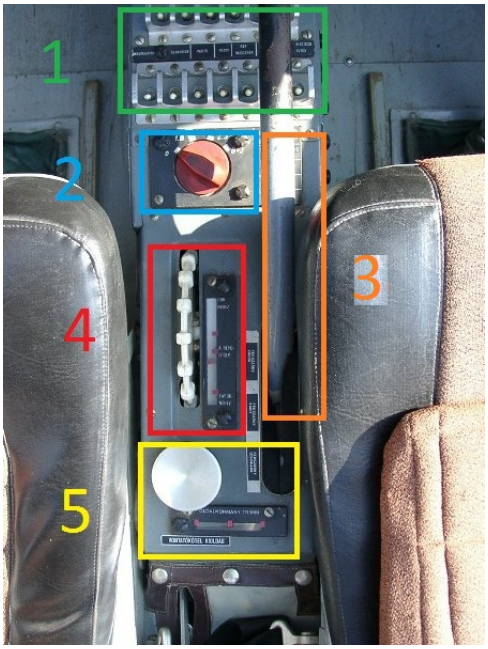


FIGURE 2.6.4: switch electriques et commandes de trim et des volets

2.6.1.3 Entre les pilotes

voir (figure 2.6.4)
- 1 : interrupteur pour le contrôle des unités électriques des instruments électroniques et capteurs et de l'éclairage intérieur et extérieur de l'appareil.
- 2 : sélecteur de réservoirs de carburants a quatre positions.
- 3 : levier des volets *(flaps)* à trois positions, totalement rétracté, totalement sortis, et medium.
- 4 : commande de trim d'aileron pour le contrôle de *pitch*
- 5 : trim de gouvernail pour les rotations de types *yaw*

2.7 Instruments de bords

Les instruments de bord servent à présenter à l'équipage, en particulier au pilote, toutes les informations utiles au maintien en vol de l'aéronef, à la navigation, aux communications avec les infrastructures de la gestion du trafic aérien.

Les instruments de bord sont regroupés selon leur fonction, éventuellement à proximité des commandes correspondantes :
- pilotage : horizon artificiel, anémomètre, altimètre, variomètre, etc.
- navigation : compas, ILS, VOR, GPS, etc.
- gestion des groupes motopropulseurs : tachymètre, température et pression, etc.
- gestion des télécommunications : radio, système d'intercommunication de bord, etc.
- gestion des servitudes : consommation de carburant, tension et intensité électrique, etc.
- accomplissement de la mission : instruments spécialisés.

Selon le type d'aéronef et le nombre de membres d'équipage les instruments sont regroupés sur des tableaux et, pour le pilote, sur le tableau de bord situé devant lui. Les quatre instruments de base sont toujours disposés de la même façon (en configuration de T basique) (figure 2.7.1) : l'horizon artificiel au centre, l'anémomètre à gauche, l'altimètre à droite, le gyro directionnel ou plateau de route en dessous. Cette disposition permet d'optimiser le circuit visuel au cours du vol. La disposition des autres instruments est variable mais respecte certains standards.

Sur les aéronefs les plus récents les instruments sont remplacés par des écrans rassemblant toutes les informations du T de base sur une seule surface de visualisation, les instruments conventionnels ne sont conservés qu'à titre de secours pour pallier une éventuelle défaillance des systèmes électroniques. Les écrans sont le plus souvent multifonctions, c'est-à-dire qu'ils sont prévus pour afficher l'ensemble des informations nécessaires à une phase de vol au gré du pilote. Originellement les écrans reprenaient les vues classiques des instruments analogiques. Ils sont progressivement remplacés par des visuels regroupant les informations selon des standards ergonomiques.

FIGURE 2.7.1: Les 4 instruments de base en T complétés par, en bas à gauche l'indicateur de virage, en bas à droite le variomètre

2.7.1 Problématique générale

Les instruments de bord mesurent une donnée utile pour le pilote. Comme tout instrument de mesure ils sont constitués d'un détecteur, d'un système de transformation et d'un système d'affichage.

Le détecteur est un élément dont une caractéristique physique varie proportionnellement avec le phénomène à mesurer. Par exemple une roue à aubes placée dans un tuyau du circuit de carburant tournera d'autant plus vite que le flux est important.

Le système de transformation change la valeur physique mesurée en une autre grandeur physique qui permettra d'actionner le système d'affichage. En reprenant l'exemple précédent, la roue à aubes actionnera un mini-générateur électrique dont la tension sera proportionnelle à la vitesse de rotation.

Le système d'affichage transforme cette dernière valeur en un déplacement mécanique, la rotation d'une aiguille par exemple, qui sera lisible par le pilote.

Pour les instruments de base les plus simples les trois éléments ci-dessus peuvent être inclus dans un boîtier unique. Dans la majorité des cas le détecteur est situé en dehors du poste de pilotage et l'information est transmise à l'afficheur sous la forme d'une tension électrique. Sur les avions les plus modernes l'afficheur est virtuel ; les informations mesurées sont transmises à l'ordinateur de bord qui élabore un afficheur virtuel sur l'écran situé devant le pilote.

2.7.2 Choix des types d'instrument

La complexification de la liaison entre le détecteur et l'afficheur augmente le risque de panne. C'est pourquoi les appareils les plus modernes conservent toujours des instruments de base de formule classique en secours.

Les avions civils peuvent utiliser des instruments mesurant des signaux reçus de l'extérieur, c'est le cas de tous les systèmes de radionavigation. Les avions militaires peuvent

utiliser ces mêmes systèmes mais sont aussi équipés de systèmes autonomes permettant de pallier les risques de compromission des informations reçues.

2.7.3 Erreurs spécifiques de mesure

L'information fournie par un instrument de bord est entachée d'erreurs. Certaines de ces erreurs sont spécifiques à l'environnement aéronautique : densité d'instrument sur le tableau de bord et proximité de générateur ou moteurs électriques entrainant un risque d'interférences, vitesse relative, pression et température de l'air externe, attitude de l'appareil, etc.

2.8 Instruments de pilotage

2.8.1 Altimètre

Un altimètre(figure 2.8.1) est un instrument de mesure permettant de déterminer la hauteur d'un aéronef par rapport à un niveau de référence : le sol, le niveau de la mer (mesure d'altitude) ou une surface isobare.

FIGURE 2.8.1: Cadran d'un altimètre de bord Le calage est affiché dans la petite fenêtre à droite (en pouce de mercure) Les trois aiguilles donnent respectivement des dizaines de milliers, des milliers, des centaines de pieds L'altitude affichée est donc de 14 500 pieds

À bord d'un aéronef, il est nécessaire de connaitre trois hauteurs ou altitudes :

- la hauteur par rapport au sol : en particulier pour la navigation locale et éviter les obstacles artificiels dont les cartes publient l'altitude et la hauteur. En utilisant la pression de l'aérodrome en référence, l'aéronef décolle ou se pose avec l'altimètre indiquant 0.
- l'altitude par rapport au niveau de la mer : pour éviter les obstacles naturels dont les cartes publient l'altitude.
- le niveau de vol : pour éviter les abordages entre aéronefs en utilisant une référence arbitraire identique pour tous et fixée à 1 013 hPa.

Pour obtenir les indications ci-dessus, il faut que l'altimètre soit calé sur la pression correspondante. Ces pressions sont transmises par radio et les pilotes utilisent toujours les codes développés à l'époque du Morse afin d'éviter les ambigüités.

- QNH : pression au niveau de la mer. Permet de mesurer l'altitude.
- QFE : pression au niveau du sol. Permet de mesurer une hauteur. Utilisé près d'un aérodrome, ce calage permet de décoller et d'atterrir avec une indication de 0.
- QNE, ou calage au FL (pour Flight Level, en français Niveau de Vol) : calage utilisé par tous les aéronefs en croisière. La référence étant identique pour tous, elle permet d'éviter les accidents.

2.8.2 Anémomètre

Un anémomètre (figure 2.8.2) est un instrument de mesure permettant de déterminer la vitesse d'un aéronef par rapport à l'air ambiant.

La connaissance de la vitesse air est indispensable pour conserver l'aéronef dans son domaine de vol, donc entre la vitesse minimale permettant sa sustentation et la vitesse maximale où les forces aérodynamiques risquent d'endommager la structure. Ces deux vitesses varient en fonction de la configuration (train sorti, volets sortis, etc.) et de l'attitude (virage, descente, etc.). C'est pourquoi un anémomètre adapté à un aéronef particulier comporte des zones de couleurs différentes :

Figure 2.8.2: Cadran d'un anémomètre de bord

- l'arc vert indique les conditions normales de vol de l'avion,
- l'arc jaune les vitesses interdites en air turbulent,
- l'arc blanc plage de sortie des dispositifs hypersustentateurs, configuration full (volets),
- enfin, le trait rouge indique la vitesse limite (VNE :velocity never exceed), particulièrement pour la structure de l'appareil.

Aujourd'hui, le dispositif utilisé est un instrument appelé badin en France (en 1911, du nom de son inventeur, Raoul Badin) associé au tube de Pitot. C'est un manomètre étalonné en fonction du Théorème de Bernoulli qui détermine la « pression dynamique » qui est égale à la différence entre la pression totale et la pression statique. Cette pression dynamique, est fonction de la vitesse de l'avion par rapport à l'air et permet d'afficher une information de vitesse air. Elle est généralement mesurée en nœuds, mais, sur quelques avions français et sur les avions russes, elle est donnée en kilomètres par heure.

Pour les avions volant à des vitesses proches de celle du son et au-delà, d'autres lois sont applicables et, donc, d'autres instruments : le machmètre.

2.8.3 Machmètre

Le machmètre mesure le rapport entre la vitesse de l'avion et la vitesse du son. Cette information est utile en vol subsonique pour éviter de pénétrer dans le domaine de vol transsonique et en vol supersonique.

2.8.4 Variomètre

Dans sa version classique, cet instrument (figure 2.8.3)utilise les variations de pression statique pour indiquer des variations d'altitude, c'est-à-dire des vitesses verticales. De l'air à la pression statique extérieure est stocké dans une bouteille appelée « capacité » qui se met à pression avec un temps connu. La pression dans la capacité est donc en retard par rapport à la pression courante. Au moment de la mesure, l'instrument fait la différence entre la pression extérieure et la pression de la capacité. Le variomètre fonctionne avec un léger temps de retard, dû au temps de remplissage de la capacité.

2.8.5 Horizon artificiel

L'horizon artificiel ou indicateur d'assiette (figure 2.8.4)mesure l'assiette de l'aéronef par rapport à l'horizon c'est-à-dire les angles de tangage et roulis. Il utilise un gyroscope qui, en principe, conserve le calage initial réglé avant le décollage. Il est particulièrement utile pour le pilotage sans référence visuelle extérieure.

FIGURE 2.8.3: Variomètre

FIGURE 2.8.4: Horizon artificiel classique

2.8.6 Indicateur de virage et de dérapage (bille-aiguille)

L'indicateur de virage(figure 2.8.5) est un gyroscope à deux degrés de liberté qui permet de visualiser le taux de virage (et non l'inclinaison) de l'avion.

Il est associé à une bille qui se déplace dans un tube incurvé selon la verticale apparente et qui visualise le dérapage de l'avion. La bille fonctionne simplement par gravité. En effet, quand le dérapage est nul et le vol symétrique, la gravité relative (gravité équivalente créée par le poids et la force centrifuge) est selon l'axe vertical de l'avion. Si la gravité relative forme un angle avec la verticale du planeur, c'est qu'il existe un dérapage.

FIGURE 2.8.5: Indicateur de virage

2.9 Instruments de navigation

2.9.1 Compas magnétique

Il utilise le champ magnétique terrestre comme référence.

Il est constitué d'une lunette de lecture sur un boitier étanche rempli d'un liquide dans lequel se déplace librement un équipage mobile formé par une rose des caps et des barreaux aimantés. C'est un instrument peu précis qui donne des indications fausses

dès que l'avion n'est pas stable sur une trajectoire rectiligne, horizontale et à vitesse constante. Il est néanmoins utile pour régler ou recaler le conservateur de cap.

De plus, il est influencé par les champs magnétiques engendrés par les équipements électriques de l'avion. Aussi, il est accompagné d'une courbe de calibration, établie dans des conditions standard de mise sous tension des équipements proches.

2.9.2 Gyro compas / gyro directionnel

Il s'agit d'un gyroscope à deux degrés de liberté qui permet de conserver une référence de cap de façon beaucoup plus précise qu'un compas magnétique. Il est asservi à une vanne de flux, qui permet de le recaler automatiquement en fonction du champ magnétique terrestre. Il est aussi appelé « plateau de route ».

2.10 Instruments de surveillance des paramètres moteurs et autres systèmes

2.10.1 Manomètres

Ils indiquent les pressions d'huile, de carburant ou d'admission.

2.10.2 Tachymètre

Il indique la vitesse de rotation du moteur (en tr/min) ou d'un réacteur (en % d'un régime nominal).

2.10.3 Systèmes d'alarmes

2.10.3.1 Avertisseur de décrochage

Il émet un signal sonore ou une vibration du manche le pilote lorsque l'avion s'approche de l'angle d'incidence maximum avant décrochage. Ce système s'appelle Stall Warning System

2.10.3.2 Avertisseur de proximité du sol

L'avertisseur de proximité du sol (GPWS - Ground Proximity Warning System) permet de prévenir (par un message vocal « terrain » ou « pull up ») le pilote lorsque l'avion s'approche du sol.

2.10.3.3 Dispositif d'évitement de collisions

Le dispositif d'évitement de collisions (TCAS - Traffic and Collision Avoidance System) permet de prévenir (sur un écran et par un message vocal « trafic ») le pilote lorsque l'avion s'approche d'un autre avion. Il peut également proposer (en se synchronisant avec le TCAS de l'autre appareil : coordination des manœuvres) une manœuvre d'évitement dans le plan vertical (climb : monter, descend : descendre).

2.11 Système radiotéléphonique

Le système radiotéléphonique permet de transmettre des clairances et des informations importantes pour la sécurité de la circulation aérienne et l'efficacité de la gestion du trafic aérien.

On distingue deux types de services mobiles aéronautiques régis par des procédures différentes :

1. le service mobile aéronautique (R) (« en route dans des couloirs aériens ») réservé aux communications relatives à la sécurité et à la régularité des vols, principalement le long des routes nationales ou internationales de l'aviation civile ;
2. le service mobile aéronautique (OR) (« hors des routes ») destiné à assurer les communications, y compris celles relatives à la coordination des vols, principalement hors des couloirs aériens.

2.12 Conclusion

Nous connaissons maintenant un peu mieux notre aéronef le Zlin et son cockpit, et avons fait une entrée dans le domaine de l'instrumentation aéronautique, élément essentiel à toute navigation et qu'il faudra restituer si on veut recréer l'ambiance visuelle d'un vol à bord de notre avion, ce qui constituera si on s'accorde ce raccourci le partie matériel de notre simulateur, passons alors à la présentation de ce qui sera le coeur de la partie logiciel.

Points clés

Positionnement

- ❏ présentation du Zlin
- ❏ détails du cockpit
- ❏ Instruments de bords dans l'aéronautique

Chapitre 3

Microsoft Flight Simulator FSX

3.1 Introduction

Ce chapitre va présenter le cœur de notre travail à savoir le simulateur de vol de Microsoft, on s'intéressera essentiellement à son SDK et l'API qu'il met a disposition des utilisateurs, représente notre porte d'entrée pour interagir avec la simulation et les données de vol de notre Zlin Z-142.

3.2 Microsoft Flight Simulator

Flight Simulator est un logiciel de simulation de vol le plus réussi au monde, crée et développé par Microsoft depuis les années 80. Considéré dans la plupart du temps comme un jeu vidéo, MFS était l'un des premiers produits dans le portefeuille de Microsoft. Le programme Flight Simulator a été développé à partir de l'année 1977, sous la direction de Bruce Artwick ; et sa société subLOGIC l'a vendu pour divers ordinateurs personnels.

En 1982, la société d'Artwick accorda une licence de développement à Microsoft pour l'IBM PC qui fut commercialisée sous le nom de Microsoft Flight Simulator 1.00, cette ère représente le début d'un commerce juteux concernant les simulateurs de vol. Le développement de Flight Simulator est passé par plusieurs étapes durant ces trois dernières décennies que l'on peut citer :

- 1982 – Flight Simulator 1.0
- 1983 – Flight Simulator 2.0
- 1988 – Flight Simulator 3.0
- 1989 – Flight Simulator 4.0
- 1993 – Flight Simulator 5.0
- 1995 – Flight Simulator 5.1
- 1996 – Flight Simulator 95
- 1997 – Flight Simulator 98

- 1999 – Flight Simulator 2000
- 2001 – Flight Simulator 2002
- 2003 – Flight Simulator 2004 : A Century of Flight
- 2006 – Flight Simulator X (l'outils qu'on va utiliser)

L'un des points les plus fort de la dernière version est le faite qu'elle soit extensible grâce à des programmes développé par les clients, ceci est possible en utilisant la SDK.

3.3 Flight simulator SDK

Microsoft Flight Simulator SDK (Software Development Kit) est un ensemble de fichier et de programmes, fournis à l'utilisateur ui donnant la possibilité d'intéragir avec le Simulateur. le SDK est considéré comme un outils de développement qui permet de développer, créer ou modifier le contenu de la simulation. La majorité des outils sont sous forme de ligne de commande, mais certains sont accessibles via le menu du jeu Outils.

Le SDK de Flight Simulator contient une multitude de fonctions (65 en tout) et une large documentation détaillé pour bien comprendre son fonctionnement. Les applications crées peuvent être nouvelles ou une simple modification pour améliorer des missions, des terrains, des aéroports, des effets spéciaux, les jeux de caméra, et pleins d'autres éléments pour la simulation.

Le SDK est divisé en quatre parties (figure 3.3.1) :

1. Core Utilities Kit : Ce kit couvre, la commande des avions, les données instantanées, la configuration camera, et la modification des évènements imposés par le client.
2. Environment Kit : Ce kit couvre la création des terrains, des scènes, et des effets spéciaux.
3. Mission creation Kit : Ce kit couvre la création de missions (aventures et défis)
4. SimObject Creation Kit : Ce kit couvre la création de panneau pour l'avion, ainsi que tous les autres objets de simulation pouvant apparaître (animaux, bateaux, trains, etc).

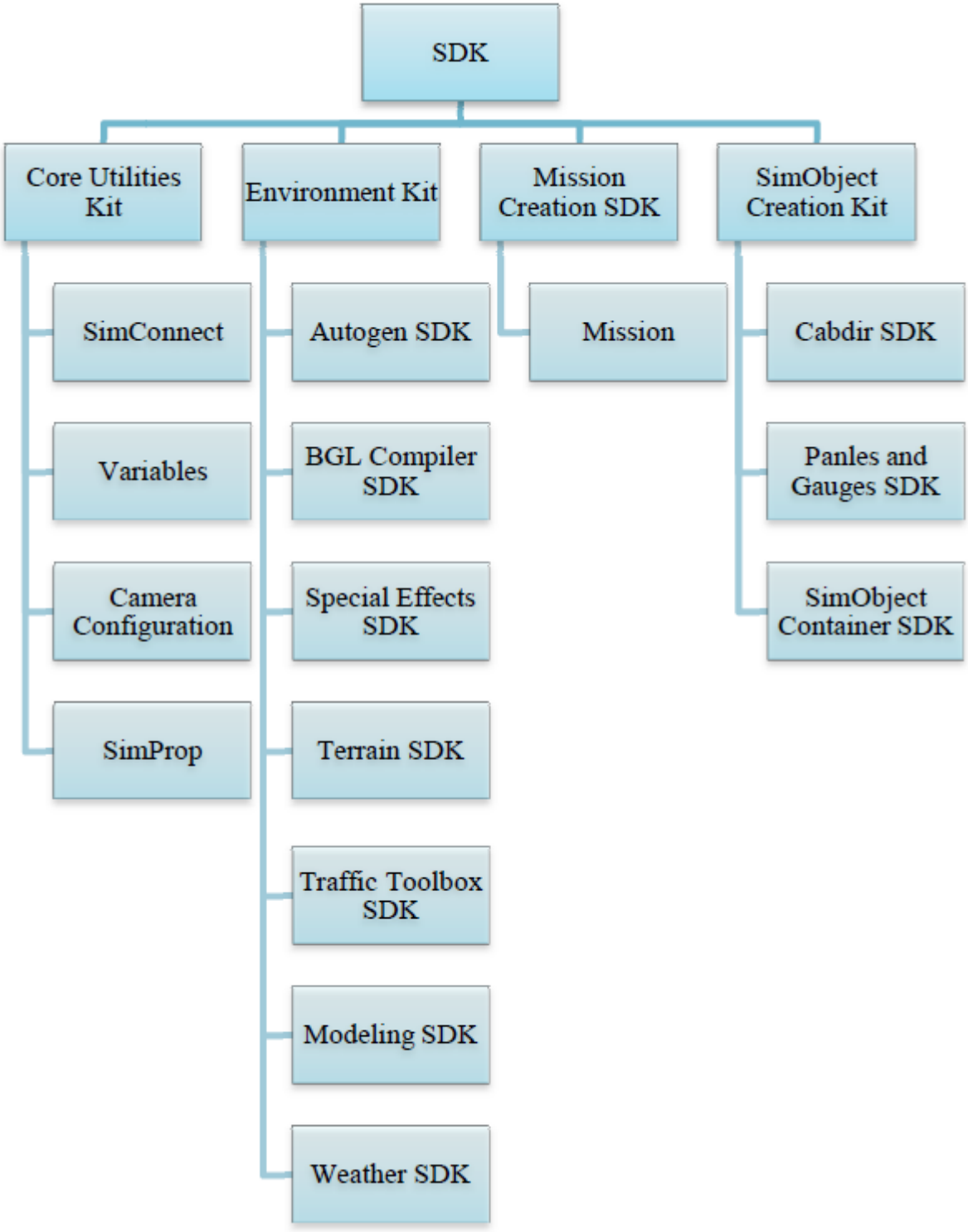
Le développement d'une application propre à soi est possible, en utilisant les API propre à Flight Simulator.

3.4 Généralités sur les API

3.4.1 Définition de l'API

Une API (Application Programming Interface, traduite en langue française « interface de programmation » ou « interface pour l'accès programmé aux applications) est

FIGURE 3.3.1: Les composants du SDK



un ensemble de fonctions permettant d'accéder aux services d'une application, par l'intermédiaire d'un langage de programmation.

3.4.2 Principe de l'API

L'interface n'est pas un « décor » gérant l'aspect externe de l'application. Elle réalise une fonction d'adaptation bidirectionnelle entre l'usager et la partie fonctionnelle de l'application qui effectue les traitements, en d'autre termes, elle masque la complexité de l'accès à une application, en lui offrant un jeu d'instructions assez simple avec comme valeurs connus : les entrées et les sorties, en gros un développeur n'a pas à s'inquiéter de la manière dont elle est implémentée pour pouvoir l'utiliser dans un programme. (figure 3.4.1) :

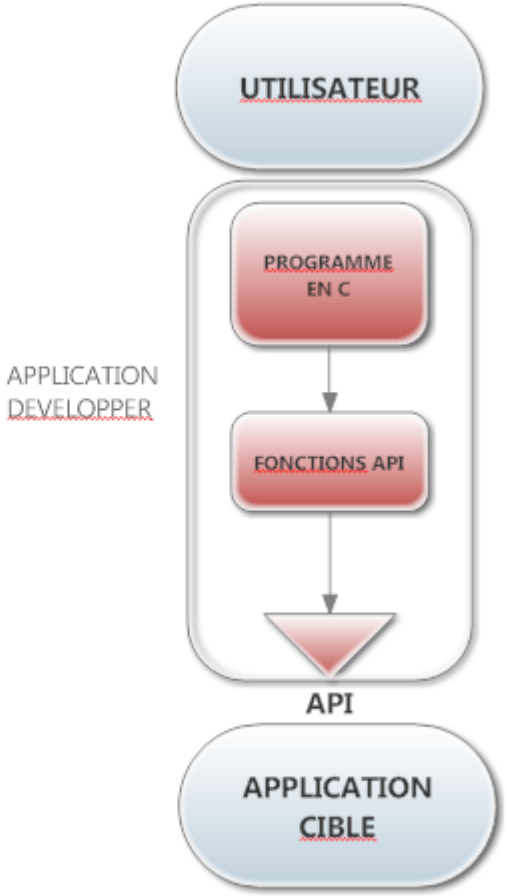


FIGURE 3.4.1: Principe de fonctionnement d'une API

Au finale, une API n'intervient que sur l'application propre à elle, de ce faite elle ne travaille pas seule, elle utilise toujours des fichiers DLL connus sous le nom « dynamique

link library ».

3.4.3 Bibliothèques de lien dynamiques

Une DLL est une bibliothèque contenant du code et des ressources nécessaires à d'autres applications, la raison pour laquelle on lui a attribué ce nom est le faite qu'elle soit appelée durant l'exécution des applications, contrairement aux habituelles bibliothèques dites statiques qui sont introduites après compilation lors de l'edition de lien.

Les bibliothèques logicielles se distinguent des exécutables dans la mesure où elles ne représentent pas une application. Elles ne sont pas complètes, elles ne possèdent pas l'essentiel d'un programme comme une fonction principale et par conséquent ne peuvent pas être exécutées directement. Les bibliothèques peuvent regrouper des fonctions simples (par exemple le calcul d'un cosinus, ou l'inversion d'une matrice) comme des fonctions complexes avec de nombreuses fonctions internes non accessibles directement.

L'intérêt des bibliothèques réside dans le fait qu'elles contiennent du code utile que l'on ne désire pas avoir à réécrire à chaque fois.

Dans notre cas L'API Flight Simulator fait appel à la DLL propre à lui : *SimConnect.dll*, cette dernière interagie directement avec le logiciel Flight Simulator en récupérant les handles des API.

3.4.4 Handle d'application

Cet élément représente l'immatriculation d'un objet informatique (e. g. une fenêtre ou un fichier « essentiellement utilisé par la plateforme windows ») permettant sa « manipulation » ou sa gestion. Ce numéro est attribué par le système d'exploitation, pour différencier les processus actifs, en français Handle pourrait se traduire par poignée, comme une poignée de porte qui permet d'entrer dans une chambre, ainsi en accédant au handle d'un objet, on peut modifier ses propriétés.

Dans notre cas, on a manipulé les objets à travers le handle, mais tout cela n'était pas réalisable sans les API Flight Simulator.

3.5 les APIs Flight Simulator

le SDK est composé de quatre principaux domaines pour faciliter son utilisation. Cependant, il faut souligner le fait que certaines taches telles que la création de nouveaux avions demande beaucoup d'implication ainsi qu'une grande connaissance sans oublier le facteur temps. Pour ces raisons, notre travail sera basé principalement sur la partie « Core Utilities Kit ».

3.5.1 Core utilities Kit

Le Core Utilities Kit contient quatre composant : SimConnect SDK, Variables, Camera Configuration et le SimProp.

Le SimConnect SDK est une bibliothèque dédiée aux programmeurs permettant d'élargir Flight Simulator X, elle peut être utiliser par le programmeur pour écrire des add-on components (ajout de composant) pour le FlightSim. Ces composants peuvent être écris en C, C++ ou voir même en C#.NET. Typiquement, les composants ajoutés peuvent réaliser un ou plusieurs éléments, qu'on peut citer :

- Remplacer le processus de traitement d'un ou plusieurs événements de Flight Simulator avec une nouvelle logique.
- Enregistrer ou surveiller un vol.
- Créer et configurer les plans de vol pour l'intelligence artificielle des avions.
- Configurer différents systèmes météorologiques.
- Activer un nouveau matériel pour travailler avec Flight Simulator.
- Contrôle d'une caméra supplémentaire.

3.5.2 Environment Kit

Durant le jeu, les développeurs peuvent étendre le monde réel à l'aide du kit Environment. Des textures de terrain et des bâtiments en 3D haute résolution peuvent être importés ou modifiés à l'aide des utilitaires fournis dans le SDK. Outre des objets statiques, des effets spéciaux dynamiques peuvent être ajoutés à un emplacement spécifique dans le monde réel, tels que des feux d'artifice, des fontaines ou des éclairs.

3.5.3 Kit Mission Creation

Ce kit permet la création de nouvelles missions, la difficulté est bien sur réglé par l'utilisateur. Un développeur peut créer des scénarios de missions ou de nouveau chalenge en les combinant avec l'API Core Utilities. Il peut aussi intégrer les expériences structurées dotées d'une prise en charge multiutilisateur avec un mode cockpit partagé, qui lui permettrai de prendre place « virtuellement » aux côtés d'un autre utilisateur dans le même avion, pour les cas des applications réseau ; la communication serra établie par la VoIP (Voice over IP).

3.5.4 SimConnect Creation Kit

Ce kit offre la possibilité de personnaliser son avion, en modifiant ses attributs (nom, couleur, bruit, panneaux, mesures, et ainsi de suite), ainsi que la possibilité d'ajouter des objets (animaux, bateaux, trains ... etc). Il permet aussi de customiser un tableau de bord, propre à soi, avec les indicateurs de notre choix (altimètre, anémomètre, horizon artificiel, indicateur de carburant ... etc).

3.6 Utilisation des API Flight Simulator

L'accès aux API de Flight Simulator a été facilité par Microsoft SDK, en les séparant en plusieurs domaines bien spécifiques, l'utilisateur n'aura pas besoin de se perdre dans les applications inutiles, par exemple, pour le changement d'un traitement de Flight Simulator en une autre logique propre à lui, on n'aura pas besoins de chercher, on se dirige directement vers le SimConnect.

Apres la détermination du domaine de travail, une familiarisation avec les fonctions est nécessaire, on doit bien connaitre les paramètres des fonctions pour pouvoir les utiliser correctement par exemple : le handle, la valeur retourné, le nom et le type des variables ... etc.

L'exploitation des API, tel que la construction d'un add-on doit se faire en utilisant le logiciel de Microsoft Visual Studio, ce dernier est un ensemble complet d'outils de développement, utilisant un environnement de développement intégré (IDE) incluant le Visual Basic, Visual C++, Visual C# . . . il permet ainsi de faciliter la création de solutions faisant appel à plusieurs langages, ce qui est le cas des API de Flight Simulator. L'utilisation des API Flight Simulator se fait principalement en utilisant le langage de programmation C++, chose qui est due essentiellement au développement de la bibliothèque SimConnect en ce langage.

3.7 Conclusion

Dans ce chapitre et pour clore la partie théorie, nous avons présenter le logiciel de Microsoft à savoir Flight Simulator qui va de paire avec son kit de développement logiciel le SDK, ce dernier facilitant l'accés à l'API SimConnect, ce concept d'API à eu droit à une présentation de base, nous avons abordés dès lors l'organisatiob interne du SDK pour mieux nous préparer à exploiter ses fonctionalités et à mettre l'API en action, nous rentrons dans la phase de conception proprement dit dans la partie suivante, munis de ces connaissances théoriques de base.

Points clés

Positionnement

- ❏ Microsoft FSX
- ❏ Notion de API
- ❏ SDK et SimConnect

Conception et Réalisation (Restitution Visuelle)

« hatta tyara ma tir fe had eddar. (trd) No plane will fly in this house. (avion en papier) »"

(mother and grandmother)

Chapitre 4

Structure Générale

4.1 Introduction

Nous presentons dans ce chapitre la structure générale de notre projet et en détaillons la partie de Restitution Visuelle aussi bien materiel (tableau de bords en taule), que logiciel (instruments de navigation sur écran)

4.2 Schéma de travail

Sur la figure suivante (figure 4.2.1)apparait notre approche du probleme
On peut distinguer quatres parties essentielles :

1. FSX : c'est la bloc détaillé dans le chapitre précedent qui s'occupera de la simulation et des calculs mathématiques liés a l'aérodynamique de l'avion
2. Application en C : cette partie sera détaillée dans le chapitre suivant.
3. Hardware : c'est essentiellement le circuit MBED, qu'on programmé pour faire le lien avec la parie logiciel et les commandes materiel de l'utilisateur, nous le présenterons dans l'avant dernier chapitre
4. Restitution Visuelle : c'est le bloc qui ajoute du réalisme à la simulation consitité d'un tableau de bords réel construit en taule galvanisée, d'un environnement en 3D restitué par FSX et d'instruments affiché sur un écran emulant de vrai jauges de mesures.

4.3 Tableau de bord

la version finale de notre tableau de bord ressemble a ce rendu (figure 4.3.1) en 3D nous fournisson les plans détaillé en annexe, déssinés avec le logiciel SolidWorks sous licence étudiant

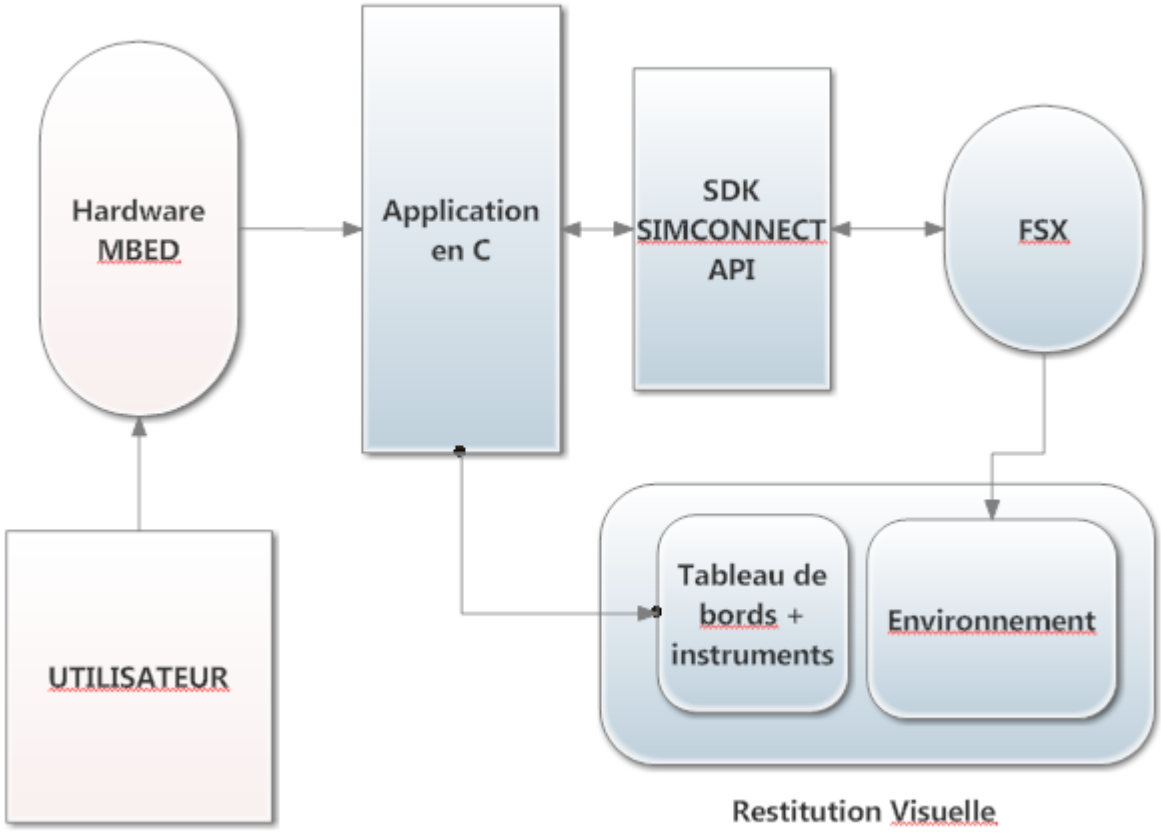


FIGURE 4.2.1: Structure générale

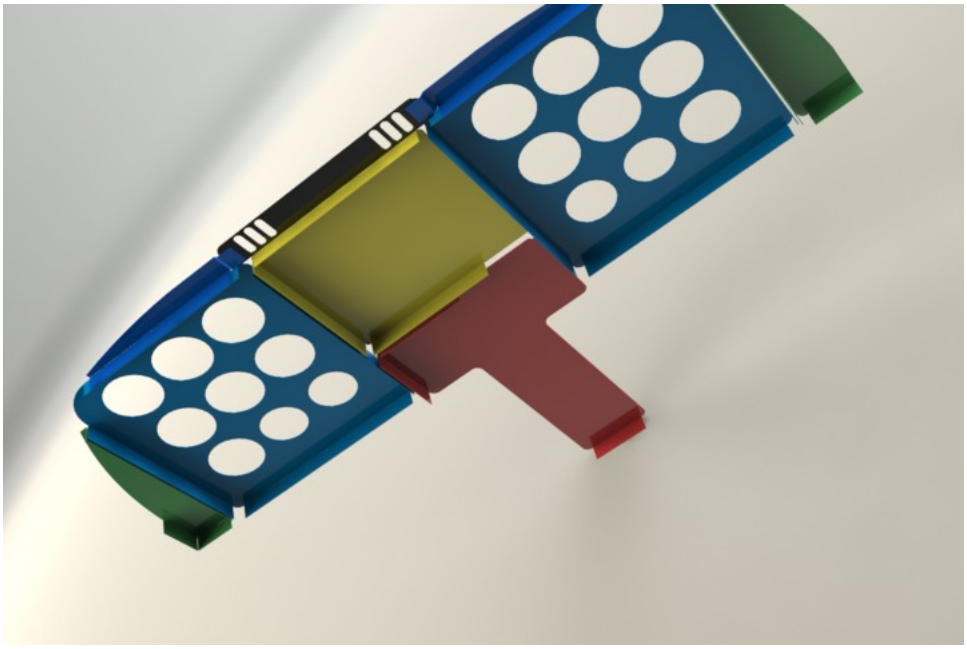


FIGURE 4.3.1: tableau de bords en 3d

pour arriver a ce résultat la fabrication comporte les étapes suivante :

1. conception sur logiciel de CAO 3D du model de tole pliée.
2. déplier le model et le convertir en un format de dessin CAD adapté aux machines numériques
3. transférer le model déplié vers la machine laser pour effectuer les découpes nécessaire dans un carré de matière. A noter que pour optimiser le processus et gagner de l'argent il est evident qu'on essaie de loger le plus de pièce dans une même fourné, on les place l'une a coté de l'autre et on effectue les découpes en une seule fois
4. les pièce déplié sont envoyé vers une plieuse numérique automatisant le processus et les plies nécessaires son effectué
5. on obtient ainsi une reproduction fidèle du model en 3d, qu'on pourra si envie faire passer par une étape de peinture pour lui donner l'aspect adéquat et cacher les défauts, ce qui n'as pas été le cas pour notre model pour des raison financières.

4.4 Instruments simulés

Pour donner vie à l'insrumentation de bord, une des solutions aurait été d'utiliser de vrai innstruments de mesures éléctronique couplés à un circuit de commande connecté a notre simulateur logiciel et qui traduirait le changements des différentes variables de

simulation par des rotation des moteur faisant par la suite tourner une aiguille qui indiquerait, par exemple, l'altitude de l'aoéronef.

Cette approche bien qu'intéressante et souvent utilisé par la communauté des créateur de cockpit, présente des désavantages essentiellement liés au contraintes auquelles on est soumis, tout d'abord le prix des instruments de mesures est exhorbitant, et il ne faut pas perdre de vue qu'il nous faut en afficher au moins neufs à la fois, puis vient le probleme de réutilisabilité, un instrument n'est compatible qu'avec un seul type d'avion, et des qu'on voudra se tourner vers une autre catégorie il faudra changer une partie ou tout le jeu d'instruments

La solution qu'on propose consiste a remplacé les instruments par des fenetres graphiques affichées sur écran plat, qu'on montera derriere le tableau de bord, ainsi chaque instrument possèdera sa propre fenetre propre à lui.

Il faut noter que cette méthode favorise l'indépendance des instruments et des ressources qui lui sont dédiées, ainsi si un probleme ou un bogue survient il n'affectera qu'une seul fenetre et donc qu'un seul instrument

4.4.1 Organigramme De Simulation d'Instruments

le Schéma suivant (figure 4.4.1)présente les étapes nécessaire à l'affichage d'un instrument en générale

4.4.1.1 Détails

Nous expliquons brièvement les étapes présentes dans le schéma

1. Connexion SimConnect, est l'étapes nécessaire pour obtenir un lien avec l'application de microsoft Flight Simulateur, c'est la source des données de navigation d'ou on poura tirer toutes les variables liées aux instruments de bords
2. Initialisation fentere, consiste a aller lire dans un fichier de configuration sous forme de texte les parametre reprsentant la fenetre a afficher c'est un moyen simple de pouvoir enregistré les configuration utilisé lors de la derniere fermeture de la fenêtre et de pouvoir garder une trace des modifications entreprise par l'utilisateur, ces variables peuvent être, la taille de la fenetre, la position dans le plan 2d de l'écran, la présence ou pas de bordure etc ...
3. REQUEST DATA SIMULATION, comme dit plus haut nous avons besoin d'etre connecté a FSX pour en extraire les données de simulation, cette étapes est décrite en détail dans le chapitre suivant, notemment la fonction *RequestDataOnSimObject de l'Api simconnect,* on test par la même occasion les eventuelles erreurs et on agis en conséquence
4. Parser DATA : un parser en informatique est tout objet qui permet d'extraire des donnée qui sont présente dans une forme précise est les mettre sous une forme différente pour être utilisé par un bloc procédurale bien précis, pour faire simple c'est

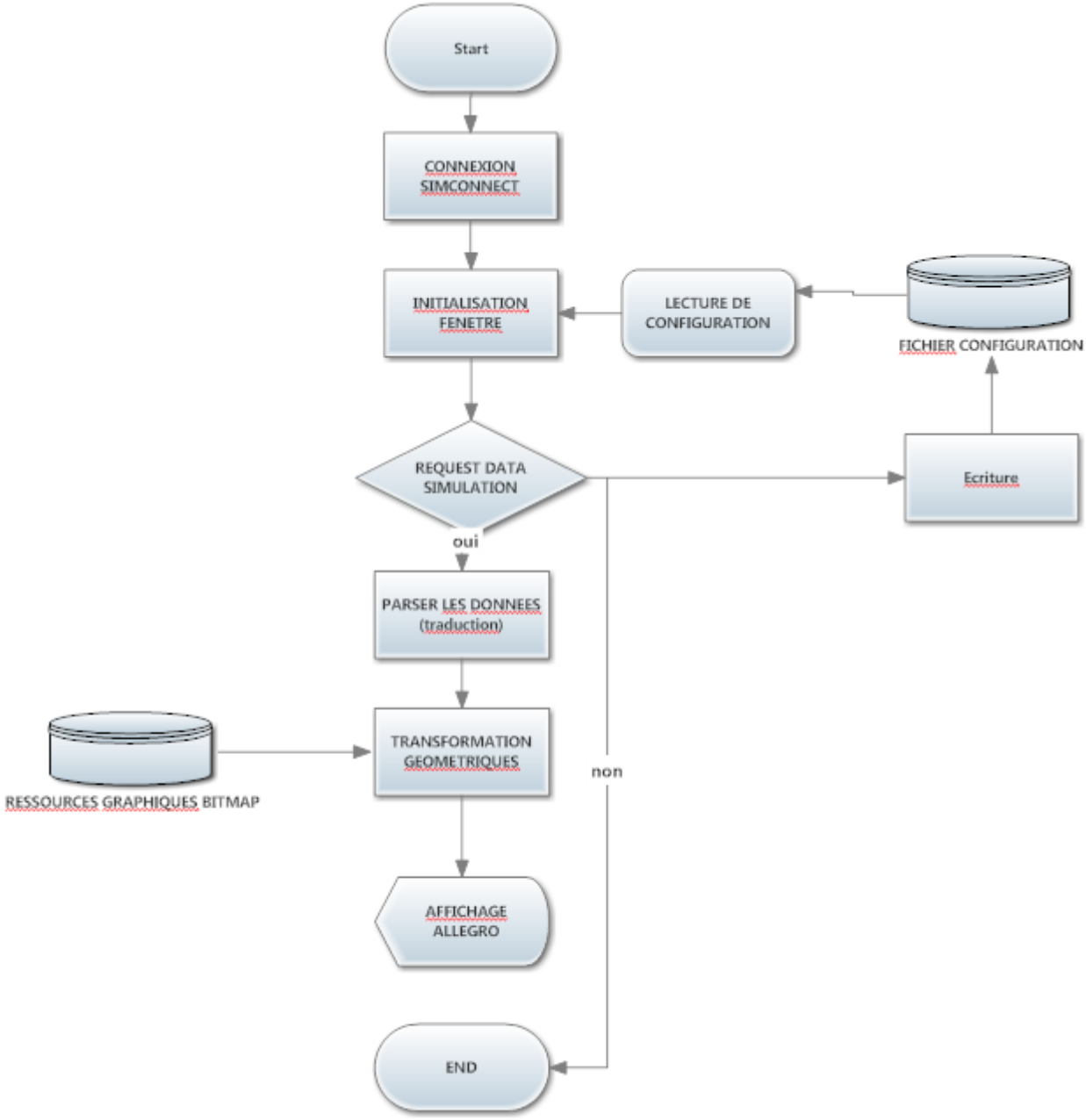


FIGURE 4.4.1: organigramme de création d'instruments virtuelle

une sorte de traduction ou de translation de domaine, par exemple, pour les données d'altitude FSX fournis une altitude en miles, cette valeur n'est pas utilisable en tant qu'elle par le bloc suivant, il faut la traduire par un angle représentant l'angle de l'aiguille qui indique l'altitude dans un altimetre

5. Transformations Géometriques sont de simple rotation d'images bitmap représentant une aiguille par exemple autour d'un axe, nous faison appel pour cela a une bibliothèque de developpement de jeu vidéo mais qui ne se limite pas à ça pour effectuer les différentes opérations le plus efficacement possible sans avoir a manipuler des routines bas niveau et des algorithme de traitement d'image, cette étape est plus banale qu'il n'y parrait, un simple appel a Allegro_rotate suffit a résoudre tout le processus
6. On affiche les résultat obtenus et cela suffit a créer l'illusion d'un vrai instrument de bord qui se mouvoit en temps réel avec l'appareil.

remarque

le code est normalement placé dans une boucle infinie pour tourner en continue tout le long de l'application, ce qui n'est pas décrit dans l'organigramme pour des raison d'encombrement, il suffit juste de savoir que les étapes 3,4,5 et 6 se répete plusieurs fosi par secondes pour avoir un mouvement fluide

ressource graphique

la (figure 4.4.2) représente un exmple de ressource graphique utilisées pour l'affichage des instruments, on distingue la partie fixe qui sert d'arriere plan a l'instrument et de cadran gradué, et une partie mobile comportant de la transparence qui subira les transformation géométrique décrite précedemment.

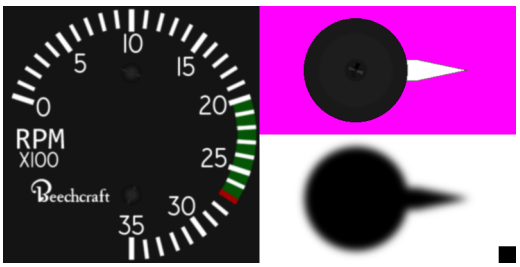


FIGURE 4.4.2: ressource graphique

4.5 Conclusion

Ce chapitre a été un prélude, il sert a établir un cadre bien définis des démarches que nous allons suivre pour atteindre notre but, en définnissant un plan structuré de

notre simulateur et des différents blocs sous jacents qui s'harmonisent pour donner vie à l'experience de vol !

Points clés

Positionnement

- ❏ tableau de bord
- ❏ strucuture générale
- ❏ solidworks

Programmation et Interface

« 01010010100010111011111100101 »"

(Van Wire)

Chapitre 5

API SIM CONNECT

5.1 Introduction

dans cette partie on verra plus en détail le fonctionnement de l'api SimConnect au travers de trois exemple de bases, qui mis ensemble permettent une communication presque complete avec tous les aspects (variables, objets etc) de l'environnement simulé par Flight Simulator X, sans plus tarder commençant avec l'exemple le plus simple qui consiste en l'ouverture d'une connexion avec le serveur de FSX

5.2 SimConnect_Open

Cette fonction envoie une requête au serveur Flight Simulator pour ouvrir une communication avec un nouveau client.

5.2.1 Syntaxe :

HRESULT SimConnect_Open(
HANDLE* phSimConnect,
LPCSTR szName,
HWND hWnd,
DWORD UserEventWin32,
HANDLE hEventHandle,
DWORD ConfigIndex) ;

5.2.2 Paramètres :

– *phSimConnect :* Ce premier paramètre est le handle de la fonction déclaré comme pointeur. C'est un numéro unique attribué par le système d'exploitation qui lui permet de l'identifier.

- *szName* Est le nom attribué par le client au programme.
- *hWnd* Est utilisé au cas où l'utilisateur utilisait un objet handle de Windows. Toujours mis à la valeur NULL s'il n'y en a pas.
- *UserEventWin32* Est un nombre que le client peut spécifier s'il veut présenter un code à cette fonction. Mettre égale à 0 s'il n'est pas utilisé.
- *hEventHandle* Est un handle d'évènement Windows. Le client peut l'ajouter pour synchroniser d'autre application windows qui peuvent interagir avec l'application Flight Simulator. Mettre à 0 s'il n'y en pas.
- *ConfigIndex* L'index de configuration, est toujours ajouté au cas où le client voudrait apporter des modifications au niveau du fichier SimConnect.cfg. Mettre à 0 s'il n'y a pas de modification à mettre.

5.2.3 Valeur retournée

Cette fonction retourne un HRESULT qui peut prendre les trois valeurs suivantes :
- *S_OK* L'appel à la fonction a réussie.
- *E_FAIL* L'appel à la fonction a échouée.
- *E_INVALIDARG* Une erreur s'est produite lors de l'appel du dernier paramètre, probablement un argument manquant est signalé.

5.2.4 Exemple d'utilisation

Le programme suivant essaye d'établir une connexion avec le serveur de FSX, et la referme immédiatement si la procédure a réussis, sinon il affiche un message d'erreur et retourne la main au programme principale pour quitter l'application

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>
#include <strsafe.h>
#include "SimConnect.h"
HANDLE hSimConnect = NULL;
void testOpenClose()
{
HRESULT hr;
if (SUCCEEDED(SimConnect_Open(&hSimConnect, "Open and Close",
NULL, 0, 0, 0)))
{
printf("\nConnected to Flight Simulator!");
hr = SimConnect_Close(hSimConnect);
printf("\nDisconnected from Flight Simulator");
}
```

```
else printf("\nFailed to connect to Flight Simulator");
}
int ___cdecl _tmain(int argc, _TCHAR* argv[])
{
testOpenClose();
return 0;
}
```

5.3 Les Evenement client

Nous illustrons l'utilisation de trois fonctions principales a travers de cet exemple qui utilise *SimConnect_MapClientEventToSimEvent*, *SimConnect_AddClientEventToNotificationGroup,* et SimConnect_SetNotificationGroupPriority.

cela va nous permettre d'avoir des notification sous forme de signaux qu'on traitera dans des strucutres de controle généralement de type *IF ... ELSE* ou *Switch ... Case* et d'agir en conséquence en effectuant des actions bien précises en rapport avec la partie logiciel ou matériel de notre simulateur.

5.3.1 Code source Exemple (evenement action de la commande des freins)

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>
#include <strsafe.h>
#include "SimConnect.h"
int quit = 0;
HANDLE hSimConnect = NULL;
static enum GROUP_ID { GROUP0, };
static enum EVENT_ID { EVENT_BRAKES, };
void CALLBACK MyDispatchProc1(SIMCONNECT_RECV* pData, DWORD
cbData, void *pContext) {
switch(pData->dwID){
case SIMCONNECT_RECV_ID_EVENT :
{
SIMCONNECT_RECV_EVENT *evt = (SIMCONNECT_RECV_EVENT*)pData;
switch(evt->uEventID) {
case EVENT_BRAKES : printf("\nEvent brakes : %d", evt->dwData);
break;
default :
break; }
```

```
break;
}
case SIMCONNECT_RECV_ID_QUIT :
{
quit = 1;
break;
}
default :
break;
}
}
/**************************************************************/
void testClientEvents() {
HRESULT hr;
if (SUCCEEDED(SimConnect_Open(&hSimConnect, "Client Event", NULL,
0, NULL, 0))) {
printf("\nConnected to Flight Simulator!");
hr = SimConnect_MapClientEventToSimEvent(hSimConnect, EVENT_BRAKES,
"brakes");
hr = SimConnect_AddClientEventToNotificationGroup(hSimConnect, GROUP0,
EVENT_BRAKES);
hr = SimConnect_SetNotificationGroupPriority(hSimConnect, GROUP0,
SIMCONNECT_GROUP_PRIORITY_HIGHEST);
while( 0 == quit ) {
SimConnect_CallDispatch(hSimConnect, MyDispatchProc1, NULL);
Sleep(1); }
hr = SimConnect_Close(hSimConnect); } }
/**************************************************************/
int __cdecl _tmain(int argc, _TCHAR* argv[])
{
testClientEvents();
return 0;
}
```

5.3.2 Fonctions Principales

5.3.2.1 SimConnect_MapClientEventToSimEvent

Cette fonction associe un ID event définie par le client avec un événement attribué à Flight Simulator.

Syntaxe

HRESULT SimConnect_MapClientEventToSimEvent(
HANDLE hSimConnect,
SIMCONNECT_CLIENT_EVENT_ID EventID,
const char* EventName) ;

Paramètres

- hSimConnect Un handle vers un objet SimConnect.
- EventID Spécifie l'ID de l'événement client.
- EventName Spécifie le nom de l'événement de Flight Simulator

Valeur retournée

Cette fonction retourne un HRESULT qui peut prendre les deux valeurs suivantes :
- S_OK L'appel à la fonction a réussie.
- E_FAIL L'appel à la fonction a échouée.

5.3.2.2 SimConnect_AddClientEventToNotificationGroup

Cette fonction est utilisée à fin d'ajouter un événement -préalablement définie et associé à un événement Flight Simulator- à un groupe d'événement spécifique au client

Syntaxe

HRESULT SimConnect_AddClientEventToNotificationGroup(
HANDLE hSimConnect,
SIMCONNECT_NOTIFICATION_GROUP_ID GroupID,
SIMCONNECT_CLIENT_EVENT_ID EventID,
BOOL bMaskable = FALSE) ;

Paramètres

- hSimConnect Un handle vers un objet SimConnect.
- GroupID Spécifie l'ID du groupe défini par client
- EventID Spécifie l'ID de l'événement client.
- bMaskable De type booléen, la valeur True indique que l'évènement a été masqué par le client et ne peux être transmit à d'autre client Flight Simulator au cas d'une application réseau. False, le contraire.

Valeur retournée :

Cette fonction retourne un HRESULT similaire à la fonction décrite précedemment

5.3.2.3 SimConnect_SetNotificationGroupPriority

La fonction SimConnect_SetNotificationGroupPriority est utilisée pour ajouter la priorité pour un groupe d'événements (utilisée pour les applications réseau).

Syntaxe

HRESULT SimConnect_SetNotificationGroupPriority(
HANDLE hSimConnect,
SIMCONNECT_NOTIFICATION_GROUP_ID GroupID,
DWORD uPriority) ;

Paramètres

- hSimConnect Un handle vers un objet SimConnect.
- GroupID Spécifie l'ID du groupe défini par client
- uPriority Spécifie la priorité du groupe.

Valeur retournée

Cette fonction retourne un HRESULT

Priorités SimConnect

Priorités	Valeur	Description
SIMCONNECT_GROUP_PRIORITY_HIGHEST	1	haute .
SIMCONNECT_GROUP_PRIORITY_HIGHEST_MASKABLE	10	haute msquable
SIMCONNECT_GROUP_PRIORITY_DEFAULT	20	par default
SIMCONNECT_GROUP_PRIORITY_LOWEST	40	basse priorité

Table 5.1: Priorités SimConnect

5.3.2.4 DispatchProc

Cette fonction est développée et appelée par le client, comme une fonction de rappel (callback fonction), à fin d'établir toutes les communications nécessaires avec le serveur Flight Simulator.

Syntaxe

void CALLBACK DispatchProc(
SIMCONNECT_RECV* pData,
DWORD cbData,
DWORD cbData void * pContext) ;

Paramètres

- pData Pointeur de données traitées comme une structure SIMCONNECT_RECV.
- cbData La taille de la données en octet.
- pContext C'est le même paramètre utilisé dans la fonction précédente.

Valeur retournée

Cette fonction ne retourne aucune valeur.

5.3.2.5 SimConnect_CallDispatch

Cette fonction est utilisée pour traiter le prochain message reçut par une fonction de rappel (callback).

Syntaxe

HRESULT SimConnect_CallDispatch(
HANDLE hSimConnect,
DispatchProc pfcnDispatch,
void * pContext) ;

Paramètres

- hSimConnect Un handle vers un objet SimConnect.
- pfcnDispatch Spécifie la fonction de rappel (the callback fonction).
- pContext Spécifie un indicateur que le client peut définir qui sera retourné dans le rappel de service.

Valeur retournée

Cette fonction retourne un HRESULT.

5.4 Transmission d'evenement

Il arrive qu'on aie besoin de créer des evenements personnalisable, qu'il faille déclencher indépendemment de la boucle de simulation de FSX, et ainsi créer de nouvelles situation.

Les créateurs de l'API SimConnect ont pensé à ça et developpés une fonction trés utile, qu'on présente ci-dessous

5.4.1 SimConnect_TransmitClientEvent

Cette fonction est utilisée pour transmettre des événements à l'application Flight Simulator

Syntaxe

HRESULT SimConnect_TransmitClientEvent(
HANDLE hSimConnect,
SIMCONNECT_OBJECT_ID ObjectID,
SIMCONNECT_CLIENT_EVENT_ID EventID,
DWORD dwData,
SIMCONNECT_NOTIFICATION_GROUP_ID GroupID,
SIMCONNECT_EVENT_FLAG Flags) ;

Paramètres

- hSimConnect Un handle vers un objet SimConnect.
- ObjectID Par default égale à 0, utilisé pour les applications réseau.

- EventID Spécifie l'ID de l'événement client.
- dwData Double word contenant des données nécessaires à des événements spécifiques.
- GroupID Spécifie l'ID du groupe défini par le client.
- Flags Peut prendre une des identifications suivantes

Flags

- 0 : Ne rien faire
- SIMCONNECT_EVENT_FLAG_SLOW_REPEAT_TIMER : Ce flag remettra à zéro le temporisateur de répétition pour simuler la répétition lente. Employez ce drapeau pour remettre à zéro (activer) le temporisateur de répétition qui fonctionne avec divers événements de clavier et clics de souris.
- SIMCONNECT_EVENT_FLAG_FAST_REPEAT_TIMER : Ce flag remettra à zéro le temporisateur de répétition pour simuler la répétition rapide.
- SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY : Indique au serveur SimConnect que le traitement commence d'abords par ordre de priorité, et que la vitesse de temporisation en dépandra aussi.

Valeur retournée

Cette fonction retourne un HRESULT.

5.4.2 Code exemple :

le code est disponible en Annexe car trop volumineux.

5.5 Organigramme de l'Application Hote sur Ordinateur

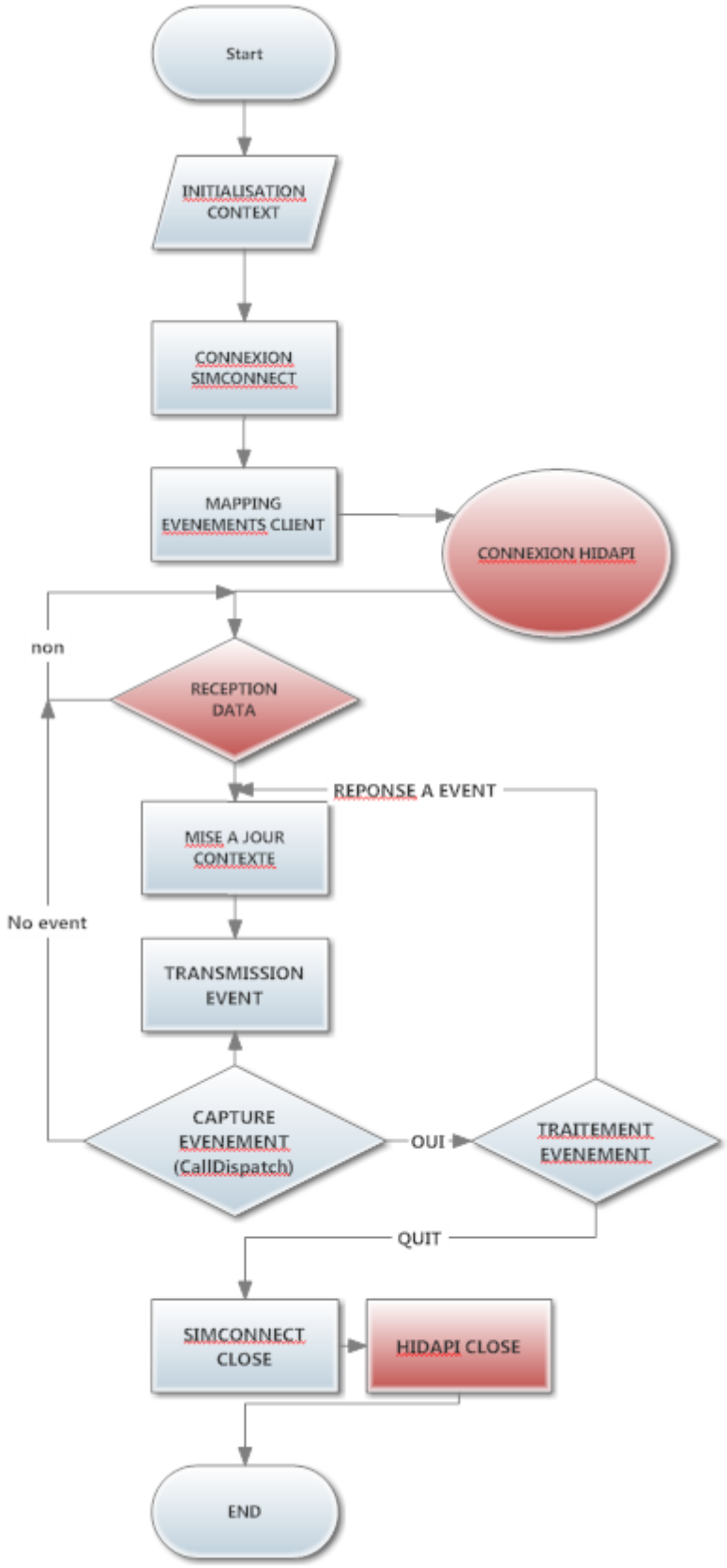
sur la figure qui suit (figure 5.5.1), est représenté le processus interne de l'application basée sur SimConnect et qui sera amené (on décrira comment dans le chapitre suivant) a communiquer avec le hardware externe et de ce fait avec l'utilisateur, mais il faut précisé qu'elle ne se limite pas a ça et que nimporte quel autre programme pour intéragir avec cette application, par exemple on peut imaginer un system qui enregistre les commandes entrées durant un vol dans un fichier, et peut relire ce fichier séquentiellement pour restituer la même situation, une sorte de boite noire virtuelle.

5.5.1 Détails

Voici les détails des différents processus présenté sur ce schéma :

1. Initialisation du Context : par contexte on entend l'ensemble des variables et structure qui définissent l'état de la simulation et des commandes de l'avion à un instant donné, le context mis sous cette forme selon une organisation simple et précise permet de consulter a nimporte quel moment l'etat de la simulation sans avoir a

FIGURE 5.5.1: organigramme application



recourir a nimporte quel aspect de la restitution visuelle, par exemple si on veut savoir si le train d'attérissage est sortis ou pas on va consulter nos variables de contexte et chercher le champs reservé a cette action de sortir le train d'attérissage etc ... Ce principe est trés utile lors des étapes de débugging ou un simple printf permet d'avoir une idée sur ce qui se passe, encore mieux on peut loguer en continue les informations dans un fichier reservé a cet effet, et faire le diagnostique en avale de la simulation

2. Connexion SimConnect : un simple appel a la fonction SimConnectOpen, en lui passant les parametres nécessaires, pour récupérer un handle sur la simulation en cours
3. Mapping : cette étape permet d'associer a chaque evenement de simulation mis à notre disposition Un evenement client correspondant qu'on manipulera a la place grace a son identifiant unique, une bonne façon d'organiser cela est d'utiliser des structure de donnée de type énumération, se chargeant automatiquement d'initialiser nos event à des valeur différentes.
4. Connexion : cette étape mis en surbrillance en rouge n'est pas nécessaire au bon fonctionnement de l'application hote en tant que telle, comme cela a été expliqué plus haut, on pourra la remplacer par toute autre source continue et régulière de donnée formaté convenablement et suivant notre protocole de traduction *(Parser souvenez vous)* des variables.
5. Mise a jour du Contexte : on met nos variables d'état de simulation suivant la réception des données et instructions pour rafraichir en quelque sorte l'application.
6. Transmission des evenement : suivant le nouveau contexte obtenu on transmet des evenement personnalisé via nos identifiant unique d'evenement décrit plus haut, pour mettre le simulateur a l'état désiré par l'utilisateur ou tout autre application, il faut préciser que cette étape doit se dérouler au minimum six fois par seconde (taux de polling d'un Joystick) pour garantire une fluidité de base du system FSX + APPLICATION, en pratique ce taux de rafraichissement peut etre largement dépassé sans probleme, les ordinateur étant bien plus rapide que cela.
7. Capture des evenement : il arrive que des evenements soient produits par le simulateur, pour nous prévenir d'une exception ou de quelque chose d'important qui doit etre traité en priorité, cette étape est primordiale donc. il faut alors traiter ces evenements et revenir a l'etape de mis ajour du contexte, sinon on revient toujours vers l'attente de donnée.
8. finalement si un evenement de type Fin d'application se produit (Event_Quit) on doit alors libérer la mémoire et nettoyer la pile de processus en faisant appel successivement à SimConnectClose et HIDAPI_exit.

Remarque :

en ce qui concerne l'étape 7 il faut savoir que tout le evenement n'ont pas la même priorité et que donc il faut les traiter en sachant bien cela, aussi même si un evenement

est trés prioritaire, il se peut qu'on puisse pas le capturer si un abonnement au préalable n'as pas été correctement fait lors de l'étape de Mapping.

5.6 Conclusion

pour conclure, nous pouvons dire que grace à l'API SimConnect et ses fonctions diverses adaptées à chaque aspect du simulateur, nous pouvons établir une connexion avec n'importe quel programme transmettre des instructions, récuperer des données, lire l'état du simulateur, gérer les évenements lorsqu'ils apparaissent, et les traiter, ce qui nous ouvre la route pour une cohabitation avec le module suivant : à savoir le microcontroleur Mbed via communication USB.

Points clés

Positionnement

- ❏ SIMCONNECT_OPEN
- ❏ communication avec FSX
- ❏ évènement client personalisable

Chapitre 6

le microcontroleur MBED

6.1 Introduction

les microntroleurs MBED sont des series de cartes ARM pour developpement, conçue pour des prototypages rapide.

le mbed NXP lpc1768 (figure 6.1.1) est conçue particulierement pour le prototypages de divers composants, spécialement ceux qui incluent l'Ethernet, USB, et bénéficie de la souplesse de beaucoups d'autres périphériques et interfaces ainsi que des mémoires flashs

FIGURE 6.1.1: mbed NXP LPC1768

6.2 Caractéristiques

la figure suivante (figure 6.2.1) montre les différentes options et le brochage du microcontroleur

6.2.1 Caractéristiques détaillées

NXP LPC1768 MCU

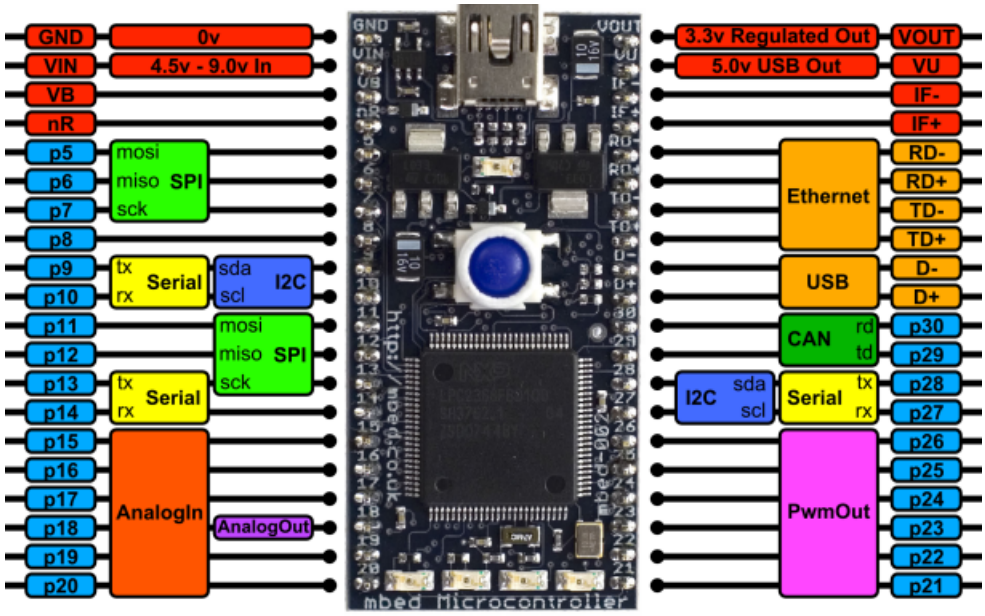


FIGURE 6.2.1: schéma détaillé du Mbed

- High performance ARM® Cortex™-M3 Core 96MHz
- 32KB RAM
- 512KB FLASH
- Ethernet,
- USB Host/Device
- 2xSPI
- 2xI2C
- 3xUART
- CAN
- 6xPWM
- 6xADC

mbed.org site internet pour developpeur

- Un compilateur en ligne léger et efficace
- language de programmation haut niveau C/C++ SDK

6.3 Création d'un programme

le mbed utilise un compilateur en ligne et un IDE complet pour éditer son code source et le tester très rapidement

6.4 Communication série avec un PC

le mbed peut communiquer avec un PC hote à travers un port serie virtuel, ceci nous permet de :

- imprimer des messages sur un terminal PC
- Lire les entrée du clavier PC hote
- Communiquer avec des application se trouvant sur le pc hote.

6.4.1 Exemple de code

Ce code permet de controler la luminosité de diodes graces a des entrées au clavier sur le pc

```
#include "mbed.h"
Serial pc(USBTX, USBRX); // tx, rx
PwmOut led(LED1);
float brightness = 0.0;
int main()
{
pc.printf("Press 'u' to turn LED1 brightness up, 'd' to turn it down\n");
while(1)
{
char c = pc.getc();
if((c == 'u') && (brightness < 0.5)) { brightness += 0.01; led = bright-
ness; }
if((c == 'd') && (brightness > 0.0)) { brightness -= 0.01; led = bright-
ness; }
}
}
```

6.5 HIDAPI

HIDAPI est une librairie multiplatforme qui permet d'interfacer facilement ses programmes ecrits en C avec un circuit eléctronique de type HID *(human interface device)* tels un clavier une souris etc.

Nous utilisons cette librairie pour envoyer des donnée et des commandes sous forme de train de bits a notre application basé sur SimConnect pour commander notre simulateur

6.5.1 Exemple de communication

ce code source montre une communication entre deux applications l'une sur l'ordinateur, et l'autre sur le circuit MBED.

les étapes suivantes sont décrite :

1. ouverture d'une connexion stable
2. écriture d'un buffer de données à envoyer
3. envoie du buffer
4. lecture d'un buffer de donnée reçue
5. traitement des données en consequence

```
#include <stdio.h>
#include <wchar.h>
#include <string.h>
#include <stdlib.h>
#include "hidapi.h"
#include <windows.h>
int res;
unsigned char buf[9];
hid_device *handle;
int i;
int main(int argc, char* argv[]) {
UNREFERENCED_PARAMETER(argc);
UNREFERENCED_PARAMETER(argv);
// Open the device using the VID, PID, and optionally the Serial number.
handle = hid_open(0x1234, 0x6, NULL);
if (!handle) { printf("unable to open device\n"); return 1; }
// Set the hid_read() function to be non-blocking.
hid_set_nonblocking(handle, 1);
// send and receive 20 reports
for (int r = 0; r < 20; r++)
{
/*/ writing data buf[0] = 0x0;
for (i = 1; i < sizeof(buf); i++) { buf[i] = r + i;}
res = hid_write(handle, buf, sizeof(buf));
// reading data non blocking way
res = 0;
while (res == 0) {
res = hid_read(handle, buf, sizeof(buf));
if (res == 0)
// printing read data
printf("Data read :\n ");
for (i = 0; i < res; i++)
printf("%d ", buf[i]); printf("\n");
}
```

```
//close HID device
hid_close(handle);
hid_exit();
return 0;
}
```

resultat :

on peut voir un exemple de la transmission de données aléatoire dans la (figure 6.5.1) :

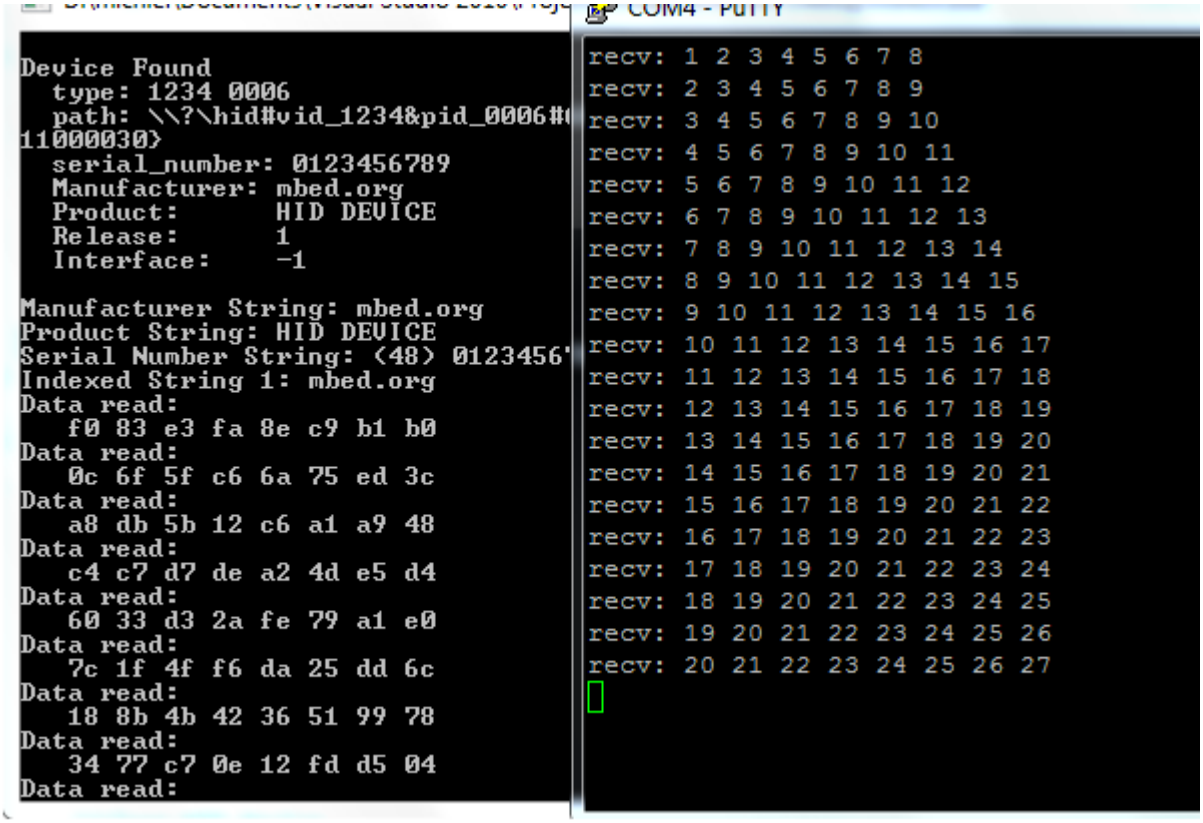


FIGURE 6.5.1: exemple d'execution

remarque :

ces donnée aléatoire sont tout simplement remplacé par un train de bits adéquat décrivant l'état de chaque borche du circuit digitale ou analogique et envoyé vers l'application hote pour traitement et execution sur la simulation.

6.6 Conclusion

Nous avons finalement réussis à raccorder le logiciel FSX et notre application basé sur l'Api SimConnect avec le monde extérieur, l'utilisateur peut ainsi intéragir à l'aide de controle réel, tel des switch des résistance variables, de sélécteurs multipositions etc ... monté de façon a emiter les commandes mécanisés de l'avion, et ce en lieu et place de la configuration de base clavier plus souris bien moins réaliste.

Points clés

Positionnement

- ❏ MBED
- ❏ C/C++ SDK
- ❏ Communication PC

Chapitre 7

EPILOGUE : Les Threads

7.1 Introduction

Le choix d'utiliser des threads pour séparer chaque unité d'exécution de notre projet s'est fait assez tardivement, lorsqu'on s'est rendu compte que le nombre conséquent de programmes tournant en mm temps se ralentissait mutuellement et provoquait des erreurs et des bogues de la machine, les threads offrent une solution efficace à ce problème récurrent, et permettent d'améliorer les performances.

7.2 Qu'est ce qu'un thread ?

Windows peut executer plusieurs programmes (plusieurs tâche - multitask) en même temps, les threads sont des sous-tâches executées dans les programmes, autrement dit, le même programme peut exécuter deux choses en même temps (plusieurs thread - multithread), évidemment, ce n'est qu'une illusion, on a toujours qu'un cpu qui ne fait qu'une seule chose à la fois.

Cela permet par exemple d'afficher les images contenues dans un dossier sous forme de petite vignettes, tout en laissant l'utilisateur changer de dossier. C'est d'ailleurs exactement ce que fais l'explorateur de Win32. Il faut également savoir que tous programme contient au moins une thread, elle est crée par Windows, et terminée lors de la fermeture du programme. Cette thread est appelée *thread principale.* Windows alloue le processeur aux thread selon leur prioritées définissable lors de leur création.

7.3 les différents types de threads

On peut séparer les threads Windows en deux types distinct, les thread de service et les thread d'interface,

les thread de service n'ont pas de pompe à message, n'affichent rien l'écran (mais peuvent demander à la thread principale de le faire...), en bref, elles n'ont pas besoin de fenêtre pour exister il s'agit en fait d'une simple fonction à laquelle, on passe un paramètre (de type DWORD (32bit), autrement dit on va lui passer un pointeur vers notre structure qui contiendra tous ce dont on aura besoin), comme un fonction main(...)¸ on peut virtuellement faire tous ce qu'on veut mais n'ayant pas de pompes à messages, on pourra pas utiliser toute l'api de Windows

Les threads d'interfaces, elles ont leur pompe à message, elles ne sont pratiquement pas utilisée et les utiliser revient souvent à créer une application dans une application. Un exemple de thread d'interface : L'explorateur de Windows, si on va dans le Gestionnaires de Tàche on verra jamais qu'un seul explorer.exe (même si on a 10 fenêtres de l'exploration lancée) cela s'explique par le fait que les différentes fenêtre de l'explorateur sont en faite des threads du même process.

ça a l'avantage, que comme toutes les threads se partagent le même espace de mémoire les nombreuses ressources consommées par l'explorateur n'existe qu'une seule fois en mémoire (les ic^ones, les informations sur les fichiers, les caches... etc...) mais ça à le désavantage que si une fenêtre de l'explorateur plante, toute les fenêtres seront tuées.

7.4 La Création de Threads

Pour créer une thread, il suffit d'appeler l'API CreateThread il faut lui passer six parametres :

1. le premier, un pointeur vers SECURITY ATTRIBUTES , passer Null.
2. Le second : un DWORD (dwStackSize), d´efinissant la taille de la pile pour le thread, le plus simple est de passer NULL , de cette façon Win32 prendra une valeur par défaut qui est bien dans la majorités des cas.
3. LPTHREAD_START_ROUTINE lpStartAddress, un pointeur des parametres qu'on va passer a notre thread.
4. Et puis la fonction qui correspondra au thread, à l'image du main() d'un programme dos, cette fonction est appelée, quand elle se termine, la thread est finie.
5. signale si la thread est lancé au d´ebut du programme ou pas
6. Et enfin un pointeur vers un DWORD qui contiendra le 'thread identifier' (un numéro identifiant le thread de façon unique).

7.4.1 Un petit exemple

L'exemple suivant est un exemple basique de création d'une thread.

```
DWORD WINAPI Thread1(LPVOID lpParameter);
{
```

```
printf("Ceci est ma thread\n");
}
int main()
{ DWORD threadID;
CreateThread(NULL, 0, Thread1, NULL, 0, &threadID);
Sleep(1000);
printf("fin du programme de test\n");
return 0;
}
```

Il ne fait qu'afficher deux lignes de textes dans la console. la seule chose qui peut paraitre bizarre : le "Sleep(1000) ; ", que fait-il ?

Il faut alors se rappeler quand Win32 termine un programme : le programme est considéré comme fini quand le premier thread (le thread principal) est terminé, autrement dit : sans le délai d'attente, le Thread1 pourrait ne jamais être exécuté, la fin du main marquant la fin du thread principal.

7.5 Les bases de la synchronisation

Cette section va vous donner quelques moyens pour synchroniser les threads

7.5.1 Les events

Nous en sommes resté au problème de faire attendre la thread, la solution qui saute aux yeux, définir une variable globale, contenant un ´etat : 1 ou 0... et pour faire attendre le thread, on ferait quelque chose du genre :

```
while(ma_variable != 1)
(la on exécute le code du thread)
```

Mais cela n'est pas à faire ! tous simplement parce qu'une boucle consomme du temps cpu, temps qui pourrait être beaucoup mieux utilisé ailleurs. Il faut donc chercher autre chose. mais pourtant le principe est bon, en fait ce qu'il faudrait c'est remplacer la boucle par autre chose, un appel Win32, qui aurait le même rôle, mais qui serait correctement géré par le kernel afin de ne pas consommer bêtement du cpu... cela s'appelle les "events" (évènement). Il s'agit exactement d'une variable définie à 1 ou 0 (signalé ou non), pour créer ce type de varibale, on utilise la fonction : *HANDLE CreateEvent()*

7.5.1.1 Exemple de code

Nous allons, puisque nous savons maintenant demander à un thread d'attendre, "corriger" notre exemple de façon à éviter le Sleep(1000) ;. Ce qui donne :

```
HANDLE g_event;
DWORD WINAPI Thread1(LPVOID lpParameter)
{
printf("Ceci est ma thread\n");
SetEvent(g_event);
}
int main()
{
g_event=CreateEvent(NULL, 0, 0, NULL);
DWORD threadID;
CreateThread(NULL, 0, Thread1, NULL, 0, &threadID);
WaitForSingleObject(g_event, INFINITE);
printf("fin du programme de test\n");
return 0;
}
```

Voilà ! Le thread principale lance notre thread, elle s'exécute, pendant que la première attend l'´evènement g event, dès que la seconde a fini, elle signale cette event, le thread principal continue alors, et se termine.

7.5.2 Les mutex

Les mutex sont d'autre objets de synchronisation bien pratique, ils sont très proche des event, on peut les voir comme une spécialisation des events. La différence étant dans le fait qu'un mutex est automatiquement remis à l'état non-signalé dès qu'une fonction d'attente a retourné du fait de son signalement : ça permet d'éviter d'avoir plusieurs threads qui se ruent sur la même ressource suite au signalement d'un event.

7.6 Les autres mécanismes de synchronisations

7.6.1 Les sections critiques

On peut les voir comme des mutex, mais elles présentent certaines particularités dans un environnement multiprocesseur. Elles offrent aussi des api plus concises (une fonction reprenant plusieurs opérations).

7.6.2 Les sempahores

Comme les events, ils peuvent être signalés ou non. Ils ont la particularité de maintenir un compteur : lorsqu'un thread appelle une fonction d'attente, le compteur est décrémenté, quand le compteur atteint 0, l'objet est défini comme non signalé. Un appel à ReleaseSemaphore(...) incrémente le compteur.

7.6.3 Les autres

Il existe encore d'autre objets de synchronisation il s'agit :
- des *timer-queue* permettant de gérer une série d'opération devant s'exécuter à un moment donné
- des *waitable-timer* qui sont des "events" qui seront signalés au bout d'un certain moment, on peut aussi y associer une fonction qui sera exécutée à ce moment là.

Tous ces objets sont en fait parfaitement évitable : on peut les recréer à partir des objets de bases (qui sont les events).

7.7 Conclusion

Ce chapitre majoritairement théorique a servis à introduire le concept de thread, ou sous unité d'execution, pour exploiter le plus efficacement les ressources de l'ordinateur, et est de ce fait une sorte de *Cloture* en douceur du projet et non une partie indispensable au bon fonctionnement du tout.

Points clés

Positionnement

- ❏ Threads
- ❏ Synchronisation
- ❏ event, mutex, sémaphores

Conclusion et perspectives

Rappel des objectifs

Le but de départ était de s'approcher le plus possible d'un simulateur de vol matériel comme on en trouve dans les centres de formation pour pilote de ligne par exemple en réalisant une restitution physique et visuelle d'un tel environnement, à l'aide de logiciels, de carte éléctroniques, et d'une petite touche de fabrication mécanique, mais ne disposant pas de ressources suffisantes et d'un temps limité, les résultats ne pouvait concurrencer ce qui se trouve dans le marché, nous dressons ci-dessous les quelques réussites que nous avons eus.

Bilan des travaux effectués

à l'heure du bilan nous nous rendons compte que beaucoups d'obstacles ont été franchis, mais restons tout de même lucides, et avouons que beaucoups reste à faire pour obtenir un simulateur digne de ce nom, on marque sans doute des points dans la diversité des travaux effectués

Nous avons conçue la partie avant d'un cockpit qu'on nomme tableau de bords, et ce a partir de simple pièces modulaire de taule pliée qu'on a assemblé pour former un tout, on a ensuite simulé des instruments et des jauges de mesure a l'aide d'un ecran et de programmes qui affichent les données collecté sur Flight Simulator, cette approche étant selon nous la mieux adapté car réutilisable et economiquement accessible vu la disponibilité des ecrans de nos jours, l'acquisition de vrai instruments d'affichage aéronautiques est tout bonnement à banir et la solution logiciel prédomine.

Nous avons réussis à interfacer la carte microcontroleur Mbed avec l'ordinateur hote grace à l'USB et transmettre des bits de données pour controler la simulation,

ce qui a permis d'implementer des controles analogiques pour restituer les vrais commandes d'un avion et augmenter la précision.

limites et perspectives

la limite premiere à été le temps qui n'est, il faut l'avouer jamais suffisant, vient s'ajouter à cela la nature du marché algérien basé sur l'importation, et ou trouver un

composant electronique ou une pièce d'assemblage devient un vrai parcours du combatant, certaines parties du projets assez ambitieuse qu'on a voulu entrependre sont tout bonnement tombé l'eau à cause de ce facteur, réalisable sur le papier, mais impossible sur le terrein, dès lors que trouver une rotule à 2 dégrés de liberté, ou un servo moteur, est comparable a chercherr une aiguille dans une botte de foin, et que les réponses qu'on a se limite à : rupture de stock, importation interrompu, cherchez chez mon voisin ... etc.

une des perspective de developpement de ce projet c'est d'apporter toutes les commandes nécessaires au tableau de bord que nous avons construit et de le peupler avec les circuits nécessaire pour restituer les controles complets d'un vrai Zlin, c'est aussi de le mettre au sein d'un vrai cockpit et d'introduire l'element de restitution du mouvement avec l'utilisation de plateformes mobiles de 2 voir plutard 3 dégrés de liberté, et par la suite intégrere un affichage par projection panoramique pour une immersion totale dans l'environnement de vol

Annexes

Annexe A

Dessins Techniques Du Tableau de bord

Disponible sur Documents tiers

Annexe B

Code source

Fournis sous forme électronique.

Références

1. J W R Taylor 1971, p.32.
2. a b J W R Taylor 1980, p,43.
3. J W R Taylor 1980, p,44.
4. London, Bruce (May 2007). "Flying Tigers rule the air". The Australian. Retrieved 2008-10-29.
5. a b Athas, Iqbal (October 2008). "Tigers bomb army base, power station". CNN. Retrieved 2008-10-29.
6. a b TamilNet (October 2008). "Tigers launch airstrike in Mannaar, Colombo". Retrieved 2008-10-29.
7. Jackson 2003, p. 113.
8. Transport Canada (September 2011). "Canadian Civil Aircraft Register". Retrieved 12 September 2011.
9. Flight International 16–22 November 2004, p. 42.
10. Hatch Flight International 29 November–5 December 1989, p. 45.
11. Flight International 16–22 November 2004, p. 53.Nacional.hr. Retrieved 3 April 2006.
12. Flight International 16–22 November 2004, p. 54.
13. Flight International 16–22 November 2004, p. 73.
14. a b c Jackson 2003, p. 114. "Zlin Z-242". Ministry of Defence : Slovenian Armed Forces. Retrieved 9 January 2012.
15. "EASA TYPE-CERTIFICATE DATA SHEET EASA.A.027 Z 42 - Series" European Aviation Safety Agency. 23 March 2007. Retrieved 0 February 2009.
16. http ://en.wikipedia.org/wiki/Flight_simulator
17. http ://www.microsoft.com/games/flightsimulatorx/
18. http ://msdn.microsoft.com/en-us/library/cc526981.aspx
19. http ://msdn.microsoft.com/en-us/library/cc526983.aspx
20. http ://msdn.microsoft.com/en-us/library/cc526948.aspx
21. http ://mbed.org/
22. http ://mbed.org/cookbook/Homepage
23. http ://mbed.org/handbook/Homepage
24. http ://mbed.org/handbook/USBHID
25. http ://mbed.org/nxp/lpc1768/
26. http ://en.wikipedia.org/wiki/Mbed_microcontroller
27. http ://www.signal11.us/oss/hidapi/

28. http ://alleg.sourceforge.net/
29. http ://www.siteduzero.com/
30. http ://www.developpez.com/
31. http ://www.solidworks.com/
32. PFE, juin 2011, Utilisation d'une carte FPGA dans la contribution à la réalisation d'un simulateur de vol

République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la recherche scientifique

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Annexe

Dessins Techniques

THEME :

CONCEPTION D'UN SIMULATEUR DE VOL

MATERIEL ZLIN 142

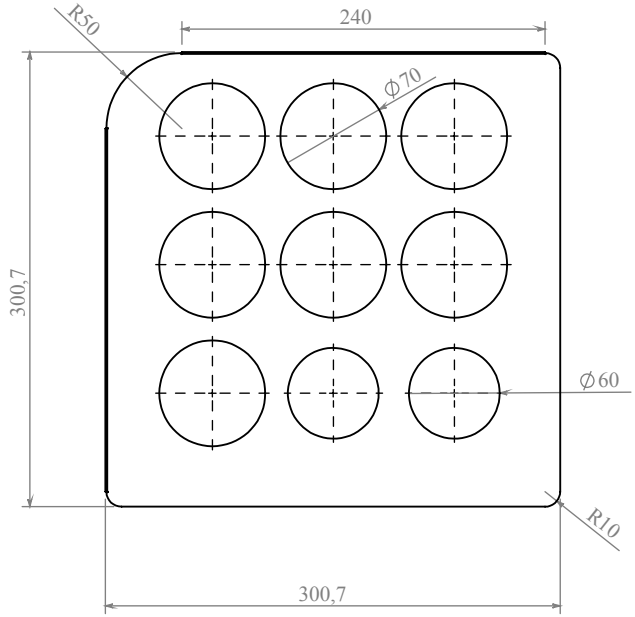
Présenté par :

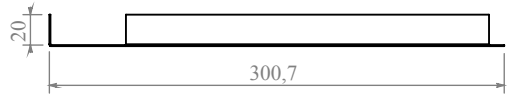
Mr. TALABOULMA Walid

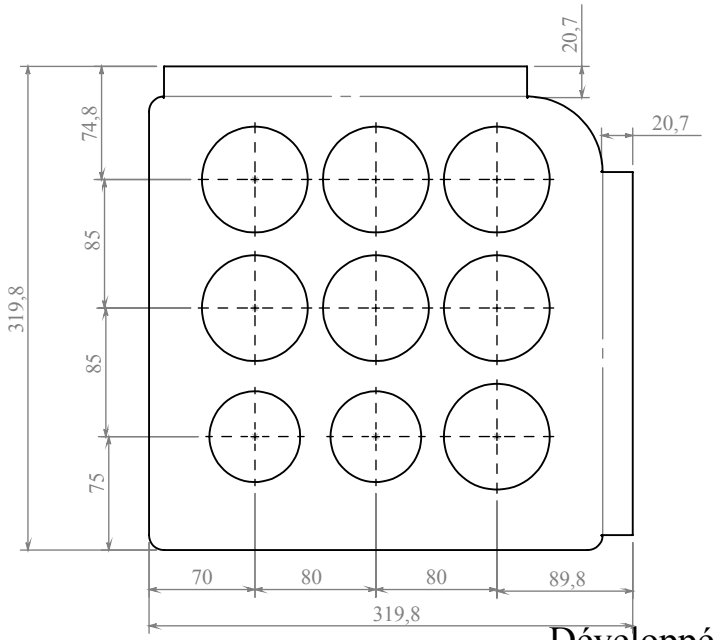
Promotion Septembre 2012

Ecole Nationale Polytechnique

10, Avenue Hacen Badi, El-Harrach, Alger





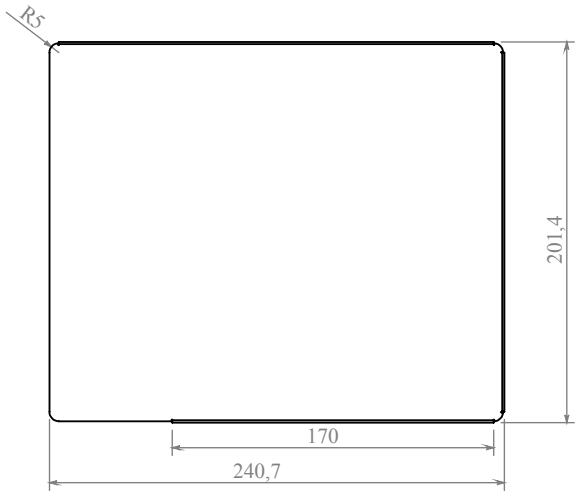


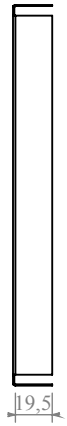
Développé

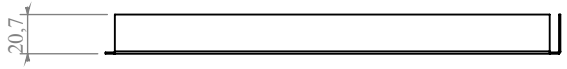
Tolérance générale : (+/- 0.5 mm)

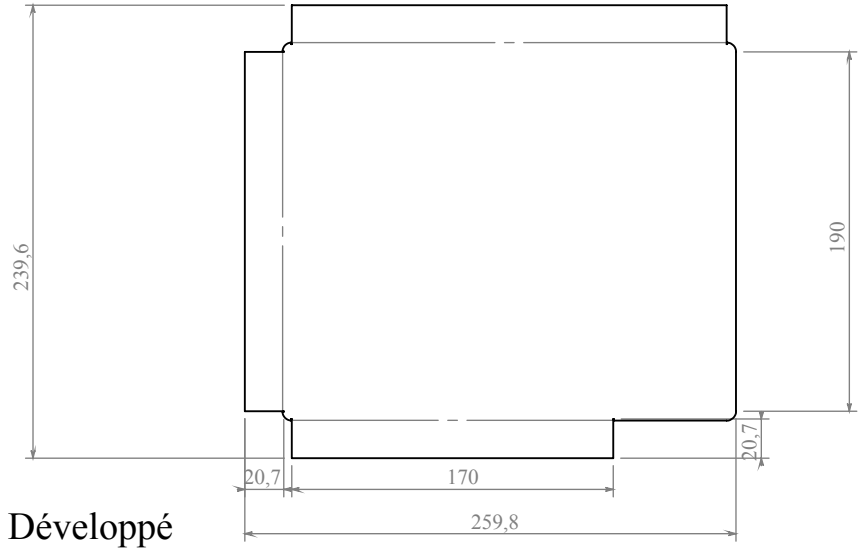
V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	LEFTINSTRUMENT	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4







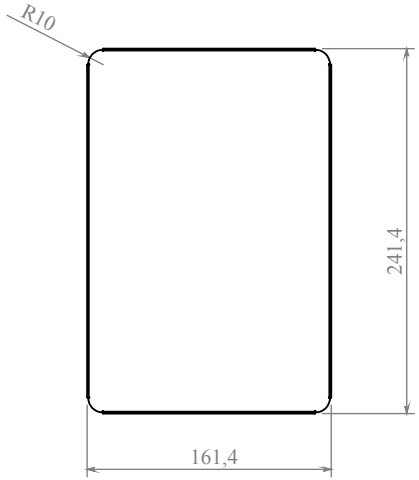


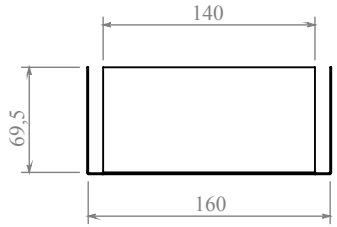
Développé

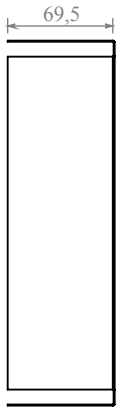
Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

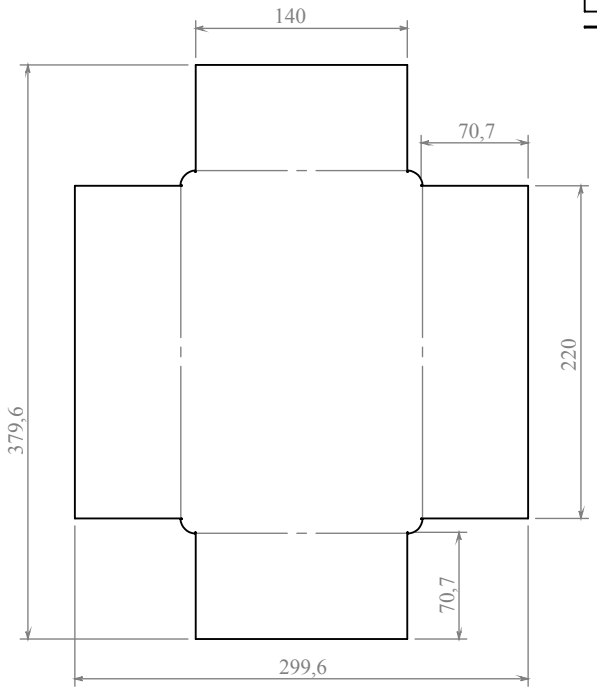
Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Déssiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	MIDDLE	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4







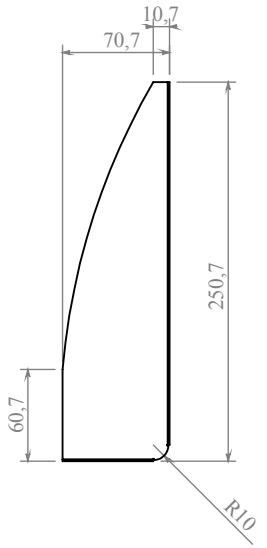
Développé

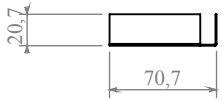


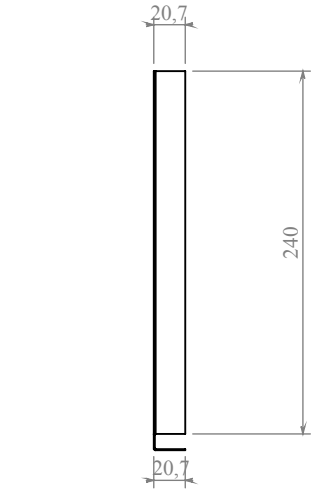
Tolérance générale : (+/- 0.5 mm)

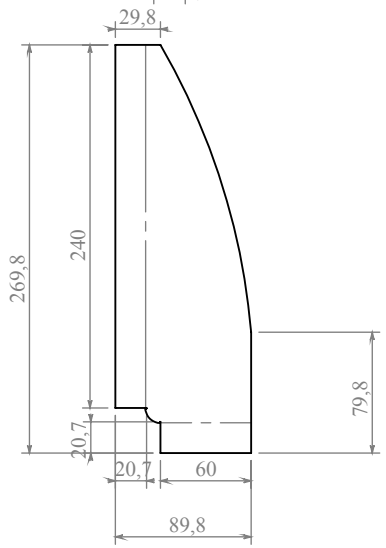
V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	BASLIGHT	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4





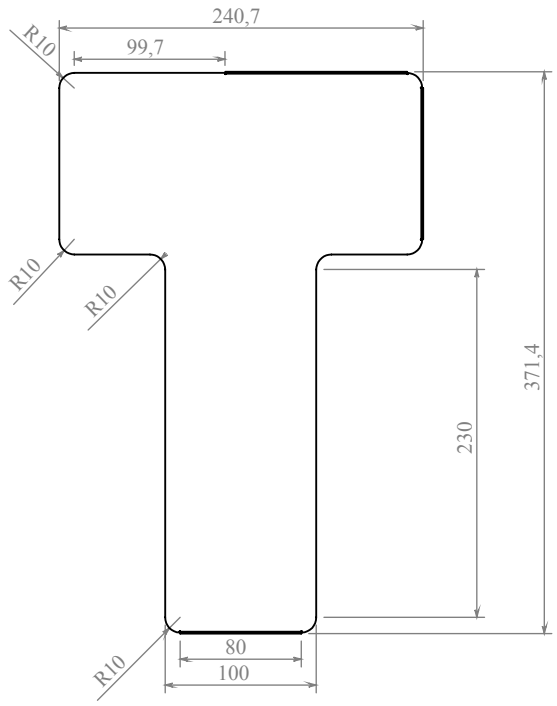


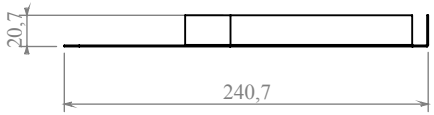


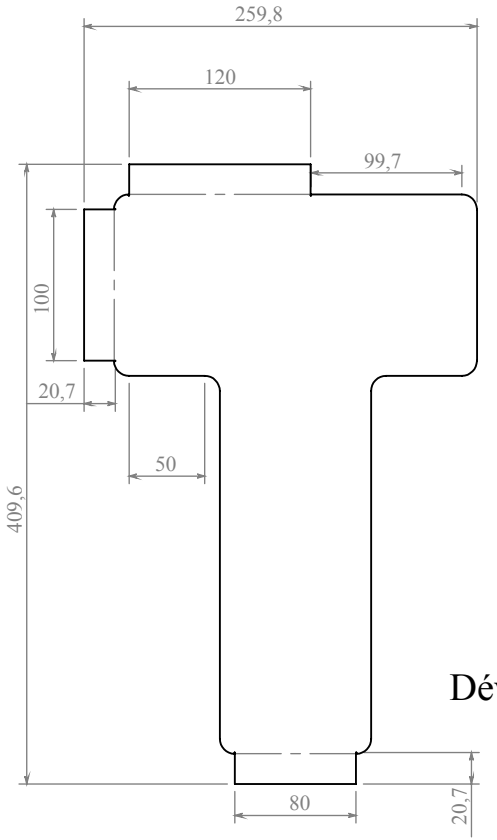
Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

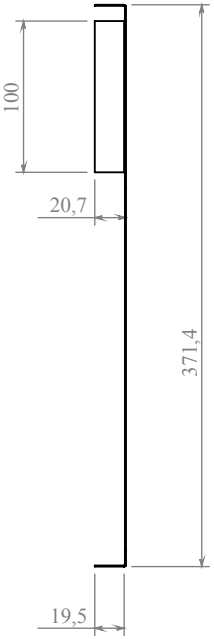
Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Déssiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	LEFT	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4







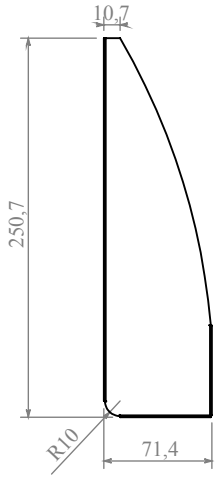
Développé

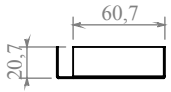


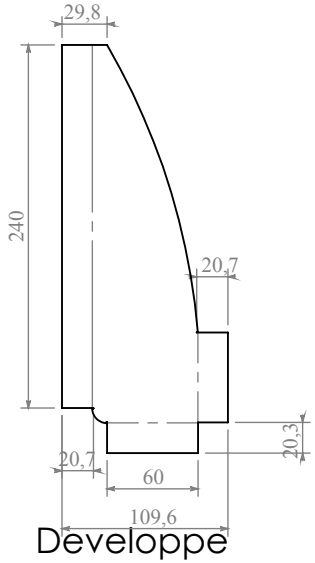
Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Déssiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	MIDDLEMOTOR	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4



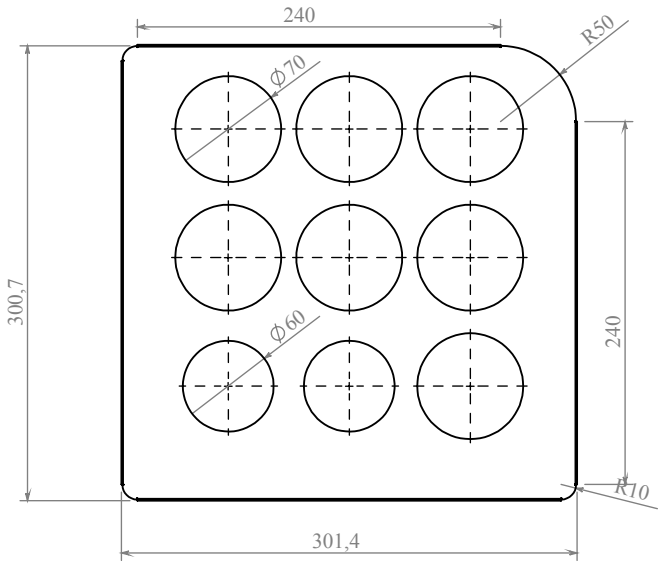


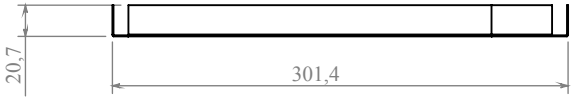


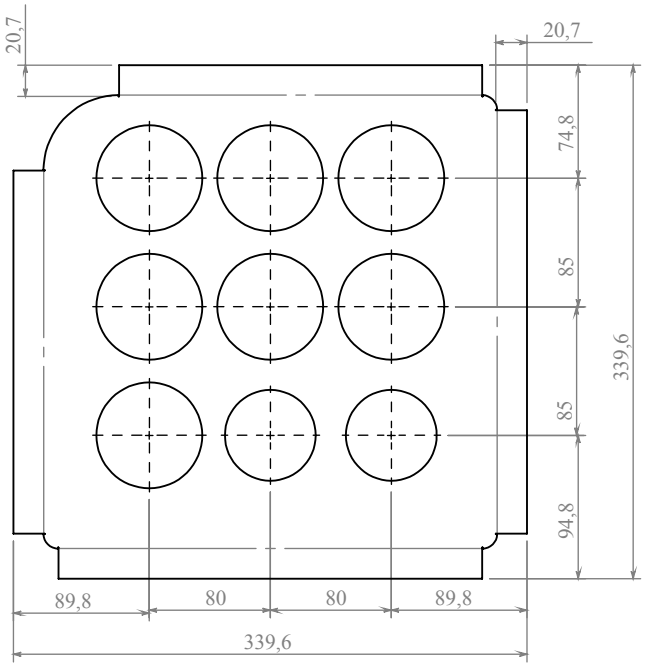
Tolérance générale : (+/- 0.5 mm)

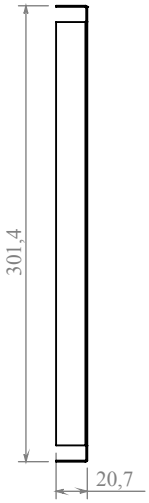
V de pliage : (V8)

Matiére: TOLE ACIER GALVA		Quantité:	Plan No:
Désiné par: TALABOULMA	Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	RIGHT		Echelle:1/4
			Réf:
	Projet: Chemin de cable		Format: A4







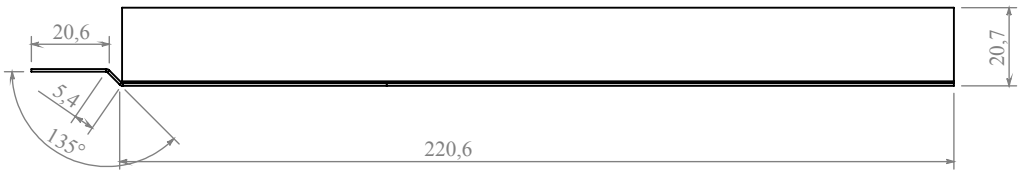


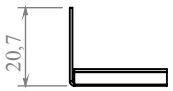
Développé

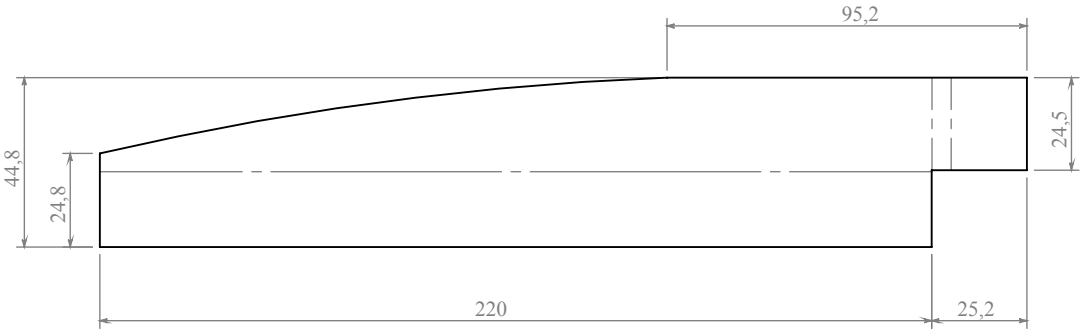
Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Déssiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	RIGHTINSTRUMENTS	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4





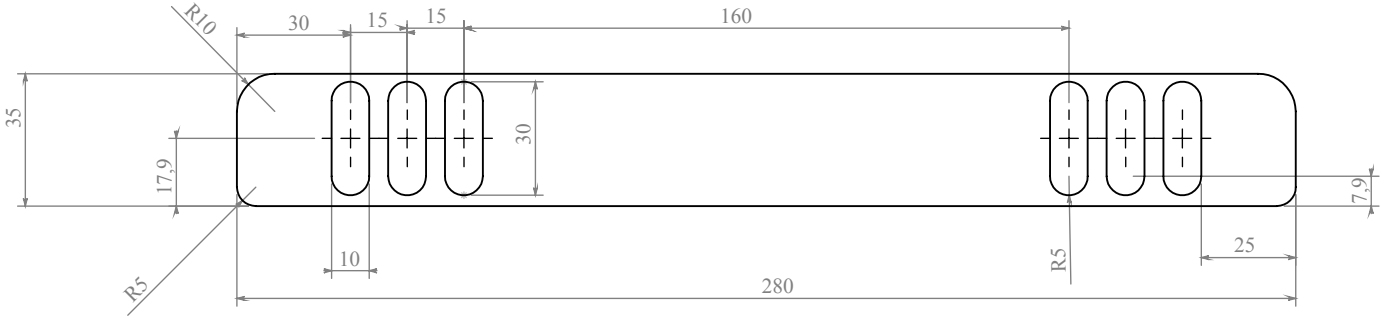


Développé

Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	TOPLEFT	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4

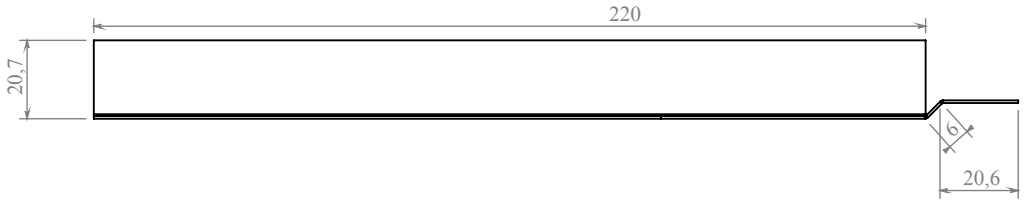


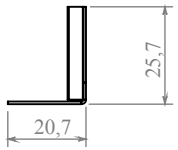
Développé

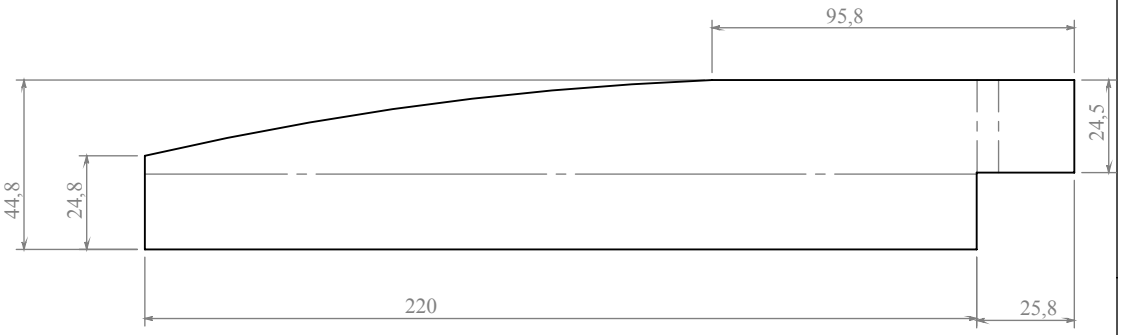
Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	TOPMIDDLE	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4





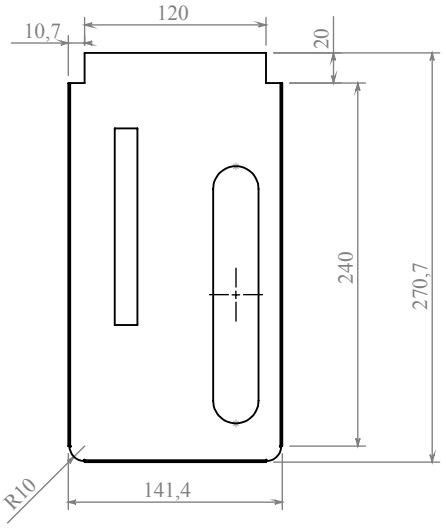


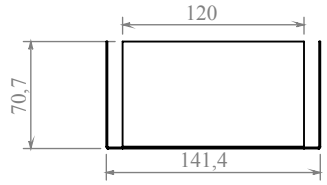
Développé

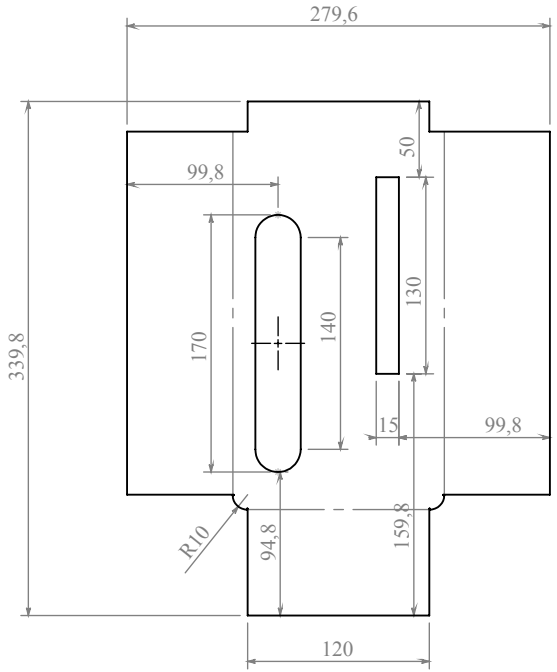
Tolérance générale : (+/- 0.5 mm)

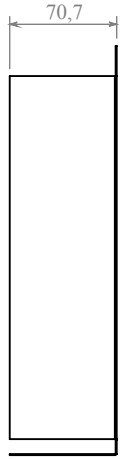
V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	TOPRIGHT	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4









Développé

Tolérance générale : (+/- 0.5 mm)

V de pliage : (V8)

Matiére: TOLE ACIER GALVA	Quantité:	Plan No:
Désiné par: TALABOULMA Date:Le 14/06/2012	Vérifié par : Date:Le	Ep:0.7 mm
telemetal	TRIM	Echelle:1/4
		Réf:
	Projet: Chemin de cable	Format: A4