الجمهورية الجزائرية الديمقراطية الشعبية
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique
Universite Paris Saclay

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

université
PARIS-SACLAY

Département d'électronique

Mémoire de projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en électronique

Simulateur de Vol : Interface Cockpit, CAN Avionique

Auteurs :

MEDJDOUB Omar **BOUCHELLAL Mohamed Larbi**

Présenté et soutenu le 12/07/2021, devant les membres du jury :

Président	Cherif	LARBES	Prof.	ENP, Algerie
Examinateur	Rabah	LOUALI	PhD.	EMP, Algerie
Promoteur	Samir	BOUAZIZ	Prof.	Paris Saclay, France
Promoteur	Mourad	ADNANE	Prof.	ENP, Algerie

ENP 2021

الجمهورية الجزائرية الديمقراطية الشعبية
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique
Universite Paris Saclay

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

université
PARIS-SACLAY

Département d'électronique

Mémoire de projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en électronique

Simulateur de Vol : Interface Cockpit, CAN Avionique

Auteurs :

MEDJDOUB Omar **BOUCHELLAL Mohamed Larbi**

Présenté et soutenu le 12/07/2021, devant les membres du jury :

Président	Cherif	LARBES	Prof.	ENP, Algerie
Examinateur	Rabah	LOUALI	PhD.	EMP, Algerie
Promoteur	Samir	BOUAZIZ	Prof.	Paris Saclay, France
Promoteur	Mourad	ADNANE	Prof.	ENP, Algerie

ENP 2021

ملخص ـ محاكيات الطيران هي أدوات أساسية في تدريب الطيارين والمهندسين في كافة المجالات بشكل عام. يتيح استخدام محاكيات الأنظمة تدريب المهندسين على التصميم والتحقق من الصحة والاختبار والصيانة. علم الطيران هو مجال يتم فيه استغلال المحاكيات متعددة الفيزياء على نطاق واسع ، لتدريب الطيارين على الاستخدام ، والمهندسين في الصيانة ، ودراسة عواقب الأعطال وتوقع حدوثها ، وما إلى ذلك. يتكون هذا المشروع من دراسة جدوى لزيادة واقعية محاكيات الطيران من خلال توصيل أجهزة كمبيوتر *LRU* (وحدات قابلة لإعادة الاستخدام للخط) تعتمد على ناقل اتصالات *CANaerospace* للفضاء. سيحاكي هذا البيانات الخاصة بالطائرة على ناقل اتصال حقيقي من بيئة محاكاة (للطائرة). سيمكن مثل هذا النظام من دراسة تكامل الأدوات الجديدة في بيئة المحاكاة قبل تثبيتها في الطائرة الحقيقية. كما أنه يجعل من الممكن تدريب مهندسي أنظمة المستقبل في عالم الطيران (إلكترونيات الطيران) دون الحاجة إلى اللجوء إلى طائرات حقيقية مكلفة للغاية ولا يمكن الوصول إليها بميزانية محدودة لمدرسة الهندسة.
كلمات مفتاحية ـ *CANaerospace* ، محاكيات الطيران ، موصل كان.

Abstract— Flight simulators are essential tools in the training of pilots and engineers in all fields in general. The use of systems simulators allows to train engineers in design, validation, test and maintenance. Aeronautics is a field where the use of multi-physical simulators are widely exploited, to train pilots for use, engineers for maintenance, to study the consequences of failures and anticipate their occurrence, etc. This project consists of a feasibility study to increase the realism of flight simulators by connecting real ECUs in the form of computing nodes using an LRU (Line Reusable Units) architecture based on the CAN aerospace communication bus. This will allow to emulate aircraft specific data on a real communication bus from a simulated environment (of an aircraft). The objective of such a system will allow to study the integration of new instruments on the simulated environment before implementing it in the real aircraft. It also allows to train future system engineers to the world of aeronautics (avionics) without having to use real aircraft, very expensive and unattainable budget for an engineering school.
Index-terms : Flight simulator, CAN Bus, CANaerospace.

Résumé— Les simulateurs de vol sont des outils essentiels dans la formation des pilotes et ingénieurs dans tous les domaines en général. L'utilisation des simulateurs de systèmes permet de former les ingénieurs à la conception, la validation, le test et la maintenance. L'aéronautique est un domaine où l'usage des simulateurs multi-physique sont largement exploités, pour former les pilotes à l'usage, les ingénieurs pour la maintenance, à l'étude des conséquences de pannes et anticiper leur apparition, etc. Ce projet consiste en l'étude de faisabilité pour augmenter le réalisme des simulateurs de vols en y connectant des calculateurs réels sous forme de nœuds de calcul utilisant une architecture LRU (Line Reusable Units) à base de bus de communication CAN aerospace. Cela permettra d'émuler des données spécifiques aux avions sur un bus de communication réel à partir d'un environnement simulé (d'un avion). Un tel système permettra d'étudier l'intégration de nouveaux instruments sur l'environnement simulé avant de le mettre en place dans le vrai avion. Cela permet aussi de former les futurs ingénieurs systèmes au monde de l'aéronautique (avionique) sans avoir recours à de vrais avions, très coûteux et au budget inaccessible pour une école.
Mots-clés : Simulateur de Vol, Bus CAN, CANaerospace.

Remerciements

Tout d'abord, on remercie Dieu pour nous avoir guidé et nous avoir donné la force et le courage nécessaire pour l'accomplissement de notre travail.

Ensuite, on remercie nos deux encadreurs M. Samir BOUAZIZ et M. Mourad ADNANE pour l'aide qu'ils nous ont apportée avant et durant tout notre projet, ainsi que M. Rabah SADOUN pour nous avoir initié à ce projet.

On tient également à remercier les membres du jury, M. Cherif LARBES et M. Rabah LOUALI pour avoir accepté d'examiner notre travail et de l'enrichir avec leurs propositions.

On remercie aussi nos familles et nos amis respectifs pour leur soutien et leur aide continuelle.

Enfin, nos remerciements s'adressent à toute personne ayant participé de près ou de loin à la réalisation de ce travail.

Dédicaces

On dédie ce travail à nos chers parents,
à nos frères et sœurs,
membres de nos familles,
à nos amis et connaissances,
à toute personne chère à notre cœur.

Table des matières

Liste des Tableaux

Liste des Figures

Acronymes

ADF Automatic direction finder.

ADI attitude directional indicator.

AFMS Airplane Flight Manual Supplement.

API Application Programming Interface.

ARINC Aeronautical Radio, Incorporated.

CAN Controller Area Network.

CAO Conception assistée par ordinateur.

CAS Calibrated Airspeed.

CDI Course Direction Indicator.

CHT Cylinder Head Temperature.

COTS Commercial off-the-shelf.

EADI Electronic ADI.

EFIS electronic flight instrument system.

EHSI Electronic HSI.

GPIO General Purpose Input/Output.

GPS Global Positioning System.

GS Ground speed.

HSI Horizontal situation indicator.

IAS Indicated Airspeed.

IDE Environnement de développement.

ILS Instrument Landing System.

ISR Interrupt Service Routine.

LRU Line-replaceable unit.

LSA Light sport aircraft.

MVP Minimum viable product.

ND Navigation-Display.

NDB Non Directioanl Beacon.

PFD Primary flight display.

PoC Proof of Concept.

QFE Field Elevation.

QNH Height Above Sea Level.

RMI Radio Magnetic Indicator.

SPI Serial Peripheral Interface.

TAS True Airspeed.

UART Universal Asynchronous Receiver Transmitter.

ULM ultra-léger motorisé.

USB Universal Serial Bus.

VHF very high frequency.

VOR Vertical Omnidirectional Range.

Introduction

La simulation de vol est devenue un outil indispensable dans tous les domaines de l'aéronautique, elle permet d'apporter une aide importante pour développer de nouveaux aéronefs et systèmes embarqués, enquêter sur des accidents par reproduction de situation, simuler des combats militaires en entraînant les pilotes.

Un simulateur de vol est composé de trois parties. La première partie est le logiciel permettant de simuler le modèle de vol de l'avion, les paramètres aérodynamiques et l'environnement.

Ensuite, vient l'extérieur de l'avion qui représente la plateforme parfois dynamique sur laquelle repose la cabine. Cette plateforme peut parfois effectuer des rotations sur les 3 axes (Roulie, Tangage, Lacet) et des translations verticales dans le but de simuler les conditions de mouvements réels d'un vol. Cette partie n'est pas toujours nécessaire et réservée aux séances de perfectionnement pour les vrais pilotes de lignes. La majorité des simulateurs de formations sont statiques.

Enfin, l'intérieur de la cabine où on reproduit le cockpit avec ses boutons de commandes des différents organes de l'avion et les instruments de vol et de contrôle (gestion moteur, instruments de navigation, radiocommunication . . .).

Dans de le cadre de notre projet, nous nous intéressons à un avion particulier, ZLIN-142 (Fernas) utilisé par l'école de formation de l'Ecole Supérieure de l'Air de Tafraoui. L'objectif du projet est de pouvoir anticiper la modernisation de cet avion au niveau de son avionique. Un simulateur de vol logiciel permet de modéliser cet avion sur la base d'un simulateur certifié Prepar3D (entreprise Lockheed Martin). Notre travail consistera à implanter un bus CAN avionique pour pouvoir étendre les instruments de bord, notamment pour implanter un instrument Glasscockpit Garmin G5.

Après l'intégration d'un vrai bus CAN Aéronautique sur le simulateur de cet avion, nous reproduisons l'interface FD (Fly Display) du GARMIN G5, un instrument de vol Glasscockpit qui sera également connecté au bus CAN avionique. Nous proposons deux versions pour cet instrument, une version émulée par l'utilisation d'une carte basse coût Raspberry pi avec un écran classique et une version utilisant un écran dédié à l'embarqué basé sur le produit "intelligent 4D Systems".

Notre travail est considéré comme un PoC (Proof of Concept), dont le but est de valider la faisabilité de ces nouvelles approches, afin de pouvoir les appliquer pour étudier la faisabilité

de futures modifications des vrais avions, mais aussi de pouvoir préparer la formations des futurs ingénieurs avioniques en utilisant des plateformes pédagogiques bas coût (reposant sur des simulateurs de vol).

Un des objectifs pour notre école est de préparer des travaux pratiques pour les formations futures de l'école pour l'enseignement des systèmes embarqués, en particulier l'architecture LRU largement utilisée dans le monde industriel avec le Bus Aéronautique. L'idée est donc de pouvoir disposer de plateformes expérimentales en conditions réalistes. Par exemple, La gestion des trames CAN avionique permettra à des élèves ingénieurs de pouvoir expérimenter dans un TP une réalité future qu'ils auront à vivre sur de vrais systèmes avions.

Chapitre 1

Présentation du projet

1.1 Introduction

Ce chapitre permet d'apporter une description générale de notre projet et de présenter le contexte dans lequel il s'inscrit.

On présentera les notions fondamentales nécessaires pour la compréhension des problématiques traitées et les objectifs attendus.

Enfin nous présenterons les éléments électroniques et informatiques qui composent notre projet et terminerons par la démarche adoptée pour réaliser notre travail.

1.2 La simulation de Vol

1.2.1 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 numérique. 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 utilisent ces simulateurs pour la formation continue des pilotes (nouveau type d'avions ou d'équipements, situations extrêmes, opérations militaires) et développer de nouveaux avions. Il existe aussi des simulateurs de vol en jeu vidéo, pour lesquels du matériel informatique grand public suffit.

La tendance depuis plusieurs années consiste à exploiter ces simulateurs de vol dans le domaine de l'ingénierie, pour vérifier la conformité d'instruments de vol, valider de nouveaux systèmes, expérimenter de nouvelles approches de maintenance, etc. Pour cela, il est possible d'intégrer de vrais instruments sur un bus de communication avionique. Les données capteurs et comportements de l'avion seront transmises par un simulateur de vol sous forme de données réalistes sur ce même bus de communication. Les systèmes ainsi ajoutés auront l'impression d'être véritablement dans un avion. Cette approche est désignée par HIL (Hardware In the Loop). C'est ce type d'approches qui intéressent fortement notre projet.

1.2.2 Composants d'un simulateur de vol

Les deux principaux composants d'un simulateur de vol sont le cœur logiciel de simulation des modèles de vols et de l'environnement d'une part, et l'ensemble matériel physique interfacé à ce logiciel d'autre part. La partie physique interactionnelle constitue les points d'échanges et de communications avec les systèmes avioniques, nécessaires pour le pilote mais aussi pour les calculateurs qu'on souhaite valider ou construire. L'avion virtuel, simulé, va recevoir des consignes, configurations et commandes du pilote par les organes de contrôles (sidestick ou yoke)

ou bien des ordres des calculateurs grâce à des ordres transmis sur le bus de communication (control by Wire).

1.2.2.1 Partie logicielle

La partie dite logicielle est composées de tous les algorithmes pour simuler le vol de l'avion simulé, mais aussi les modèles comportementaux des équipements (les capteurs et afficheurs ont leur propre dynamique), les modèles de l'environnement (typiquement l'atmosphère et son évolution, météo). On simule aussi les modèles et scénarios de trafic aérien, véhicules au sol, les échanges radio, les incidents, etc.

1.2.2.2 Partie matérielle

Le matériel d'interface avec le logiciel c'est tous les organes d'affichage (instruments, glass-cockpit) ainsi que les commandes (boutons, switchs, Yoke, etc). Certains instruments disposent d'interfaces haptiques comme le retour d'effort, la restitution sonore réaliste, les dispositifs de restitution de mouvement (proprioceptifs). On retrouvera toujours une combinaison de ces sous-parties quel que soit le niveau de professionnalisation du simulateur.

1.2.3 Les différents types de simulateurs

On distingue deux principaux types de simulateurs : Les simulateurs à usage professionnel ou les simulateurs à usage grand public (incluant le gaming).

1.2.3.1 Simulateur de vol porfessionnel

Simulateur de conception
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 de l'avion. 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 de l'environnement externe.

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 de pilotage du pilote mais plutôt pour faire des entraînements spécifiques destinés à un personnel déjà pilote (vérifier les compétences de respect

des procédures). L'objectif principal de ce simulateur est aussi l'acquisition de nouvelles connaissances pour les nouveaux avions ainsi que pour les procédures à suivre lors de certaines phases de vols partiellement connues .

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 a été confronté et d'en tirer des renseignements pour l'amélioration éventuelle des interfaces ou des procédures.

1.2.3.2 Simulateur de vol grand public

Simulateur pour ordinateur personnel

Les simulateurs de vol, comme un jeu, ont été parmi les premiers types de logiciels de simulation développés pour les ordinateurs individuels. Le simulateur d'avions de combat ont été les premiers logiciels de jeu avion pour le grand public. Ces simulateurs ont beaucoup évolué et sont devenus des outils de promotion des métiers de l'aéronautique pour le grand public. Ils peuvent aussi être utilisés pour confronter de vrais pilotes à des conditions de vols impliquant des novices dans un espace de simulation en réseau (VATSIM, IVAO). On rentre dans le domaine des simulateurs "Serious Games".

Simulateur pour console de jeux

Les simulateurs de vol sur consoles de jeu sont assez rares. La politique commerciale pour ces simulateurs n'est pas cohérente avec la logique commerciale des consoles de jeux, consistant à vendre des jeux sans possibilité d'ajouter des add-ons. Or dans les simulateurs de vol, l'évolutivité est inévitable de par l'évolution des aérodromes et des avions.

Le jeu sur console 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 à être simplistes et à conserver des sensations de pilotage, un côté arcade , mais néanmoins, ils visent à recréer le plus fidèlement possible la sensation d'un vol à vue (VFR).

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 marché des panneaux complets d'instruments fonctionnels ayant l'aspect des instruments réels, de tels simulateurs utilisent généralement un des logiciels présentés précédemment. Ces simulateurs privés permettent à leur développeur d'approcher et de toucher de plus près le pilotage de simulateurs professionnels.

1.3 Objectif et problématique

Nous visons la mise en place d'une plateforme de simulation qui a pour objectif de s'adresser aux futurs ingénieurs avionique et un peu aux élèves pilotes pour l'utilisation d'instruments de vols spécifiques. On va donc favoriser une approche consistant à utiliser une architecture matérielle la plus proche de la réalité. Pour cela, on va considérer l'architecture LRU, Bus CAN Avionique, comme étant la base dorsale de notre plateforme. Les simulateurs de vol construits en Algérie par des passionnés sont essentiellement destinés à un usage privé car il est très complexe de certifier un simulateur de vol pour la formation de pilotes. Les rares simulateurs dits professionnels sont dans des laboratoires de recherche. La plupart utilisent des protocoles standards hors avionique tels UART ou USB. Ce sont donc le plus souvent des assemblages de modules vendus avec des protocoles propriétaires (de fabricant de jeux) point à point. Les contraintes temps réel ne sont pas du tout un enjeu ni un objectif pour ces systèmes.

Une expérience a été réalisée à l'EMP Borj El Bahri, en 2011, la conception d'une plateforme orientée recherche pour l'expérimentation d'instruments et de calculateurs de vol en phase de laboratoire [1].

Les aéronefs réels embarquent des bus de communications, pas toujours facile à aborder lors de formations d'ingénieurs. Le Bus CAN Avionique pourrait être une bonne solution pour construire des plateformes pédagogiques, pour enseigner les architectures LRU. Ces bus présentent les avantages de réduire considérablement le câblage et son architecture de déploiement. D'autre part, le Bus CAN offre la possibilité de pouvoir mélanger des instruments réels tels que le PFD ou HSI dans un environnement simulé[2]. Le HIL [1] est donc une base fondamentale pour la conception des systèmes et leur ingénierie., Le bus CAN peut être déployé à plus de 40 mètres avec des débits intéressant de 500kbits/sec à 1Mbits/sec.

Par ailleurs, l'Algérie dispose d'une structure de montage des avions sous licence. C'est le Firnas-142 et Safir-43, dérivées des ZLIN-142 et ZLIN 143 respectivement sous licence. Ces avions sont utilisés pour l'entraînement des futurs pilotes. Ils disposent d'instruments de vol discrets et mécaniques, pas forcément en phase avec les cockpits des nouveaux avions qu'auront à piloter les futurs élèves pilotes de chasse. De plus, certains des instruments de vol déjà installés suivent la norme russe, comme l'Horizon Artificiel qui présente une inversion de couleur (marron = ciel,

bleu = sol) ce qui crée une confusion pour le pilote lors du vol et lors du passage à la norme occidentale (marron = sol, bleu = ciel), il est important de noter que cette différence d'instrumentation a contribué à l'écrasement du vol 498 de Crossac [2]. C'est pourquoi les pilotes instructeurs et les ingénieurs veulent étudier la possibilité de remplacer l'Horizon Artificiel mécanique par un instrument de vol électronique, type GARMIN G5. Cet instrument (Glasscockpit) répond parfaitement aux critères demandés et est largement utilisé dans les avions privés et sur des avions disponibles au sein des flottes aériennes militaires (information publique sur le web). La fonctionnalité PFD et HSI qui combinent les informations traditionnellement affichées sur plusieurs instruments électromécaniques en un seul affichage électronique. Les pilotes doivent se former sur l'utilisation de cet instrument au sol sur des simulateurs avant de l'utiliser durant les vols réels. En raison de son prix (2.500 USD pour la version certifiée), il n'est pas raisonnable d'installer de vrais instruments sur des simulateurs d'avions au sol. Cela explique l'intérêt de concevoir une solution moins coûteuse. C'est l'un des objectifs de notre projet de reproduire l'interface du GARMIN G5 et le relier au Bus CAN Avionique du simulateur.

1.4 Zlin 142

1.4.1 Introdution

Notre PFE s'inscrit dans le cadre d'une vision mixte : plateforme pédagogique d'ingénierie et outil d'aide à l'amélioration du ZLIN Z-142, qui est produit en Algérie par l'ECA (Entreprise de Construction Aéronautique) sous le nom de Firnas-142. Cet avion es vendu à des clients civils et militaires. Il est utilisé pour des missions de navigation aérienne, les opérations postales, la surveillance maritime et terrestre, l'évacuation sanitaire, etc. La mission principale de cet aéronef est l'entraînement des nouveaux pilotes.

Les détails et compartiments du cockpit se trouvent dans l'annexe A.

Figure 1.1 – Cockpit du ZLIN 142

1.4.2 Caractéristique 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

1.5 GARMIN G5

1.5.1 Introduction

Le kit Garmin G5 est un instrument de vol conçu pour l'aviation non certifiée, il peut être utilisé comme instrument de vol primaire combinant tous les instruments de vol : horizon, altimètre, variomètre, anémomètre, turn-coordinator, etc. On peut l'intégrer à une installation glass cockpit.

Le récepteur GPS intégré permet d'afficher la localisation ainsi que la vitesse précise de l'appareil.

Il a également une batterie Li-Ion qui fournit une autonomie de 4h en cas de défaillance

d'alimentation de l'avion.

Il se monte sur tous les types d'ULM/ LSA/ Gyro ou Planeurs.[3]

Figure 1.2 – GarminG5

1.5.2 G5 Alttitude Indicator

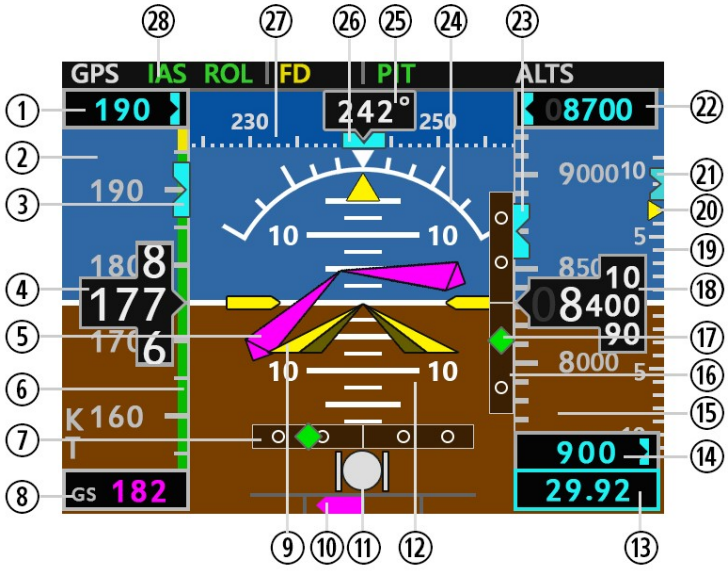


Figure 1.3 – PFD du GarminG5 [3]

L'instrument de vol électronique Garmin G5 configuré comme indicateur d'attitude est illustré à la Figure 1-8. Le G5 Alttitude Indicator affiche les paramètres suivants :

1. Vitesse de l'air sélectionnée.
2. Vitesse de l'air Indiquée.
3. Vitesse de l'air bug sélectionnée.
4. Vitesse de l'air actuelle.
5. Directeur de vol.
6. Echelle colore de la Vitesse de l'air.
7. CDI.
8. Vitesse par rapport au sol (GS).
9. Indicateur d'avion.
10. Indicateur du Turn Rate.
11. Indicateur Slip/skid.
12. Echelle d'indicateur d'attitude.
13. Indicateur Barométrique.
14. Vitesse verticale sélectionnée.
15. Altimètre.
16. Indicateur du GlideSlope.
17. Indicateur CDI/GS.
18. Altitude actuelle.
19. Echelle de la vitesse verticale.
20. Indicateur de la vitesse verticale.
21. Vitesse verticale bug sélectionnée.
22. Altitude selectionée.
23. Altitude bug sélectionnée.
24. Indicateur d'Atitude.
25. Cap actuel.
26. Cap bug sélectionné.
27. Indicateur du Cap.
28. Indicateur Autopilote.

1.5.3 G5 DG/HSI Indicator

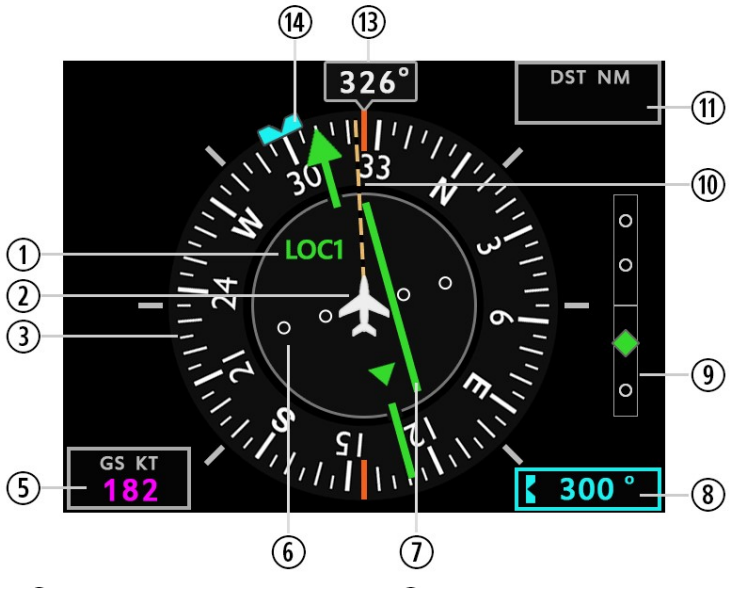


Figure 1.4 – GarminG5 HSI [3]

L'instrument de vol électronique Garmin G5 configuré comme indicateur DG/HSI est illustré à la Figure 1.4.
Le G5 DG/HSI affiche les paramètres suivants :

1. Source de navigation,
2. Symbole de l'avion,
3. Boussole,
4. Vitesse relative au sol,
5. Echelle GPS CDI,
6. Indicateur CDI,
7. Cap selectionné,
8. Indicateur GlideSlope,
9. Ground track actuel,
10. Waypoint distance,
11. Cap actuel,
12. Cap bug.

1.6 Outils électroniques et informatiques

On passe maintenant à la présentation des différents outils et notions électroniques et informatiques utilisés dans ce projet.

1.6.1 Simulateur de vol PREPAR3D

1.6.1.1 Présentation

Le simulateur que nous avons utilisé au cours de notre projet se nomme PREPAR3D et est développé par Lockheed Martin qui est un constructeur d'avions, c'est un simulateur certifié. Il succède en 2010 à Microsoft Flight Simulator après avoir acheté les droits de l'application à Microsoft. Il s'agit d'un simulateur de vol pour pc qui simule le fonctionnement d'un avion ainsi que son interaction avec l'environnement extérieur. Il permet d'effectuer des vols avec des conditions de réalisme poussées au maximum. Il permet également d'intégrer de la réalité augmentée avec un casque nommé Oculus. On retrouve à l'intérieur du cockpit d'un avion choisi exactement les mêmes commandes et instruments que dans un avion réel, comme on peut le voir dans la Figure 1.5 et avec un fonctionnement identique.[4]

Figure 1.5 – PREPAR3D

1.6.1.2 FSUIPC

Cet “Add-on” est basé sur l’API de Peter Dowson : FSUIPC, qui permet la communication avec PREPAR3D en temps réel [5] . L’échange de données avec PREPAR3D utilise une bibliothèque de liens dynamiques spécifique : FSUIPC.dll (Flight Simulator Universal InterProcess Communication). Cette bibliothèque réalisée par Peter Dowson et téléchargeable depuis son site [5] , peut être installée simplement en la copiant dans le répertoire ”PREPAR3D Modules”. FSUIPC permet aux applications externes de lire et d’écrire des données dans PREPAR3D en temps réel en exploitant le mécanisme de communication inter-processus (d’où son nom) en utilisant un tampon de 64 Ko [5] . Ce dernier contient une grande quantité de données qui affectent différents aspects de la simulation de vol (contrôle, état dynamique, pilote automatique, météo, etc.). Pour lire ou écrire une donnée, il faut connaître son adresse dans le buffer FSUIPC, son format et les conversions nécessaires. Par exemple, la vitesse verticale est lue comme un entier signé (32 bits), à 0x02C8, qui doit être divisé par 256 pour le convertir en m/s et pour y écrire on suit l’opération inverse [5] [1].

1.6.2 Microcontroleur

Un microcontrôleur est un circuit intégré comportant un processeur, une mémoire et quelques périphériques. Sa taille peu encombrante ainsi que sa faible consommation d’énergie en font un composant principal dans les systèmes embarqués. Nous avons travaillé avec le Mbed LPC1768 de ARM. C’est un microcontrôleur basé sur un processeur ARM Cortex-M3 32 bits avec une

horloge de 96 MHz. Il inclut une mémoire flash de 512 Ko et une mémoire RAM de 32 Ko.

Figure 1.6 – Mbed LPC1768

Il comprend diverses interfaces d'entrées sorties tels que : Ethernet, USB, SPI, , UART, CAN, 6 sorties PWM, 6 Convertisseurs analogiques numériques et des ports GPIO. Cette carte est programmable en langage C/C++. Il suffit de la brancher à un ordinateur en USB et de lui transmettre le programme compilé. On peut écrire et compiler le programme sur le logiciel Keil μVision qui est un outil professionnel installable ou bien directement sur un compilateur en ligne, le Mbed Compiler [6].

1.6.3 Raspberry Pi

Le Raspberry Pi est un nano-ordinateur monocarte à processeur ARM de la taille d'une carte de crédit conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation Raspberry Pi.

En termes de puissance de calcul, le Raspberry Pi 4 est livré avec un nouveau processeur ARM quadricœur à 1,5 GHz, 2Gb de RAM.

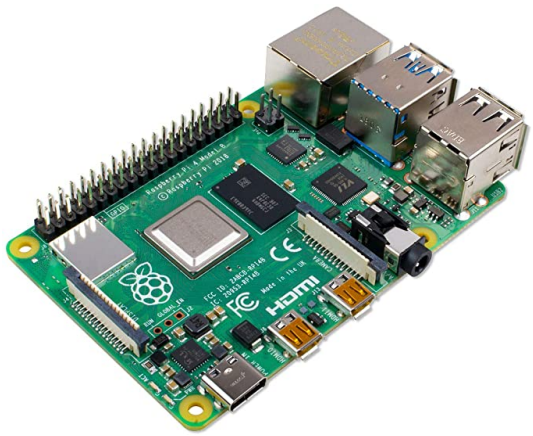


Figure 1.7 – Raspberry Pi 4

Il comprend diverses interfaces d'entrées sorties tels que : Ethernet, USB, SPI, I2C, UART, CAN, sorties PWM, Convertisseurs analogiques numériques et des ports GPIO. Le Raspberry Pi permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais il fonctionne également avec le système d'exploitation Microsoft Windows : Windows 10 IoT Core, Windows 10 on ARM.[7]

1.6.4 Instruments

On passe maintenant aux différents instruments utilisés dans notre projet.

1.6.4.1 Ecran 4D System

Ce module tactile capacitif uLCD-35DT est doté d'un écran LCD TFT couleur de 3,5" (89 mm) et est livré avec un cadre autocollant. Il dispose de stockage de mémoire microSD (carte non incluse), de GPIO et de ports de communication ainsi que de minuteurs de résolution de plusieurs millisecondes et de génération audio. Il est piloté par le processeur graphique 4D Systems Diablo16, qui offre un éventail de fonctionnalités et d'options pour tout concepteur / intégrateur / utilisateur. L'écran est compatible avec l'IDE Workshop4 et ses 4 environnements de développement différents, offrant à l'utilisateur une multitude d'options pour programmer et contrôler son système.[8]

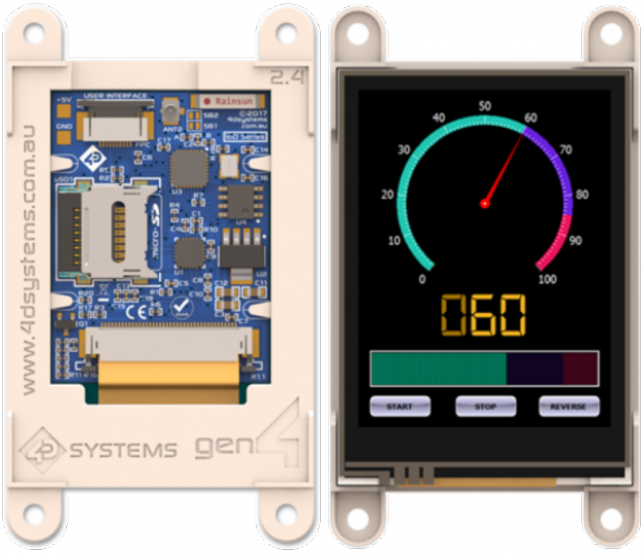


Figure 1.8 – Ecran 4D Systems Gen4

1.6.5 Écran Adafruit 3.5pouce

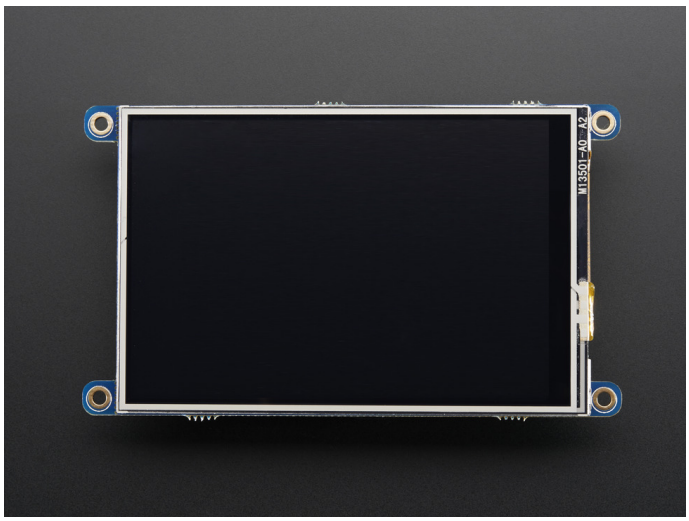


Figure 1.9 – Ecran Adaftuit 3.5pouce

L'écran Adaftuit dispose d'un écran de 3,5 pouces avec 480x320 pixels, couleur 16 bits et d'une super résistivité tactile, conçu spécifiquement pour fonctionner avec les nouveaux "2x20 connecteur" Raspberry Pi 3 et 4.
L'écran utilise les broches SPI matérielles (SCK, MOSI, MISO, CE0, CE1) ainsi que les GPIO 25 et 24. Toutes les autres GPIO sont inutilisées.[9]

1.6.6 Protocoles de communication

Afin de faire communiquer nos différents instruments, nous avons eu recours à plusieurs protocoles que nous présentons ci-dessous.

1.6.6.1 Bus CAN

Le bus de données CAN est un protocole de communication série qui prend en charge le contrôle distribué en temps réel avec un haut niveau de sécurité.

Il s'agit du bus principal qui relie notre simulateur (P3D) aux différentes branches LRU qui constituent notre projet, d'abord on utilise un module USB-CAN pour la partie simulateur sur pc, le mcp2551 pour relier avec le microcontrôleur Mbed et enfin le mcp2515 + mcp2551 pour relier la raspberry pi au bus .

1.6.6.2 Bus CAN Avionique

1.6.6.2.1 CANaerospace

CANaerospace est une définition de protocole / format de données extrêmement légères qui a été conçu pour la communication hautement fiable de systèmes basés sur micro-ordinateur dans les applications aéroportées via CAN.

Le but de cette définition est de créer une norme pour les applications nécessitant une surveillance efficace des flux de données et synchronisation facile des délais dans les systèmes redondants. La définition est maintenue largement ouverte pour permettre la mise en œuvre de types de messages et protocoles.

CANaerospace peut être utilisé avec CAN 2.0A et 2.0B[10].[11]

1.6.6.2.2 ARINC 825

ARINC 825 est une spécification de protocole pour l'industrie aéronautique, gérée par arinc. Elle spécifie la communication fondamentale au sein des sous-systèmes basés sur le bus CAN, elle offre des mécanismes d'adressage, des mécanismes de communication, une structure de service, des descriptions de profils et bien plus encore. L'importance de ce protocole est démontrée par le fait qu'Airbus et Boeing ont lancé le groupe de travail technique CAN pour spécifier des méthodes et des protocole pour l'utilisation générale du CAN, y compris l'interopérabilité entre les sous-systèmes, cette activité a débouché sur l'ARINC 825. [11]

1.6.6.3 Bus SPI

C'est un protocole de communication série, synchrone, bidirectionnel et full duplex. Il permet de relier le joystick au premier microcontrôleur.

1.6.6.4 Bus I2C

C'est un protocole de communication série, synchrone, bidirectionnel et half duplex permettant la communication entre certains périphériques.

1.6.7 Logiciels et interfaces de programmation

Nous avons utilisé plusieurs logiciels et interfaces de programmation pour réaliser notre travail, on présente les plus importants ci-dessous :

1. Bibliothèque QT :Qt est une bibliothéque logicielle orientée objet(API) développée en C++ par Qt Development Frameworks, filiale de Digia.Elle est fournie à l'origine par la société norvégienne Troll Tech, rachetée par Nokia en février 2008 puis cédée intégralement en 2012 à Digia (www.qt.io).
   Elle fournit un ensemble de classes décrivant des éléments graphiques (widgets) et des éléments non graphiques : accès aux données (fichier, base de données), connexions réseaux (socket), gestion du multitâche (thread), XML, etc.
   Qt permet la portabilité des applications (qui n'utilisent que ses composants) par simple recompilation du code source. Les environnements supportés sont les Unix (dont Linux), Windows et Mac OS X. Les deux interface Garmin G5 et 9 instruments sont développé en utilisant cette bibliothèque.[12]
2. PcanView : PcanView est un logiciel fourni avec le driver de l'adaptateur USB-CAN qui permet d'observer en temps réel l'échange des trames. On peut y lire les données, leur taille, leur identifiant, la période d'envoi ainsi que le nombre de fois qu'une donnée a été transmise avec cet identifiant. Il permet également d'envoyer des données sur le bus en spécifiant l'identifiant, la taille du message, le message et la période entre chaque envoi. Nous l'avons utilisé principalement pour faire du monitoring de toutes les données qui circulent sur le bus.
3. SolidWorks : SolidWorks est un logiciel de CAO développé par Dassault Systèmes SE. Il s'agit d'un logiciel permettant la conception en 3D de pièces avec une multitude de paramètres. Nous l'avons utilisé pour concevoir des pièces dont nous avions besoin pour notre projet. A la fin de la conception, le logiciel génère, entre autres, un fichier que l'on peut passer à une imprimante 3D pour obtenir la pièce désirée.
4. Keil μvision : Afin de programmer notre Mbed, nous avons utilisé le logiciel Keil μVision de ARM qui est un environnement de développement complet permettant d'éditer le code

et de le compiler. Une fois la compilation terminée, un fichier ".bin" est généré et transféré sur la carte. Il faut ensuite flasher cette dernière pour que le programme soit chargé dans sa mémoire flash.

1.6.8 Systèmes temps-réels

La classe EventQueue, de l'Mbed OS 5.15.6, fournit une file d'attente flexible pour la planification des événements. On peut l'utiliser pour la synchronisation entre plusieurs threads ou pour contrôler des événements hors du contexte d'interruption (exécution différée d'opérations de sécurité chronophages à éviter à mettre dans les ISR) [13].

L'envoi et la réception des messages CAN ne peut pas s'exécuter dans un contexte d'interruption ou de thread, cela provoquerait une erreur d'exécution et arrêterait l'exécution du programme.

1.7 Démarche de conception

1.7.1 Partie électronique

Comme il a été précisé, l'architecture électronique de notre système repose sur un bus CAN sur lequel sont greffés tous les modules logiciels et matériels communiquant des données venant du simulateur (capteurs, états des systèmes ...) et permettant de transmettre des données des différents organes de commandes vers le simulateur. La figure ci-dessous illustre l'architecture générale.

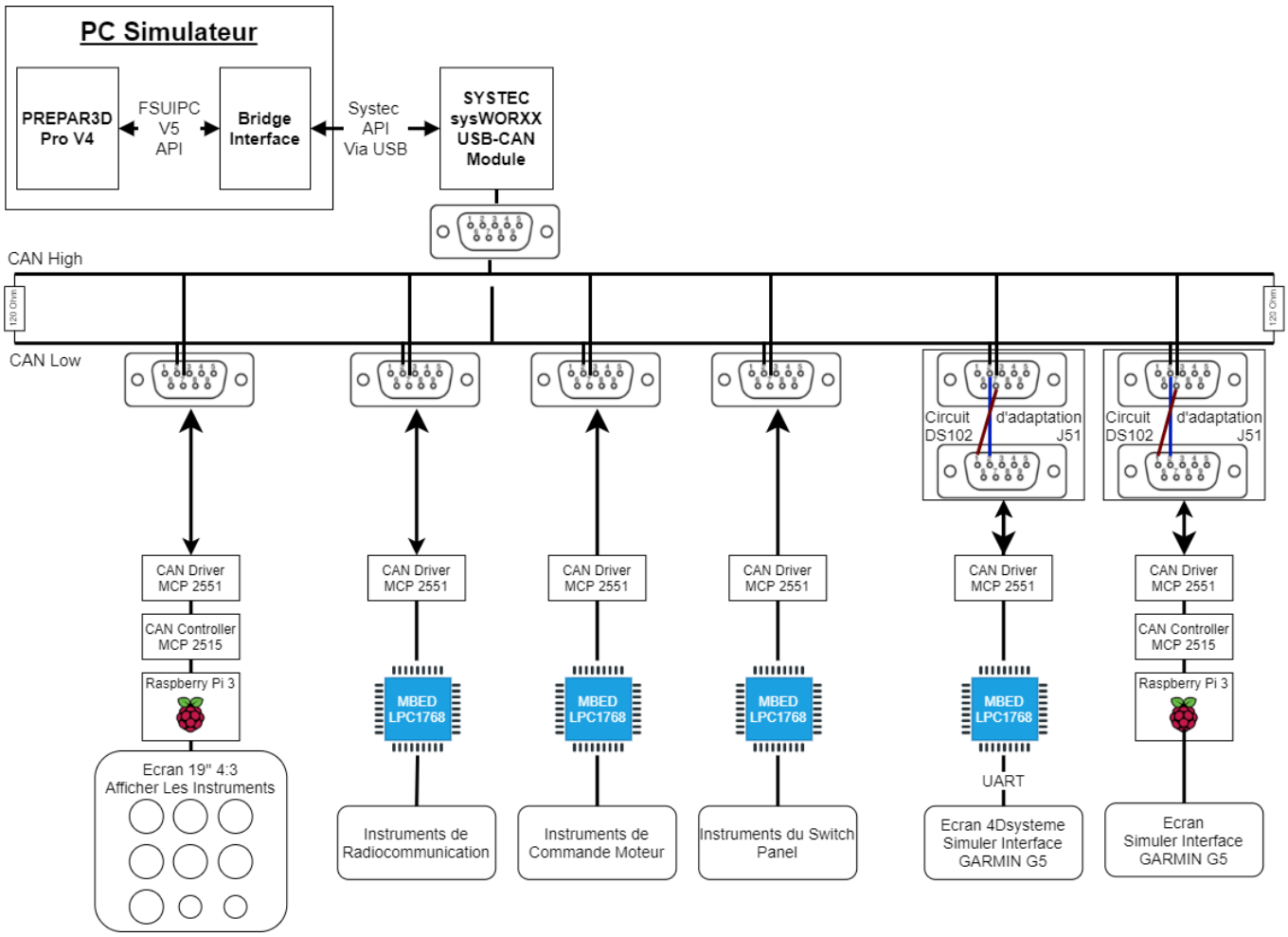


Figure 1.10 – Schéma Global de la partie électronique

Nous avons réalisé plusieurs modules interdépendants pour répondre aux besoins du projet.

— Concevoir une Interface Utilisateur “Bridge” qui relie entre le simulateur P3D et l’interface du cockpit :
  — Affiche, lit et écrit les données de vol sur le simulateur via l’API FSUIPC.
  — Envoie et reçoit les données de de vol via l’API du contrôleur SYSTEC (ou via serial à un Mbed qui fait la conversion Serial-CAN).

— Regrouper les boutons et instruments en noeuds par catégorie :
  — Les 9 instruments de vol, de navigation et de contrôle moteur
  — Radiocommunication
  — Commande moteur
  — Switch panel + Joystick

  - GARMIN G5 version raspberry pi
  - GARMIN G5 version Mbed + 4DSystem
- Implémenter le Bus CAN Avionique sur tous les noeuds du système
- Mettre en place un noeud Mbed de debuggage des trames CAN.
- Mettre en place un nœud Mbed + Écran 4dSystem pour donner la main à l'instructeur de vol pour gérer la simulation.

1.7.2 Partie mécanique

- Réalisation d'un modèle de tableau de bord de Zlin en forex pour supporter les instruments réalisés dans le cadre du projet..
- Conception et impression des pièces 3D sous solidworks avec une imprimante 3D.

1.7.3 Fusion des deux parties

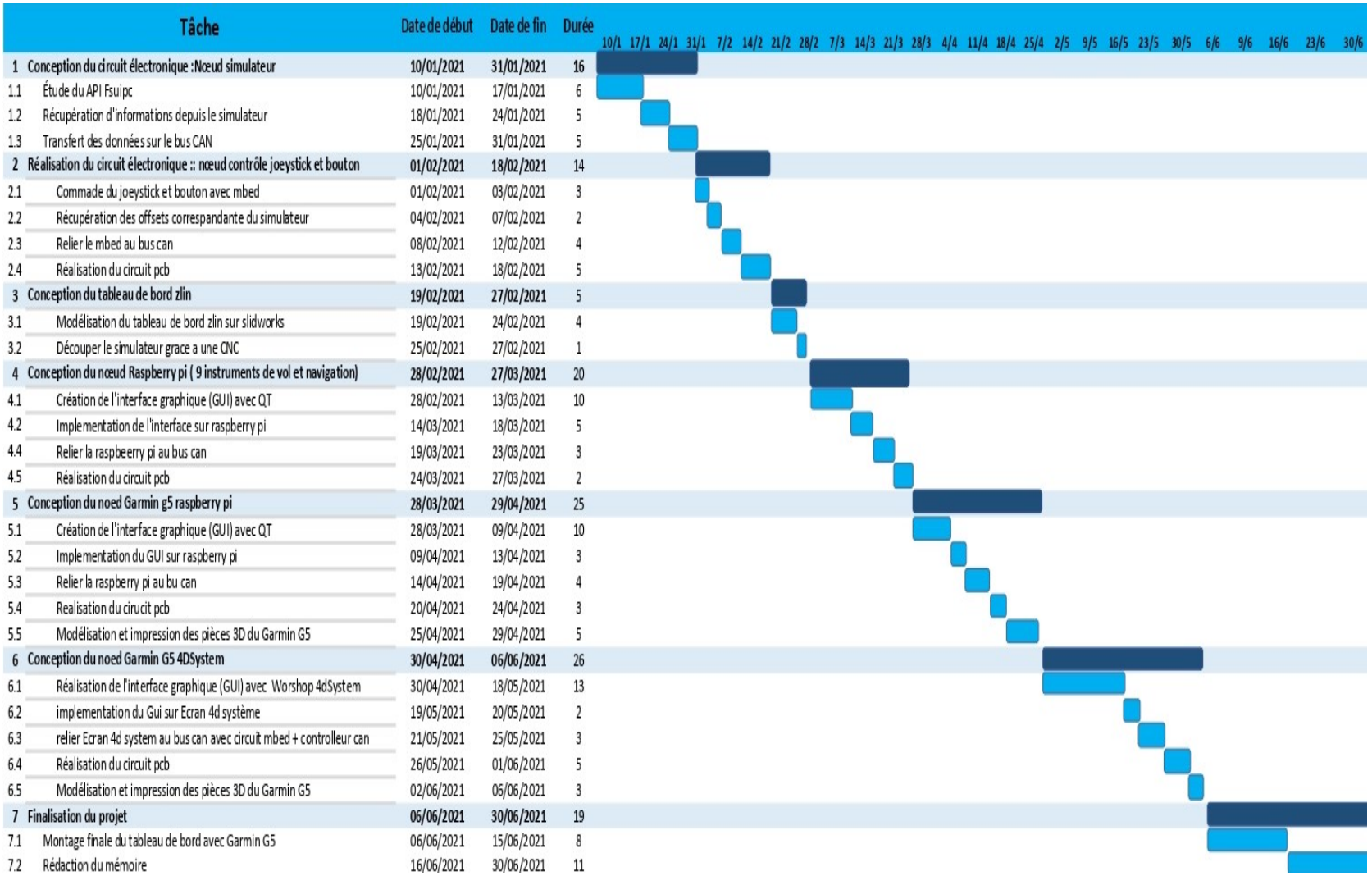
Finaliser le tableau de bord en relient tous les noeuds électroniques à la planche forex.

1.7.4 Tableau des tâches

Afin d'avancer dans notre travail de façon optimale, nous avons eu recours à un outil de gestion de projet. Nous avons utilisé le diagramme de Gantt pour nous représenter nos différentes tâches avec les durées de chacune. La Figure 1.11 dans la page suivante, présente le diagramme de Gantt prévisionnel établi au début du projet. Nous verrons dans le dernier chapitre le diagramme effectif et nous expliquerons les raisons des changements du planning.

1.7.5 Conclusion

À travers ce chapitre, nous avons clairement compris les objectifs du projet, les étapes pour réaliser le projet et les contraintes auxquelles le projet doit répondre. Nous avons également vu les différents éléments qui le composent.



	Tâche	Date de début	Date de fin	Durée
1	**Conception du circuit électronique :Nœud simulateur**	**10/01/2021**	**31/01/2021**	**16**
1.1	Étude du API Fsuipc	10/01/2021	17/01/2021	6
1.2	Récupération d'informations depuis le simulateur	18/01/2021	24/01/2021	5
1.3	Transfert des données sur le bus CAN	25/01/2021	31/01/2021	5
2	**Réalisation du circuit électronique :: nœud contrôle joeystick et bouton**	**01/02/2021**	**18/02/2021**	**14**
2.1	Commade du joeystick et bouton avec mbed	01/02/2021	03/02/2021	3
2.2	Récupération des offsets correspandante du simulateur	04/02/2021	07/02/2021	2
2.3	Relier le mbed au bus can	08/02/2021	12/02/2021	4
2.4	Réalisation du circuit pcb	13/02/2021	18/02/2021	5
3	**Conception du tableau de bord zlin**	**19/02/2021**	**27/02/2021**	**5**
3.1	Modélisation du tableau de bord zlin sur slidworks	19/02/2021	24/02/2021	4
3.2	Découper le simulateur grace a une CNC	25/02/2021	27/02/2021	1
4	**Conception du nœud Raspberry pi (9 instruments de vol et navigation)**	**28/02/2021**	**27/03/2021**	**20**
4.1	Création de l'interface graphique (GUI) avec QT	28/02/2021	13/03/2021	10
4.2	Implementation de l'interface sur raspberry pi	14/03/2021	18/03/2021	5
4.4	Relier la raspbeerry pi au bus can	19/03/2021	23/03/2021	3
4.5	Réalisation du circuit pcb	24/03/2021	27/03/2021	2
5	**Conception du noed Garmin g5 raspberry pi**	**28/03/2021**	**29/04/2021**	**25**
5.1	Création de l'interface graphique (GUI) avec QT	28/03/2021	09/04/2021	10
5.2	Implementation du GUI sur raspberry pi	09/04/2021	13/04/2021	3
5.3	Relier la raspberry pi au bu can	14/04/2021	19/04/2021	4
5.4	Realisation du cirucit pcb	20/04/2021	24/04/2021	3
5.5	Modélisation et impression des pièces 3D du Garmin G5	25/04/2021	29/04/2021	5
6	**Conception du noed Garmin G5 4DSystem**	**30/04/2021**	**06/06/2021**	**26**
6.1	Réalisation de l'interface graphique (GUI) avec Worshop 4dSystem	30/04/2021	18/05/2021	13
6.2	implementation du Gui sur Ecran 4d système	19/05/2021	20/05/2021	2
6.3	relier Ecran 4d system au bus can avec circuit mbed + controlleur can	21/05/2021	25/05/2021	3
6.4	Réalisation du circuit pcb	26/05/2021	01/06/2021	5
6.5	Modélisation et impression des pièces 3D du Garmin G5	02/06/2021	06/06/2021	3
7	**Finalisation du projet**	**06/06/2021**	**30/06/2021**	**19**
7.1	Montage finale du tableau de bord avec Garmin G5	06/06/2021	15/06/2021	8
7.2	Rédaction du mémoire	16/06/2021	30/06/2021	11

Figure 1.11 – Diagramme de Gantt prévisionnel

Chapitre 2

CAN BUS CANaerospace (ARINC 825)

2.1 Introduction

Ce chapitre va présenter le coeur du travail à savoir le protocole de communication Bus CAN Avionique **CANaerospace**.
Nous introduirons d'abord le protocole Bus CAN, avant de passer à l'explication détaillée de l'implémention haut niveau du bus.

2.2 BUS CAN

Le bus de données CAN (Controller Area Network) est un protocole de communication série qui prend en charge le contrôle distribué en temps réel avec un haut niveau de sécurité.

Introduit dans les années 1980 par Robert Bosch GmbH, le bus CAN a d'abord été installé dans les voitures Mercedes-Benz. Pour améliorer la sécurité et le confort, de nombreuses unités de commande électronique (UCE), telles que le système de freinage antiblocage, la gestion du moteur, la commande de traction, la commande de climatisation, le verrouillage central des portes et les commandes des sièges et des rétroviseurs électriques, ont été ajoutées dans les automobiles. Pour interconnecter ces UCE et réduire les gros faisceaux de câbles, le bus CAN a été mis en place. Il est capable de fonctionner de manière fiable, même dans des environnements difficiles.

En raison de son succès dans l'automobile, la technologie du bus CAN a attiré l'attention des fabricants d'autres secteurs, notamment le contrôle des processus, le textile et les instruments médicaux. En raison de sa polyvalence, Airbus a ouvert la porte au bus CAN dans le superjumbo A380.

2.2.1 Format de données

La trame de données du bus CAN est constituée de sept champs de bits différents : Début de trame, Arbitrage, Contrôle, Données, CRC, ACK et Fin de trame.

— **SOF** : Un message de trame du base CAN commence par le bit de départ appelé Start of Frame (SOF). Il s'agit d'un seul bit "dominant".

— **RTR** : Le champ d'arbitrage, qui se compose de l'identificateur et du bit de demande de transmission à distance (RTR), il est utilisé pour faire la distinction entre la trame de données et la trame de demande de données appelée trame distante.

— **Control Field** : Le champ de contrôle contient le bit d'extension d'identificateur (IDE), qui est le bit réservé pour faire la distinction entre la trame de base CAN (CAN 2.0A), la trame étendue CAN (CAN 2.0B) et le code de longueur de données (DLC), qui est utilisé pour indiquer le nombre d'octets de données suivants dans le champ de données. Si le message est utilisé comme une trame distante, le DLC contient le nombre d'octets de données demandés.

— **Data** : Le champ de données qui suit peut contenir jusqu'à 8 octets de données. Il représente le contenu réel de l'information dans le message.

— **CRC** : L'intégrité de la trame est garantie par la somme suivante du contrôle de redondance cyclique (CRC).

— **Slot ACK** : Le champ d'accusé de réception (ACK) comprend le slot ACK et le délimiteur ACK. Dans le champ ACK, la station émettrice envoie deux bits récessifs. Un récepteur, qui a reçu correctement un message valide, le signale à l'émetteur en envoyant un bit dominant pendant le slot ACK (il envoie "ACK"). Toutes les stations ayant reçu la séquence CRC correspondante le signalent dans le slot ACK en substituant au "bit récessif" de l'émetteur un "bit dominant".

— **Délimiteur ACK** : Le délimiteur ACK est le deuxième bit du champ ACK et doit être un bit "récessif". Les messages corrects sont acquittés par les récepteurs, quel que soit le résultat du test d'acceptation.

— **EOF** : La fin du message est indiquée par un champ de fin de trame (EOF) composé de 7 bits récessifs.

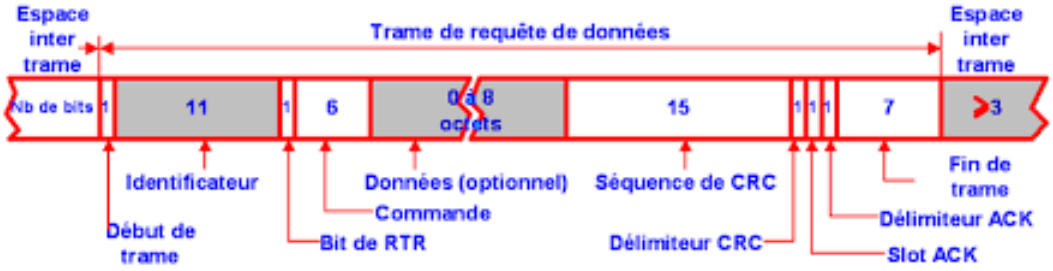


Figure 2.1 – Champs d'une trame en bus CAN

2.2.2 Relation entre le débit et la longueur du bus

Le bus CAN fonctionne à des débits de données allant jusqu'à 1 Mb/sec pour des longueurs de câble inférieures à 40 mètres. Si la longueur du câble augmente, le débit de données tombe généralement à 125 Kb/sec pour une longueur de 500 mètres (1 640 pieds). Le signal de données est normalement transmis sur une paire de fils torsadés (blindés ou non), mais on peut également utiliser un fil unique et une mise à la terre, ou une fibre optique.

Débit(kbits/s)	Distance(m)
1000	30
800	50
500	100
250	250
125	500
20	2500
10	5000

Table 2.1 – Longueur maximale du bus selon le débit

2.2.3 Relation entre le nombre de noeud attaché au bus et le débit

Le nombre de nœuds qui peuvent être attachés à un bus CAN dépend de la résistance de charge minimale qu'un émetteur-récepteur du bus est capable de supporter. Cette résistance de charge est définie par la résistance de terminaison, la résistance de la ligne de bus et la résistance d'entrée différentielle des émetteurs-récepteurs de bus. Par conséquent, les caractéristiques électriques du câble et l'intégrité de l'installation du bus sont importantes. En outre, le long cycle de vie et l'environnement de fonctionnement spécifique des systèmes avioniques doivent être pris en compte. Afin d'assurer une marge de performance adéquate pour couvrir le cycle de vie de la conception, un certain niveau de "déclassement de capacité" est généralement appliqué.

Le tableau 2.2 représente l'expérience de systèmes réussis basée sur l'environnement d'exploitation typique des avions. Les chiffres indiqués supposent que tous les composants du réseau répondent aux critères définis dans la norme ISO 11898-2.

Débit(kbits/s)	Nombre de noeud maximal(m)
1000	30
500	35
250	40
125	50
83.333	60

Table 2.2 – Relation entre le débit du bus et nombre maximal des noeuds connectés

2.2.4 Détection d'erreurs

La détection des erreurs se fait par :

— La surveillance des données transmises sur le bus. Chaque émetteur compare le niveau de bit qu'il a transmis avec le niveau de bit qu'il a reçu du bus.

— Contrôle de redondance cyclique
— Vérification de la trame du message
— Bourrage de bits (bit stuffing) : Les 1 et 0 continus sont évités dans les données transmises pour éliminer les problèmes de synchronisation. Après cinq bits égaux consécutifs, l'émetteur insère un bit de remplissage dans le flux de bits. Ce bit de bourrage a une valeur complémentaire, qui est supprimée par les récepteurs.

2.2.5 CANaerospace

CANaerospace est une définition extrêmement légère du protocole/format de données qui a été conçue pour la communication hautement fiable de systèmes à base de micro-ordinateurs dans des applications aéroportées via CAN (Controller Area Network). L'objectif de cette définition est de créer une norme pour les applications nécessitant un contrôle efficace du flux de données et une synchronisation facile des trames temporelles au sein d'un même système redondant.
La définition est largement ouverte pour permettre la mise en œuvre de types de messages et de protocoles définis par l'utilisateur.
CANaerospace peut être utilisé avec CAN 2.0A et 2.0B (identifiants de 11 et 29 bits) et n'importe quel débit de données du bus.

2.2.6 Type de message

Acronyme	type de message	ID CAN	Priorité
EED	Emergency Event Data	0-127	priorité la plus élevée
NSH	Node Service Data	128-199	↓
UDH	User-Defined DATA	200-299	↓
NOD	Normal Operation Data	300-1799	↓
UDL	User Defined Data	1800-1999	↓
DSD	Debug Service Data	1900-1999	↓
NSL	Node Service Data	2000-2031	priorité la plus faible

Table 2.3 – Type de message CANaerospace

— **EED** : Ce canal de communication est utilisé pour les messages qui nécessitent une action immédiate (c'est-à-dire une dégradation ou une reconfiguration du système) et doivent être transmis avec une très haute priorité. Les données d'événement d'urgence utilisent exclusivement la communication ATM.
— **NSH/NSL** : Ces canaux de communication sont utilisés pour les interactions client/serveur utilisant la communication PTP. Les services correspondants peuvent être de type orienté connexion ou sans connexion. Le NSH/NSL peut également être utilisé pour prendre en charge des fonctions de test et de maintenance.

— **NOD** : Ce canal de communication est utilisé pour la transmission des données qui sont générées pendant le fonctionnement normal du système et décrites dans la liste d'attribution des identifiants de CANaerospace. Ces messages peuvent être transmis périodiquement ou apériodiquement ainsi que de manière synchrone ou asynchrone. Tous les messages qui ne peuvent être attribués à d'autres canaux de communication doivent utiliser ce canal.

— **UDH/UDL** : Ce canal est dédié aux communications qui ne peuvent, en raison de leurs caractéristiques spécifiques, être attribuées à d'autres canaux sans violer la spécification CANaerospace. Tant que la plage d'identifiants définie est utilisée, le contenu du message et le type de communication (ATM, PTP) de ces canaux peuvent être spécifiés par le concepteur du système. Pour assurer l'interopérabilité, il est fortement recommandé de réduire au minimum l'utilisation de ces canaux.

— **DSD** : Ce canal est dédié aux messages qui sont utilisés temporairement à des fins de développement et de test uniquement et ne sont pas transmis pendant le fonctionnement normal. Tant que la plage d'identifiants définie est utilisée, le contenu du message et le type de communication (ATM, PTP) de ces canaux peuvent être spécifiés par le concepteur du système.

2.2.7 Structure du message

Le format général du message utilise un en-tête de message de 4 octets pour : l'identification du noeud, le type de données, le code de message et le code de service (pour les données d'exploitation normale (NOD), le champ du code de service est défini par l'utilisateur). Cela permet l'identification de chaque message par n'importe quelle unité réceptrice sans avoir besoin d'informations supplémentaires.

Les champs de données d'en-tête ont la signification suivante :

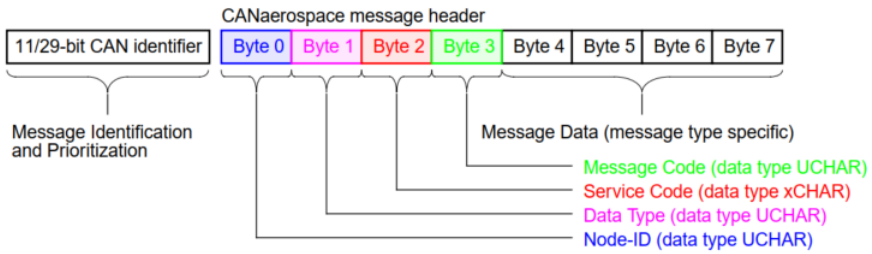


Figure 2.2 – Champs d'une trame en bus CAN Avionique CANaerospace

— **Node ID** : Nœud ID est compris entre 0 et 255, l'ID de nœud "0" étant l'ID de diffusion se référant à "tous les nœuds".

Il est à noter que pour les messages (EED) et (NOD), le node-ID identifie la station émettrice, tandis que pour les messages (NSH/NSL), le node-ID identifie la station adressée.

— **Data type** : Le type de données spécifie le codage des données transportées avec le message correspondant.

— **Message code** : Pour les messages(NOD), le code du message est incrémenté d'une unité pour chaque message. Et peut être utilisé pour contrôler la séquence des messages entrants. Le message code est ensuite remis à zéro après avoir dépassé 255. Cette fonction permet à n'importe quel nœud du réseau de déterminer l'âge d'un signal et la séquence appropriée à des fins de surveillance. Cependant, pour les messages (NSL/NSH), le message code est utilisé pour la spécification étendue du service.

— **Service code** : Pour les messages(NOD), le code de service est composé de 8 bits qui peuvent être modifiés à volonté, il doit être mis à zéro s'il n'est pas utilisé. Pour les messages (NSL/NSH), le code de service contient le code de service du nœud pour l'opération en cours.

2.3 Bande passante

Les systèmes critiques pour la sécurité du vol, doivent être définis avec précision, analysés et testés pour répondre aux exigences formelles de certification. Cela est souvent interprété à tort comme un déterminisme temporel au niveau de la microseconde, mais il s'agit en fait d'une prévisibilité. Le degré de précision requis pour la synchronisation est spécifique à chaque application et doit être quantifié par l'analyse du système.

Cependant, l'objectif ultime à atteindre est de pouvoir démontrer aux autorités de certification qu'un système de sécurité critique basé sur CANaerospace se comporte de manière prévisible en toutes circonstances.

Les nœuds transmettant des messages hautement prioritaires peuvent potentiellement consommer une quantité excessive de bande passante, ce qui bloque souvent d'autres nœuds et cause des retards de transmission imprévisibles. Un tel scénario générerait également une gigue importante dans la transmission des données et doit être totalement évité. Il est donc nécessaire de mettre en place un concept de gestion de la bande passante au niveau du système afin de garantir que la charge du bus dans le système ne soit pas trop élevée.

C'est pour cela que CANaerospace propose un concept de gestion de la bande passante disponible pour les communications de type "one-to-many" et "peer-to-peer", appelé "time triggered bus scheduling".Ce concept est basé sur une limitation du nombre de messages CAN qu'un nœud du réseau peut transmettre dans un "intervalle de temps mineur" afin qu'aucun message ne soit retardé au-delà d'une limite tolérable. La trame temporelle mineure est définie lors de la conception initiale du système. Le nombre maximal de messages transmis dans un intervalle

de temps mineur peut varier d'un nœud à l'autre et présente un potentiel de croissance si la conception du système le permet.

Avec ce concept de planification de bus déclenchée (trame temporelle mineure égale à 12,5 ms dans notre cas) soit 100 paramètres transmis tous les 12,5 ms, ou 8000 paramètres transmis par seconde, générerait une charge de bus de 100%.

Toutefois, il est plus probable qu'une combinaison de paramètres dans les différents groupes de créneaux de transmission soit utilisée. Pour notre système de base, nous avons identifié les données suivantes et leur avons attribuées les créneaux de transmission visualisés dans la colonne Intervalle de transmission (Table des offsets Annex B)[6.2].[10]

2.4 Implémentation

La définition CANaerospace est implémentée sur l'ensemble des noeuds du système, toute trame qui circule dans le bus respecte le format et le planning imposés par la norme ARINC 825.

2.5 Conclusion

Nous avons vu à travers ce chapitre les détails techniques du BUS CAN Avionique **CANaerospace**.
Ce chapitre représente donc la base du travail où toute la partie CAN qui relie les diffèrents noeuds de notre système va être développée.

Chapitre 3

Prototype tableau de bord basé sur microcontrôleur

3.1 Introduction

Le but de ce troisième chapitre est d'expliquer les démarches suivies, exposer les modèles choisis et leurs fonctionnements. Commençant par l'extraction des paramètres du simulateur PREPAR3D, jusqu'à leur arrivée sur le cockpit physique et inversement.

3.2 L'interface "Bridge"

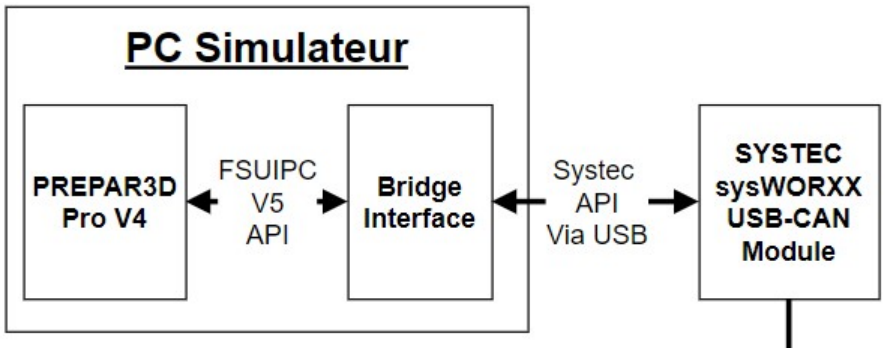


Figure 3.1 – L'organigramme de l'interface Bridge

Cette interface relie le simulateur PREPAR3D PRO V4 via l'API FSUIPC au Bus CAN du Cockpit via l'API Systec SysWorxx. Elle extrait les paramètres de vol et de l'avion, les affiche sur son interface graphique, les envoie sur le Bus CAN, et vice versa. L'affichage de ces paramètres sur la GUI a pour but de :

— La connecter au simulateur et au contrôleur USB-CAN via les boutons "Connect to P3D" et "Connect to USB-CAN".

— Proposer un outil de débuggage en cas de panne. Comparer les paramètres extraits à ceux reçus est très utile afin de déterminer si le problème se génère dans l'extraction ou dans l'insertion de données dans PREPARE3D ou dans l'envoi et la réception de l'information sur le Bus CAN.

— Permettre au pilote instructeur de visualiser tous les paramètres de vol et de l'avion afin d'évaluer l'étudiant en dehors du cockpit, à partir de l'ordinateur simulateur.

Figure 3.2 – L'interface graphique du "Bridge"

3.2.1 Paramètres de vol et d'avion

Afin de comprendre le rôle et le comportement de chaque bouton et instrument déjà listés dans 5.1, nous avons consulté le manuel officiel d'avion Zlin 142 [14]. Il était ensuite nécessaire de trouver l'Offset correspondant pour chaque élément sur la liste des Offsets de FSUIPC [5] et le manipuler afin de connaître son comportement et ses propriétés. Après cela, nous attribuons un ID CAN et un intervalle de transmission spécifiés dans le protocole CAN Avionique [10] à chaque paramètre qui sera envoyé sur le Bus CAN en observant les nœuds qui envoient et ceux qui reçoivent ce paramètre. Enfin, nous choisissons l'instrument électronique utilisé pour reproduire l'instrument avionique.

L'application de ce processus nous a donné la table suivante (le tableau complet se trouve dans l'annexe de ce mémoire) :

Liste des offsets						
Nom de l'instrument	Offset Address	Taille (byte)	Nom de la Variable	Description et comportement	CAN ID	Intervale de transmission
Air Speed	0x02BC	4(uint)	airSpeedOffset	*0.0001313 to convert to gauge value	315	12.5 ms
Gyro Horizon	0x2F70	8(double)	attitudePitchOffset		311	12.5 ms
	0x2F78	8(double)	attitudeBankOffset		312	12.5 ms
Rate Of Climb	0x02C8	4(int)	verticalSpeedOffset	*0.004 to convert to gauge value	1124	12.5 ms
....	....	....	....	....	....	....

Table 3.1 – Liste des offsets

3.2.2 Flux de données

Comprendre le flux de chaque donnée est nécessaire afin de le reproduire, pour comprendre quels sont les nœuds qui génèrent telle donnée, et quels sont les nœuds ciblés par cette donnée, on les catégorise comme suit :

3.2.2.1 Données locales dans le nœud

On prend un exemple simple pour cette catégorie : le bouton qui allume la radiocommunication. Cette dernière est présentée par un microcontrôleur Mbed relié à des afficheur 7 segments et des encodeurs rotatifs pour afficher et varier la fréquence. Le bouton qui allume et éteint cette radio est relié au même microcontrôleur et sert à éteindre et allumer les afficheurs et donc son état n'est pas envoyé sur le Bus CAN, c'est une donnée locale.

Un autre exemple que l'on peut trouver est une donnée synthétique, si l'utilisateur introduit deux données A et B au nœud qui calcule à partir de ces deux données une troisième donnée et l'envoie par le Bus CAN, on pourra dire que A et B sont des données locales.

3.2.2.2 Données locales dans le cockpit

Cette catégorie englobe les données qui sont communiquées entre les nœuds via le Bus CAN mais ne ciblent pas le simulateur (PC). Un exemple concret est le bouton "Instrument" du "Switch Panel" qui met en marche les instruments électriques du cockpit (radiocommunication, radionaviguation, indicateurs électriques, GPS). L'état de ce switch est envoyé par le microcontrôleur Mbed du Switch Panel vers le nœud de radiocommunication, le nœud des 9 instruments de bord et le nœud du GARMIN G5 mais pas à l'ordinateur simulateur.

3.2.2.3 Données communiquées avec le simulateur PREAP3D

3.2.2.3.1 Données unidirectionnelles

Simulateur vers cockpit Ces données sont envoyées en boucle par l'interface Bridge (un timer pour extraire et envoyer les données des 9 instruments de bord et un autre timer pour envoyer les données du GARMIN G5).

Cockpit vers simulateur Ces données sont envoyées lors d'une interruption (pas en boucle) liée à un changement d'état, exemple : lors d'un changement d'état d'un switch, on envoie un message sur le Bus CAN vers le simulateur. Tous les switchs et les potentiomètres génèrent des données unidirectionnelles du cockpit vers le simulateur envoyées lors de changement d'état.

3.2.2.3.2 Données bidirectionnelles Ce sont les données modifiables par plus d'un nœud, et donc chaque nœud doit mettre à jour sa donnée régulièrement avant de la modifier, par conséquent une modification de la donnée sur un nœud entraîne sa modification sur tous les autres nœuds ciblés.

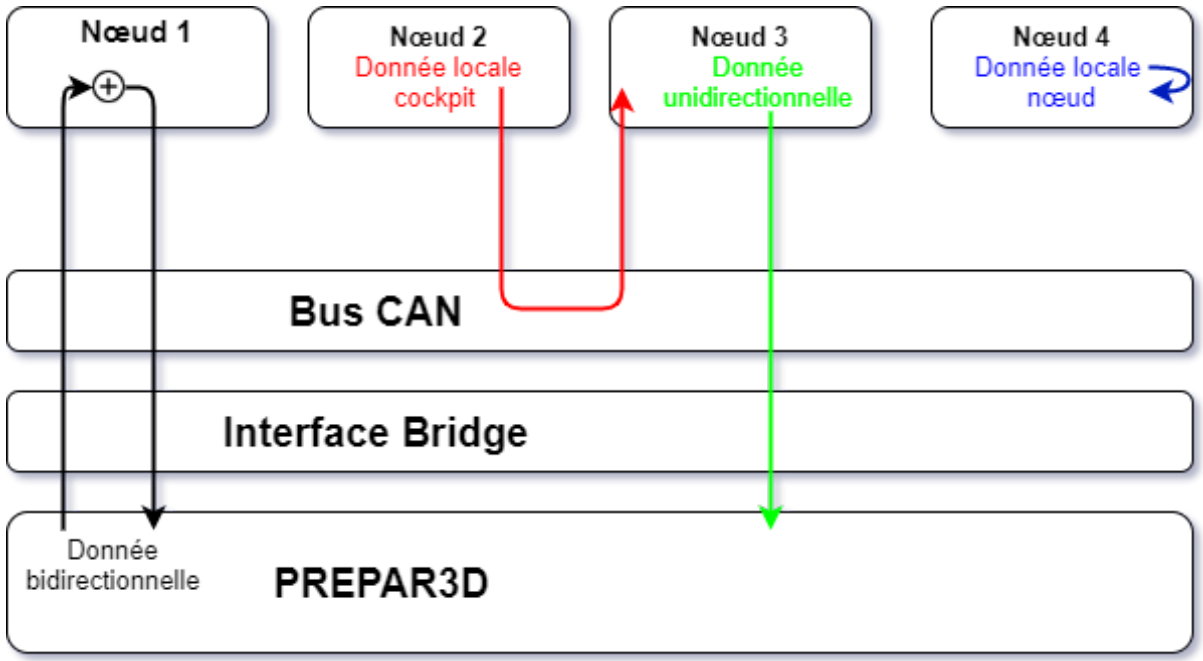


Figure 3.3 – Le flux des données

3.2.3 Les threads de l'interface Bridge

Le Bridge comporte 3 timers :

— Timer 1 : Extrait et envoie les données qui ont un intervalle de transmission de 12.5ms.

— Timer 2 : Extrait et envoie les données qui ont un intervalle de transmission de 100 ms.

— Thread 3 : Se déclenche à chaque interruption de réception de données UART, il insère la donnée reçue dans l'offset correspondante en faisant un "switch case" avec l'ID CAN reçu.

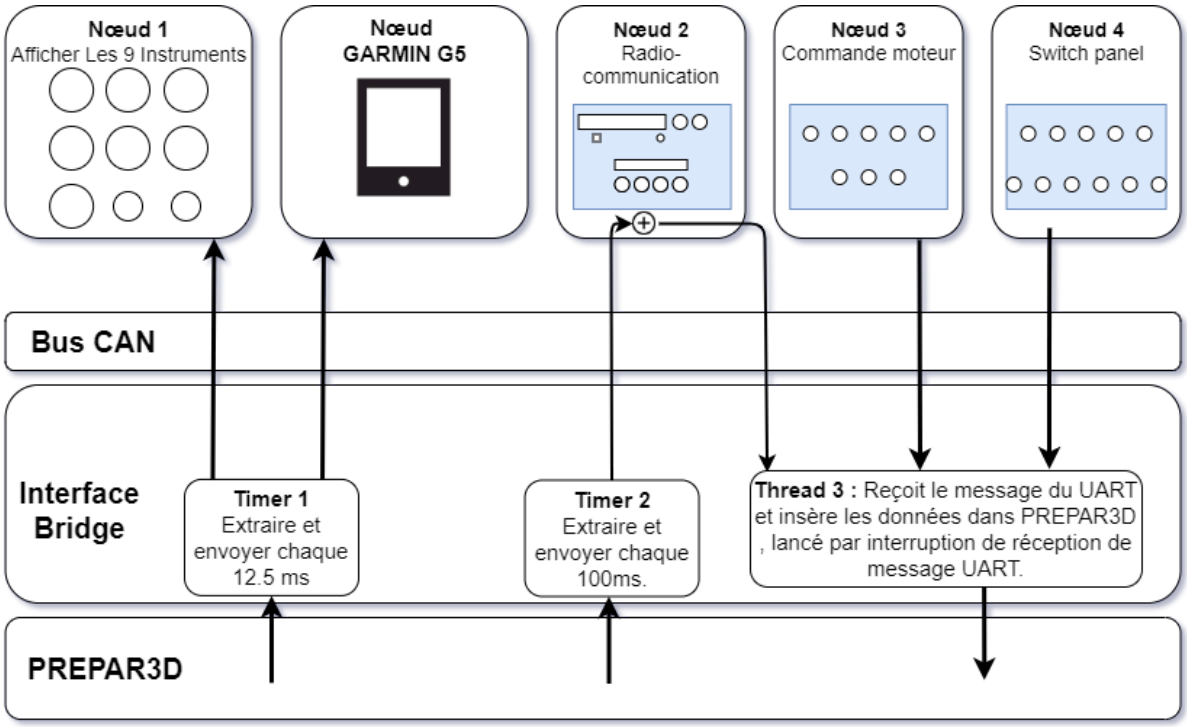


Figure 3.4 – Les threads de l'interface Bridge

3.3 Composition des nœuds

3.3.1 Nœud des 9 instruments de bord

Ce noeud comporte l'affichage des instruments de vol, de navigation et de contrôle moteur présentés dans le chapitre 1. Ces instruments sont développés grâce au framework QT. L'interface est implémentée sur une Raspberry Pi 3 qui est reliée au bus CAN grâce au CAN controller mcp2515 et CAN transciver mcp2551, le schéma suivant présente le noeud :

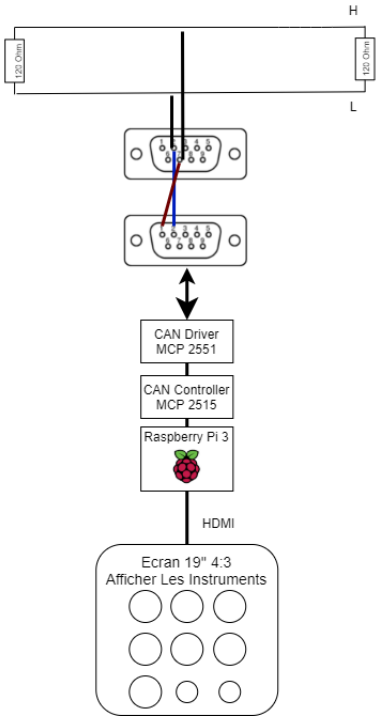


Figure 3.5 – Noeud 9 instruments

3.3.2 Nœud GARMIN G5

L'explication de ce nœud est la partie la plus importante de ce chapitre, il représente 50% de notre projet de recherche.

3.3.3 Nœud de la radiocommunication

Il comporte la radio avec ses deux fréquences "Radio Com 1" et "Radio Com 1 Standby" et le transpondeur avec une seule fréquence. Ses données sont bidirectionnelles. Son microcontrôleur est programmé avec des "Queue Events" de l'API RTOS de Mbed OS, une classe qui tolère les interruptions et les threads. Les schémas de son circuit (board) sont joints à l'annexe.

3.3.4 Nœud de la commande moteur

Ce nœud comporte les boutons de commande moteur de base (Magneto Switch, Main Switch, Throttle, Starter et Mixture). Ses données sont unidirectionnelles du cockpit au simulateur. Son microcontrôleur est programmé en boucle à détecter les changements d'état d'un bouton et l'envoyer en un message CAN. Les schémas de son circuit (board) sont joints à l'annexe.

3.3.5 Nœud du Switch Panel

Ce nœud englobe le Switch Panel. Ses données sont unidirectionnelles du cockpit au simulateur. Son microcontrôleur est programmé en boucle à détecter les changements d'état d'un bouton et l'envoyer en un message CAN. Des sorties sont prêtes pour relier un joystick et un palonnier. Les schémas de son circuit (board) sont joints à l'annexe.

3.3.6 Partie physique

Sur une planche de forex, découpée avec une machine à commande numérique en suivant les dessins techniques [15], on place un autocollant, les circuits et composants nécessaires.

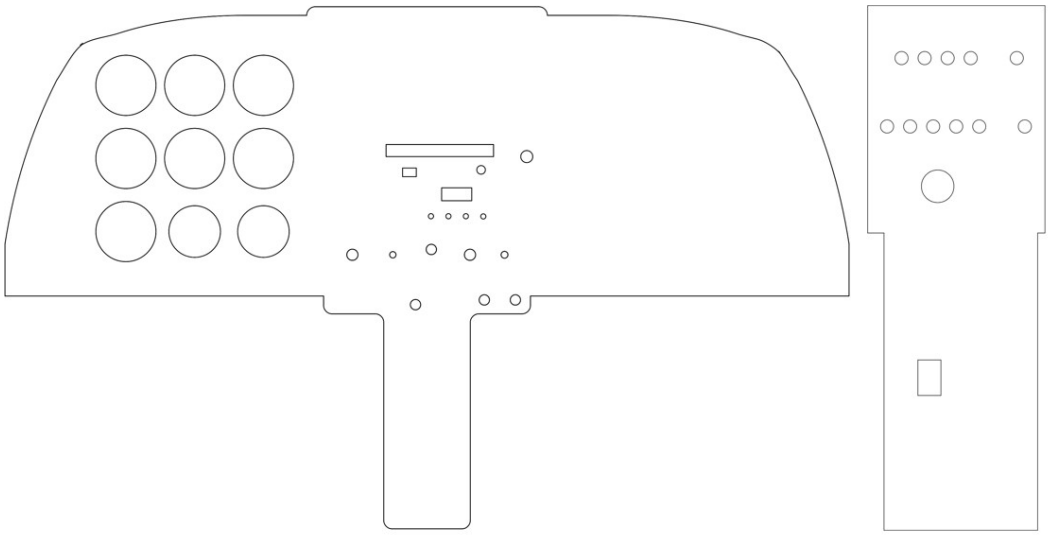


Figure 3.6 – Les chemins de la découpe

Figure 3.7 – Les autocollants décoratifs

3.4 Câblage du Bus CAN et alimentation

Le câble du Bus CAN doit être adapté à la communication numérique. De plus, une paire torsadée est nécessaire pour annuler quasiment les tensions induites par le champ magnétique que produit le courant en passant dans ces torsades. À chaque torsade, alternativement, le signe de la tension induite change, ce qui fait que globalement, lorsque l'on considère une longueur importante de torsades, la tension "d'auto-induction" est énormément réduite, aussi, un câble avec plusieurs paires de fils, doit être torsadé, chaque paire étant elle-même torsadée, pour éviter les couplages.

Le câble d'alimentation doit porter une tension de 9V (permet d'éviter une chute de tension liée à la longueur du câble et réduire aussi le courant porté en gardant la puissance constante), qui sera réduite à 5V par des régulateurs industriels K7805-2000R3 [16] à l'entrée de chaque circuit.

Le câble RJ45 répond à ces critères, il présente 4 paires torsadées, une paire est utilisée pour le Bus CAN (verte) terminée par une résistance d'adaptation de 120 Ohm de chaque côté. A partir de la catégorie 5 du câble RJ45, les paires bleue et rouge sont utilisées pour faire passer respectivement "DC+" et "DC-", cette fonctionnalité est nommée PoE (Power over Ethernet) [17], cette configuration peut porter jusqu'à 57V/960mA [17] ce qui satisfait nos besoin.

Le schéma général de l'alimentation devient :

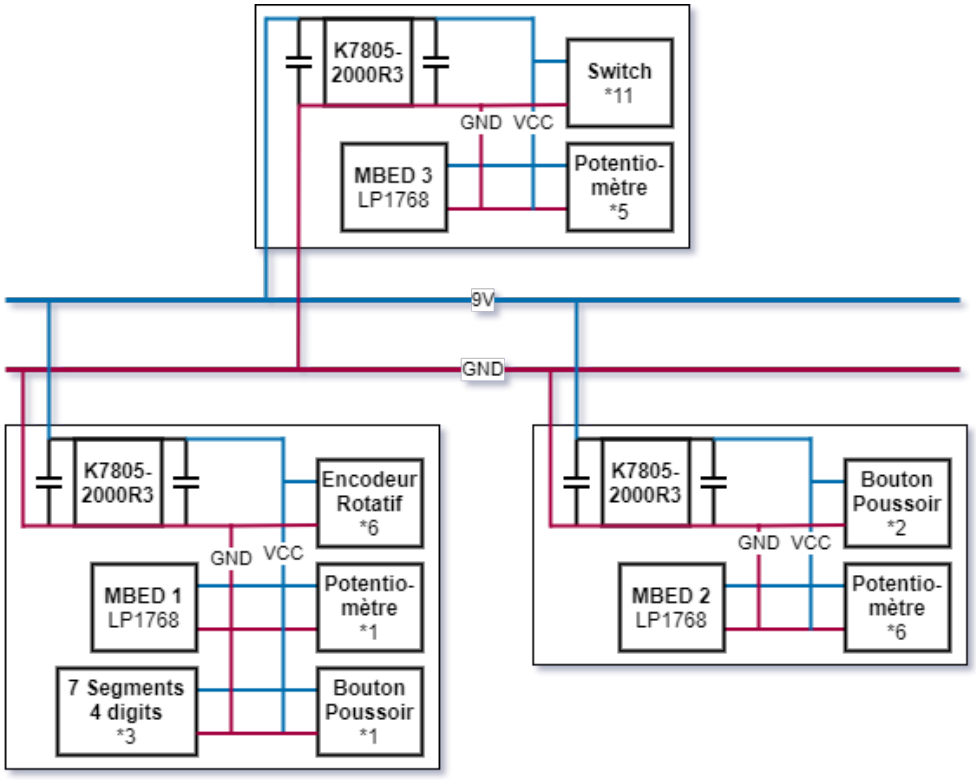


Figure 3.8 – Schéma générale de l'alimentation.

Pour les ports du Bus CAN, nous utilisons les connecteurs DB9 avec la configuration des pins conçue par "CiA DS102" :

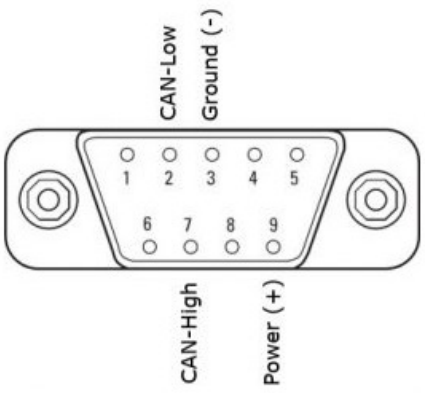


Figure 3.9 – Configuration D-SUB9 CiA DS102

D-SUB Pin	Signale	File
1	-	-
2	CAN Low	Vert et blanc
3	GND	Rouge et blanc
4	-	-
5	-	-
6	-	-
7	CAN High	Vert
8	-	-
9	9 Voltes	Bleu

Figure 3.10 – CiA DS102.

Or, le vrai GARMIN G5 utilise la configuration J51 qu'on trouve dans son manuel [3] :

D-SUB Pin	Signale	File
1	CAN High	Vert
2	CAN Low	Vert et blanc
3	UNIT ID	Rouge et blanc
4	RS-232 RX 1	-
5	RS-232 TX 1	-
6	SIGNAL GROUND	-
7	AIRCRAFT POWER 1	Bleu
8	AIRCRAFT POWER 2	-
9	POWER GROUND	

Table 3.2 – J51.

3.5 USB-CAN

Cet Mbed reçoit une ligne de caractères de la forme suivante :

idCAN,prefixCAN,dataCAN
1101,7402,2879

Cette ligne est séparée par virgule, l'idCAN est converti en uint16_t, le préfixe en uint32_t où chacun des quatre bytes a sa signification, le dataCAN en uint32_t, int32_t ou en un float de 4 bytes selon le byte du préfixe qui définit le type de data reçu. Puis, les bytes du prefixCAN sont concaténés avec ceux du dataCAN en un tableau de 8 bytes. Ce dernier est envoyé par Bus CAN avec l'ID correspondant.

De même, quand un message CAN est reçu, son ID est conservé dans une chaîne de caractères, les 4 premiers bytes reçus sont le préfixe du message, les 4 derniers bytes représentent la donnée, son type est définit par le deuxième byte du préfixe. Les trois parties sont concaténés sous forme de chaîne de caractères et envoyés en une seule ligne de caractères via UART.

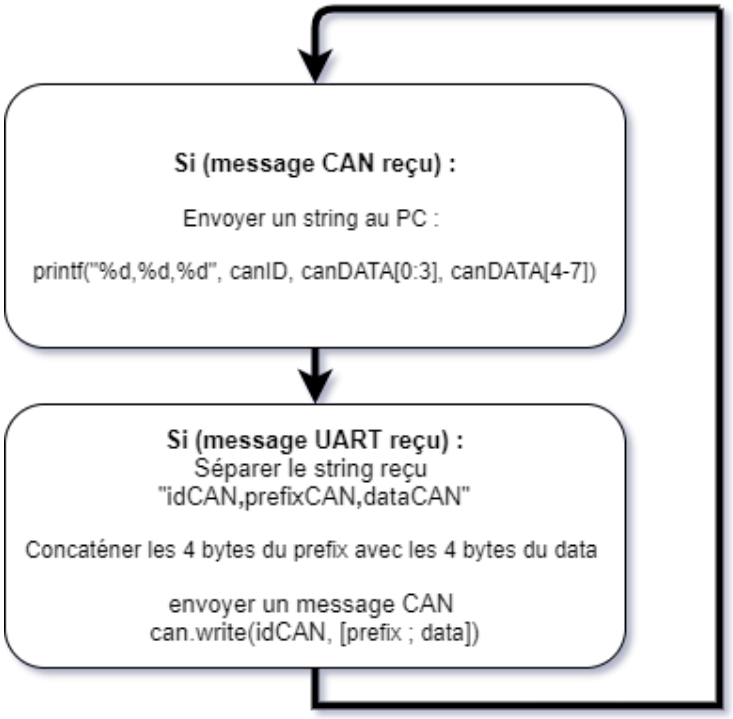


Figure 3.11 – Fonctionnement de l'Mbed USB-CAN

La même solution a été implémentée mais en envoyant les données par bytes via UART pour avoir un débit plus élevé, car l'envoi et la réception des chaînes de caractères prennent un temps plus important. Cette solution est fonctionnelle mais doit être améliorée afin d'éviter la perte des paquets de données.

3.6 Conclusion

Ce chapitre a touché à plusieurs notions techniques, de l'avionique jusqu'aux systèmes embarqués en passant par un peu d'informatique. Le travail réalisé est le fruit d'une large documentation et recherche qui nous a éclairci sur ces notions et nous a aidé à les relier.

Chapitre 4

Prototype du Garmin G5

4.1 Introduction

Ce chapitre est consacré à la deuxième partie de notre travail qui concerne le Prototypage du Garmin G5 à base de Raspberry Pi.

Nous y aborderons d'abord le développement de l'interface graphique du Garmin G5 avec le framework QT.

Par la suite, nous passerons au circuit électronique en détaillant le rôle de chaque composant et en explicitant les procédés de communications entre les périphériques.

4.2 Architecture du système

4.2.1 Schéma général

L'architecture du Garmin G5 est basé sur la carte Raspberry Pi 4 qui est relié à l'écran Adafruit 3.5", le circuit électronique utilise le bus CAN comme bus de données principal.

Ce noeud se décompose en deux parties. D'abord, la partie connexion au bus CAN où nous utiliserons un module CAN pour relier notre Garmin au bus de données du simulateur. Ensuite, la deuxième partie représente l'interface graphique qui consiste à reproduire le GUI du Garmin G5 pour afficher les multiples données illustrées dans les deux figures 4.5 et 4.6.

La figure 4.1 montre le schéma détaillé du noeud Garmin G5, les différentes connexions, ainsi que les protocoles ou les circuits permettant de relier ses composants.

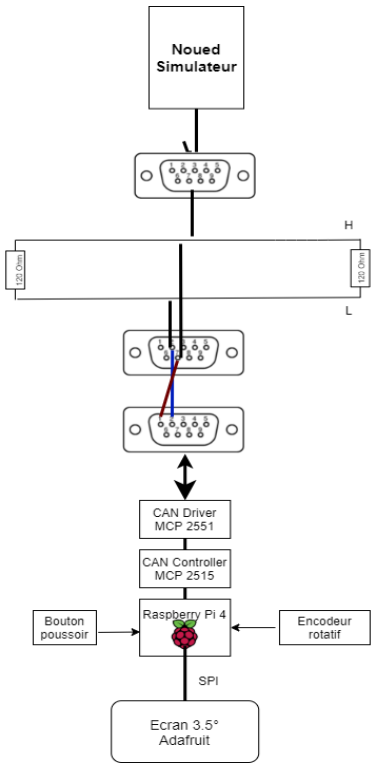


Figure 4.1 – architecture du noeud

4.2.2 Explication des sous-noeuds

4.2.2.1 Sous-noeud contrôle

Dans ce sous-noeud la Raspberry Pi est connectée au module CAN qui ce constitue du mcp2515+mcp2551.

Cette partie récupère les données du bus et les envoie vers l'interface graphique pour les afficher.

Aussi ce sous-noeud comprend un encodeur rotatif à deux canaux qui permet de fixer la valeur de l'altitude souhaitée ainsi que la valeur du cap à atteindre.

4.2.2.2 Sous-noeud GUI

Ce sous-noeud embarque une interface graphique qui a été développée avec QT framework. Pour l'affichage un écran adafruit 3.5" est utilisé, il se sert du bus SPI pour se connecter à la Raspberry pi.

4.2.3 Fonctionnement du système

Nous allons maintenant détailler le fonctionnement de chaque sous-nœud et son interaction avec les autres.

4.2.3.1 Sous-noeud Contrôle

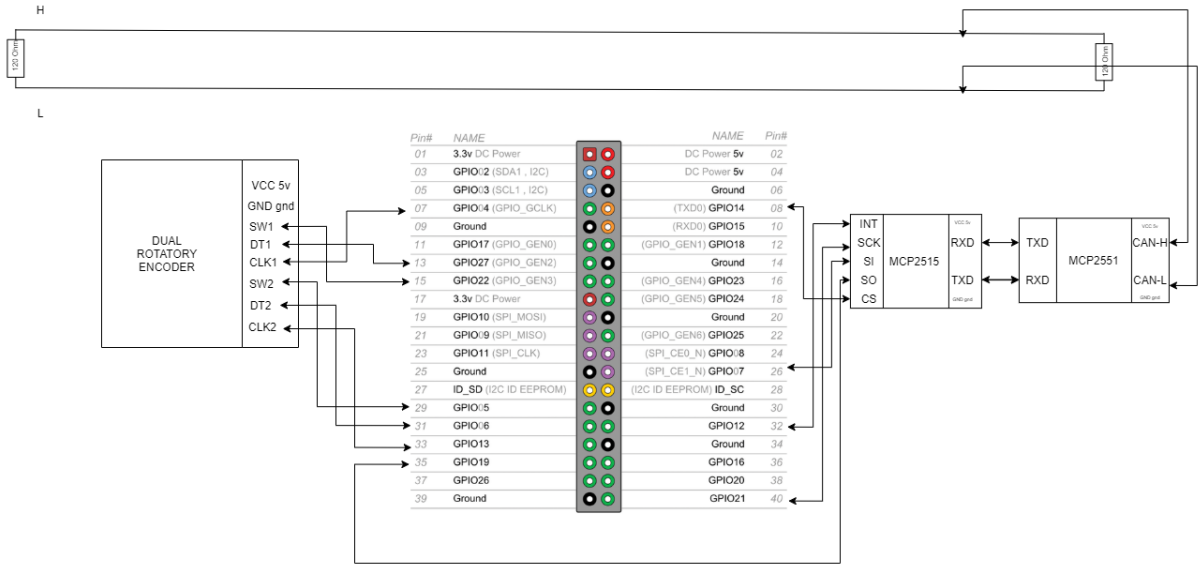


Figure 4.2 – Architecture du sous-noeud contrôle

Ce sous-noeud se charge de récupérer les données du bus CAN et les transmettre au GUI, d'abord les données sont récupérées du simulateur grâce à l'API FSUIPC puis converties en trame qui passe dans le bus CAN grâce au module systech, puis le noeud Garmin G5 applique un filtre pour récupérer uniquement les données qu'il affiche et ignore les autre trames, le filtrage se fait à base de ID.

Comme expliqué précédemment le module mcp2551 représente l'interface avec le bus CAN physique et le module mcp2515 représente la partie logique qui gère la réception et la transmission des données de la Raspberry au bus CAN.

la figure 4.3 représente les différentes étapes permettant de récupérer les données du bus CAN de la part du noeud Garmin.

Figure 4.3 – Organigramme qui décrit les étapes qui induit à la lecture des trame CAN

4.2.3.1.1 Encodeur rotatif l'encodeur rotatif utilise deux canaux, il nous donne la possibilité de modifier deux paramètres de données sur le Garmin G5(Altitude et cap).

4.2.3.2 Sous-noeud GUI

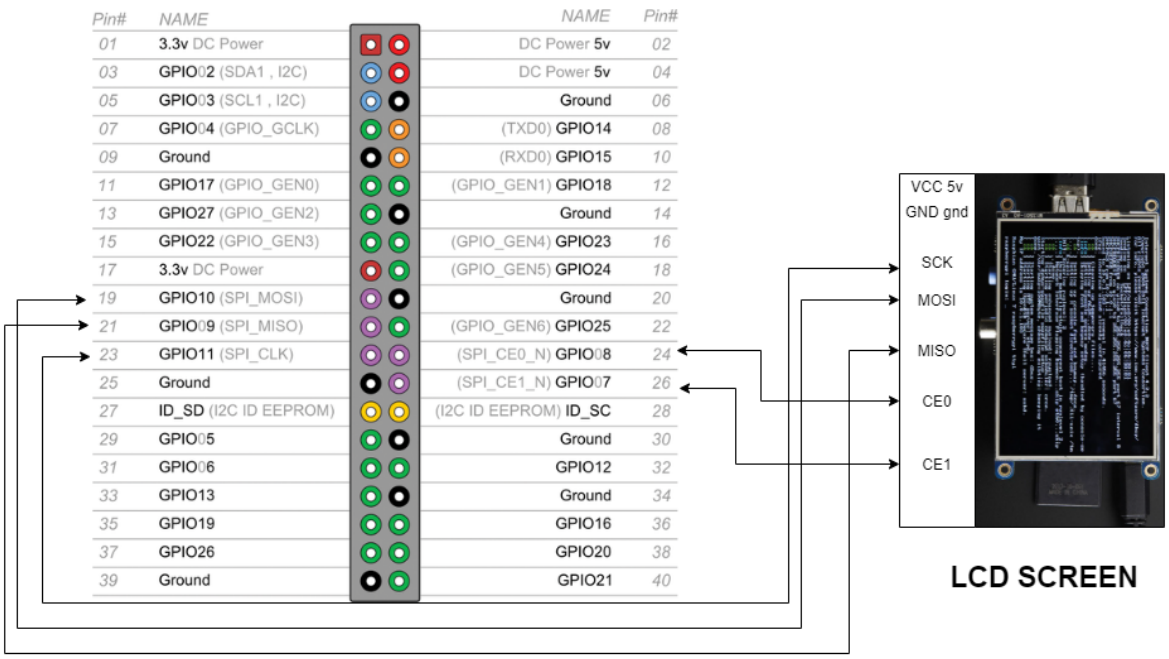


Figure 4.4 – Architecture du sous-noeud GUI

Le sous-noeud GUI s'occupe d'afficher les données récupérées du bus CAN.

Pour l'interface graphique nous avons utilisé le framework QT qui est une bibliothèque très puissante et complète utilisée pour créer des interfaces de haute résolution et qualité (Mercedes l'utilise comme outil de développement d'interface graphique sur ses modèles de luxe S class), le développement est fait en c++, l'organigramme suivant montre les étapes de développement.[18] [19][20]

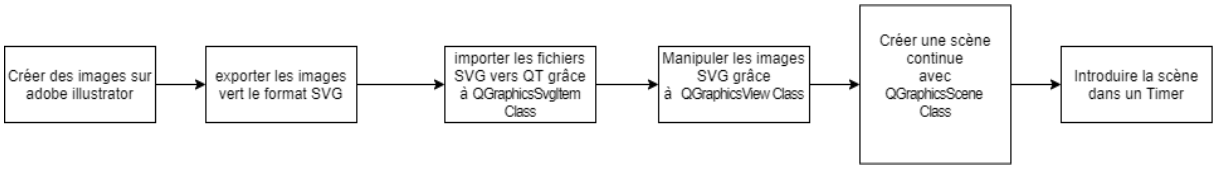


Figure 4.5 – organigramme de développement

Les figures 4.5 et 4.6 suivantes montrent la partie PFD et HSI de l'interface :

Figure 4.6 – HSI

Figure 4.7 – PFD

4.2.4 Rétrospective des sous-noeuds

La Figure 4.7 résume l'activité des trois nœuds combinés en fonction du temps. Le processus décrit dans cette figure tourne en boucle tant que le système est en marche.

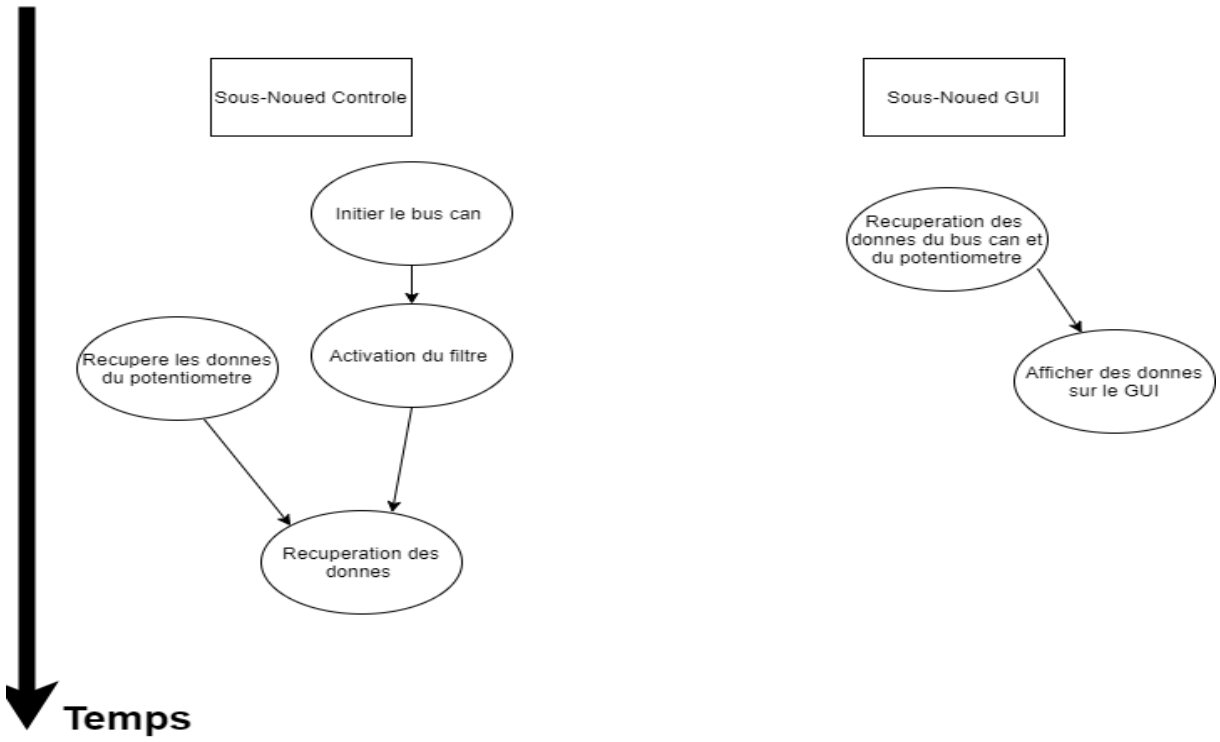


Figure 4.8 – Architecture du sous-noeud

4.2.5 Schéma d'alimentation

Le schéma de la Figure 4.8 montre les différentes connexions électriques qui permettent d'alimenter tout le circuit.

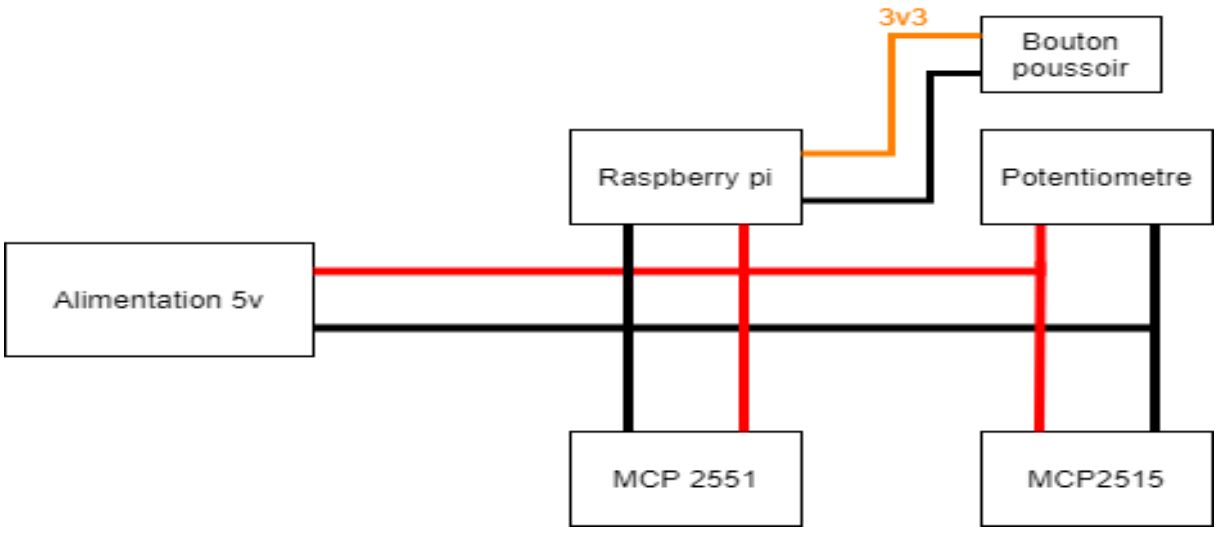
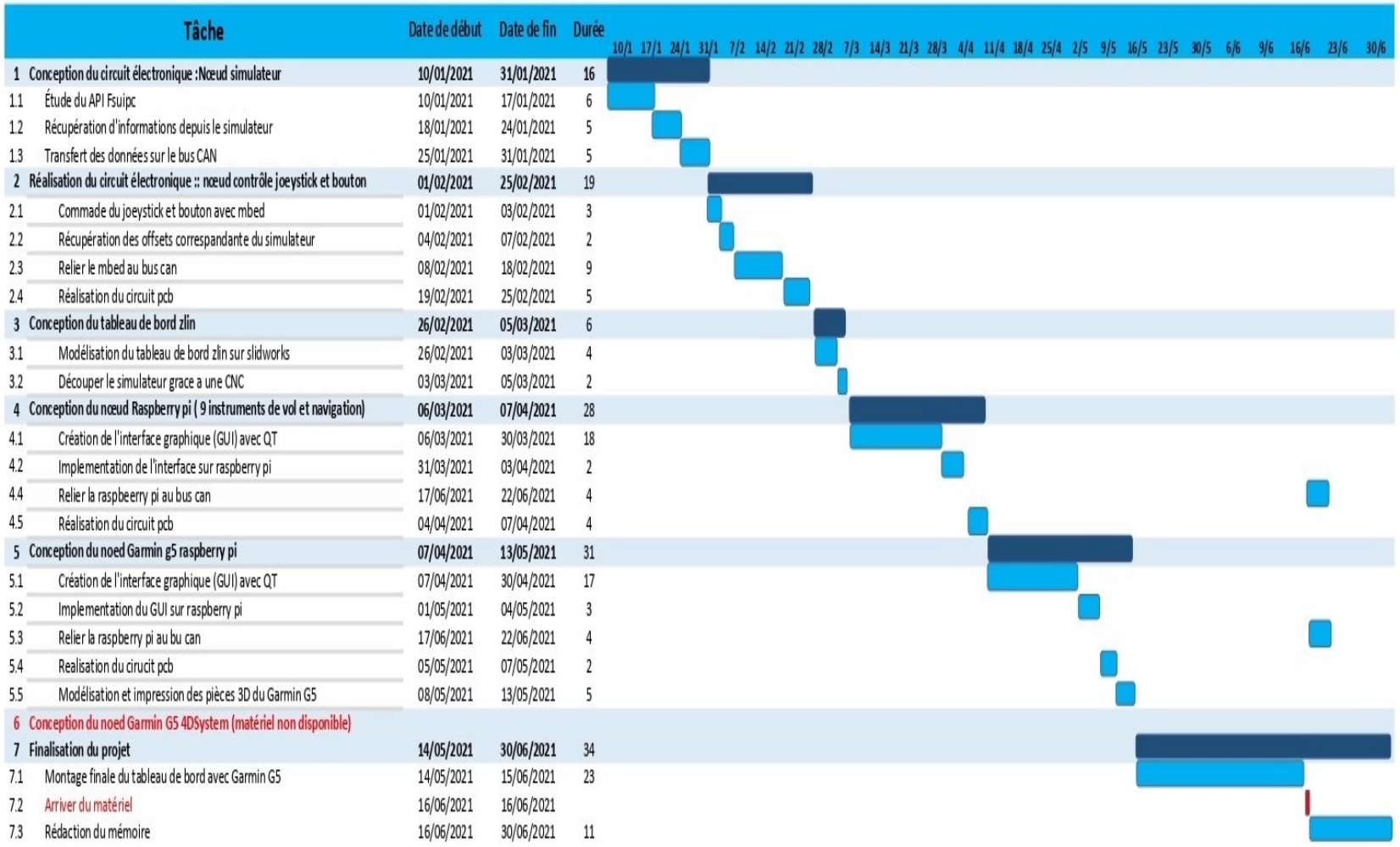


Figure 4.9 – Schéma d'alimentation du système

4.3 Conclusion

Ce chapitre détaille la conception et l'étude du prototype Garmin G5.
Le système est donc divisé en deux nœuds distincts communiquant à travers le bus CAN. Chaque nœud possède un rôle bien défini qu'on peut résumer par : la récupération de données du Bus CAN, le traitement de ces données et enfin l'exploitation des données en les affichant sur le GUI.



	Tâche	Date de début	Date de fin	Durée
1	**Conception du circuit électronique :Nœud simulateur**	**10/01/2021**	**31/01/2021**	**16**
1.1	Étude du API Fsuipc	10/01/2021	17/01/2021	6
1.2	Récupération d'informations depuis le simulateur	18/01/2021	24/01/2021	5
1.3	Transfert des données sur le bus CAN	25/01/2021	31/01/2021	5
2	**Réalisation du circuit électronique :: nœud contrôle joeystick et bouton**	**01/02/2021**	**25/02/2021**	**19**
2.1	Commade du joeystick et bouton avec mbed	01/02/2021	03/02/2021	3
2.2	Récupération des offsets correspandante du simulateur	04/02/2021	07/02/2021	2
2.3	Relier le mbed au bus can	08/02/2021	18/02/2021	9
2.4	Réalisation du circuit pcb	19/02/2021	25/02/2021	5
3	**Conception du tableau de bord zlin**	**26/02/2021**	**05/03/2021**	**6**
3.1	Modélisation du tableau de bord zlin sur slidworks	26/02/2021	03/03/2021	4
3.2	Découper le simulateur grace a une CNC	03/03/2021	05/03/2021	2
4	**Conception du nœud Raspberry pi (9 instruments de vol et navigation)**	**06/03/2021**	**07/04/2021**	**28**
4.1	Création de l'interface graphique (GUI) avec QT	06/03/2021	30/03/2021	18
4.2	Implementation de l'interface sur raspberry pi	31/03/2021	03/04/2021	2
4.4	Relier la raspbeerry pi au bus can	17/06/2021	22/06/2021	4
4.5	Réalisation du circuit pcb	04/04/2021	07/04/2021	4
5	**Conception du noed Garmin g5 raspberry pi**	**07/04/2021**	**13/05/2021**	**31**
5.1	Création de l'interface graphique (GUI) avec QT	07/04/2021	30/04/2021	17
5.2	Implementation du GUI sur raspberry pi	01/05/2021	04/05/2021	3
5.3	Relier la raspberry pi au bu can	17/06/2021	22/06/2021	4
5.4	Realisation du cirucit pcb	05/05/2021	07/05/2021	2
5.5	Modélisation et impression des pièces 3D du Garmin G5	08/05/2021	13/05/2021	5
6	**Conception du noed Garmin G5 4DSystem (matériel non disponible)**			
7	**Finalisation du projet**	**14/05/2021**	**30/06/2021**	**34**
7.1	Montage finale du tableau de bord avec Garmin G5	14/05/2021	15/06/2021	23
7.2	Arriver du matériel	16/06/2021	16/06/2021	
7.3	Rédaction du mémoire	16/06/2021	30/06/2021	11

Figure 4.10 – Diagramme de Gantt effectif

Conclusion

Notre projet a consisté en l'étude et la conception d'un simulateur de vol pour un Zlin 142 en y connectant des calculateurs réels sous forme de nœuds utilisant une architecture LRU avec comme système d'interconnexion un bus CAN. Son objectif visait à offrir non seulement une méthode de conception de simulateur pour l'avionique mais aussi la réalisation d'un simulateur à base de COTS. Elle peut constituer une solution bas coût dédiée aussi bien à la formation de pilote que d'ingénieur système.

Comme développé dans notre présent mémoire, notre étude et notre conception a traité trois volets pouvant apparaître distincts mais néanmoins complémentaires pour notre objectif global ; à savoir offrir une solution réunissant les spécifications d'un simulateur de vol d'actualité. Le premier volet nous a conduit à la mise en œuvre d'une architecture LRU classique adossée à un bus CAN et pilotée par un simulateur de vol porté par l'application P3D tournant sur un PC. L'ensemble des modules , chacun constituant un nœud, a été construit autour d'un ARM Cortex M3. Les nœuds Switch Panel, Instruments Virtuels, radiocommunication, contrôle Moteur ont tous été testés fonctionnels. L'approche modulaire a été un atout pour réduire les couplages fonctionnels. Le développement s'est fait exclusivement en langage C++ sous environnement Keil. La technologie évoluant très rapidement, le cœur des briques ainsi développé peuvent évoluer vers des solutions moins coûteuses et plus performantes à moindre coût avec une portabilité aisée des codes que nous avions développés. Couplé au langage C++, la bibliothèque Qt a été exploitée pour permettre le développement aisé de widgets adaptés pour le module instrument virtuel. L'environnement de développement utilisé a été porté naturellement par Linux sachant que notre cible matérielle est une Raspberry Pi 3.

Le deuxième volet a traité l'étude et la conception d'une surcouche protocolaire du bus CAN pour la rendre compatible avec la norme ARINC825 (CAN aerospace) . Notre but est de standardiser le format d'échange des trames, de faciliter la gestion et la planification de l'envoi des données ; le premier résultat étant la minimisation des risques de saturation du bus CAN. Le second résultat vise à fournir un outil destiné à la formation pour l'apprentissage et l'évaluation de ce protocole dédié. L'implémentation s'est faite en C++ après la définition de l'ensemble des classes nécessaires. Cette implémentation a été portée et validée fonctionnellement sur le PC faisant tourner le simulateur de vol. La même implémentation a été portée sur les diverses LRU

développées. L'évaluation de performance n'a pas été faite liée à des contraintes de temps.

Le troisième volet a abouti à une solution alternative bas coût à une Garmin G5. Ce nœud, implémenté sur une raspberry pi 4, voit son interface graphique développée en c++ et à l'aide de la bibliothèque Qt. L'environnement de développement a été commun au module dédié à l'instrumentation virtuelle. Le résultat de développement confirme son usage possible tant en instrument secondaire que primaire avec des rendus réalistes.

En termes de perspectives, comme mentionné précédemment, l'évolution vers des circuits plus récents, moins chers et plus performants peut être une opportunité. Au stade de notre développement, les améliorations du système ainsi développé nous semblent nécessaires. On cite :

— Emulation de la Garmin G5 avec un écran intelligent 4Dsystem qui offre un affichage plus performant mais une liaison indirecte au Bus CAN (passe par un microcontrôleur supplémentaire).

— Établissement d'une communication du simulateur vers le noeud de la radiocommunication qui n'était pas faite en raison de l'indisponibilité du convertisseur USB-CAN.

— Ajout des nœuds pour la radionavigation (ADF, NAV1, NAV2).

— Test des limites de la stabilité de chaque modèle de conversion USB-CAN conçu : module USB sysWORXX vers microcontrôleur (conversion des lignes de caractères en des messages CAN) et microcontrôleur (conversion des octets en messages CAN).

— Classement par fiabilité, débit et coût afin d'éclaircir les choix futurs.

— Détermination de la limite de saturation du Bus CAN Avionic et le respect de cette limite lors de l'ajout des nœuds/données dans le futur[21].

Après ces modifications dans le PoC, le projet d'amélioration du Zlin 142 pourra passer aux étapes suivantes, réalisation d'un prototype puis la réalisation d'un MVP.

Figure 4.11 – Interface Cockpit rélaisée

Bibliographie

[1] M. S. D. Rabah LOUALI Abdelmalek BELLOULA et S. BOUAZIZ., « Real-time characterization of Microsoft Flight Simulator 2004 for integration into Hardware In the Loop architecture, » *19th Mediterranean Conference on Control and Automation*, p. 1241-1246, 2011. doi : https://www.researchgate.net/profile/Rabah-Louali/publication/238009764_Real-time_characterization_of_Microsoft_Flight_Simulator_2004_for_integration_into_Hardware_In_the_Loop_architecture/links/5516a01e0cf2d70ee27525c2/Real-time-characterization-of-Microsoft-Flight-Simulator-2004-for-integration-into-Hardware-In-the-Loop-architecture.pdf.

[2] A. E. S. B. Rabah LOUALI(1) Hind GACEM(2). (). « Implementation of an UAV Guidance, Navigation and Control System based on the CAN data bus : Validationusing a Hardware In the Loop Simulation, » adresse : https://ieeexplore.ieee.org/abstract/document/8014217.

[3] GARMIN. (). « Garmin G5 Electronic Flight Instrument Part 23 AML STC Installation Manual, » adresse : http://static.garmin.com/pumac/190-01112-10_09.pdf.

[4] L. Martin. (). « Simulateur de vol P3D, » adresse : https://www.prepar3d.com/.

[5] Pete et J. Dowson. (). « Pete John Dowson's Software, » adresse : http://www.fsuipc.com/.

[6] Mbed. (). « Mbed LPC1768, » adresse : https://os.mbed.com/platforms/mbed-LPC1768/.

[7] F. R. Pi. (). « Raspberry PI, » adresse : https://www.raspberrypi.org/.

[8] D. systeme. (). « ULCD-35DT, » adresse : https://4dsystems.com.au/products/4d-intelligent-hmi-display-modules/microlcd-display-modules/ulcd-35dt, https://www.gotronic.fr/art-ecran-tactile-3-5-gen4-ulcd-35dct-clb-27274.htm.

[9] adafruit. (). « PiTFT - Assembled 480x320 3.5. »

[10] M. S. F. SYSTEMS. (). « CAN Aeorspace, » adresse : http://www.stockflightsystems.com/tl_files/downloads/canaerospace/canas_17.p.

[11] J. Klüser, *CAN-based Protocols in Avionics*. 2012.

[12] T. Vaira, *Cours Qt Module IHM*.

[13] M. Org. (). « An introduction to Arm Mbed OS 5, » adresse : https://os.mbed.com/docs/mbed-os/v5.15/introduction/index.html.

[14] C. R. M. Z. A. a.s. (). « FLIGHT MANUAL OF THE Z 142 AIRCRAFT, » adresse : https://www.manualslib.com/products/Zlin-Aircraft-Z-142-10920191.html.

[15] W. TALABOULMA. (). « Conception d'un simulateur de vol matériel ZLIN142, » adresse : http://mt.biblio.intranet.enp.edu.dz/pub/Pfe/Electronique/Annee_2012/TALABOULMA.Walid/TALABOULMA.Walid.pdf..

[16] RS. (). « Datasheet RS Pro K78xx-2000R3 DC-DC Converter, » adresse : https://docs.rs-online.com/fc3c/A700000006631878.pdf.

[17] Satoms. (). « Power over Ethernet (PoE), » adresse : https://satoms.com/power-over-ethernet-poe-explained-and-specifications/.

[18] QT. (). « QGraphicsSvgItem Class, » adresse : https://doc.qt.io/qt-5/qgraphicssvgitem.html.

[19] ——, (). « QGraphicsScene Class, » adresse : https://doc.qt.io/qt-5/qgraphicsscene.html.

[20] ——, (). « QGraphicsView Class, » adresse : https://doc.qt.io/qt-5/qgraphicsview.html.

[21] P. Janu. (). « Analysis of CANaerospace Protocol Communication Quality in Aviation System, » adresse : https://www.researchgate.net/publication/275997574_Analysis_of_CANaerospace_Protocol_Communication_Quality_in_Aviation_System.

[22] V. W. i. VID 150259 Creation. (). « AIRSPEED DEFINITION, » adresse : https://mediawiki.ivao.aero/index.php?title=Airspeed_definition.

[23] I. Peppler, *From the ground up*. Aviation Publishers Co. Limited, 1996, isbn : 0969005490.

[24] S. I. I. D. France. (). « Les Instruments de Radionavigation, » adresse : https://aeroclub-montpellier.org/wp-content/uploads/Doc_Formation/fiche_Radionavigation2.pdf.

Chapitre 5

Annexe A

5.1 Cockpit et commandes de base

Le centre de pilotage ou cockpit d'un Zlin 142 est un biplace de conception simple comportant des instruments de mesure de base nécessaires au vols et de différentes commandes de pilotage présentes sur la majorité des aéronefs.

On distingue les compartiments suivant :

1. Instrument de vol, de navigation et de contrôle moteur.
2. Instrument de radionavigation.
3. Commande moteur.

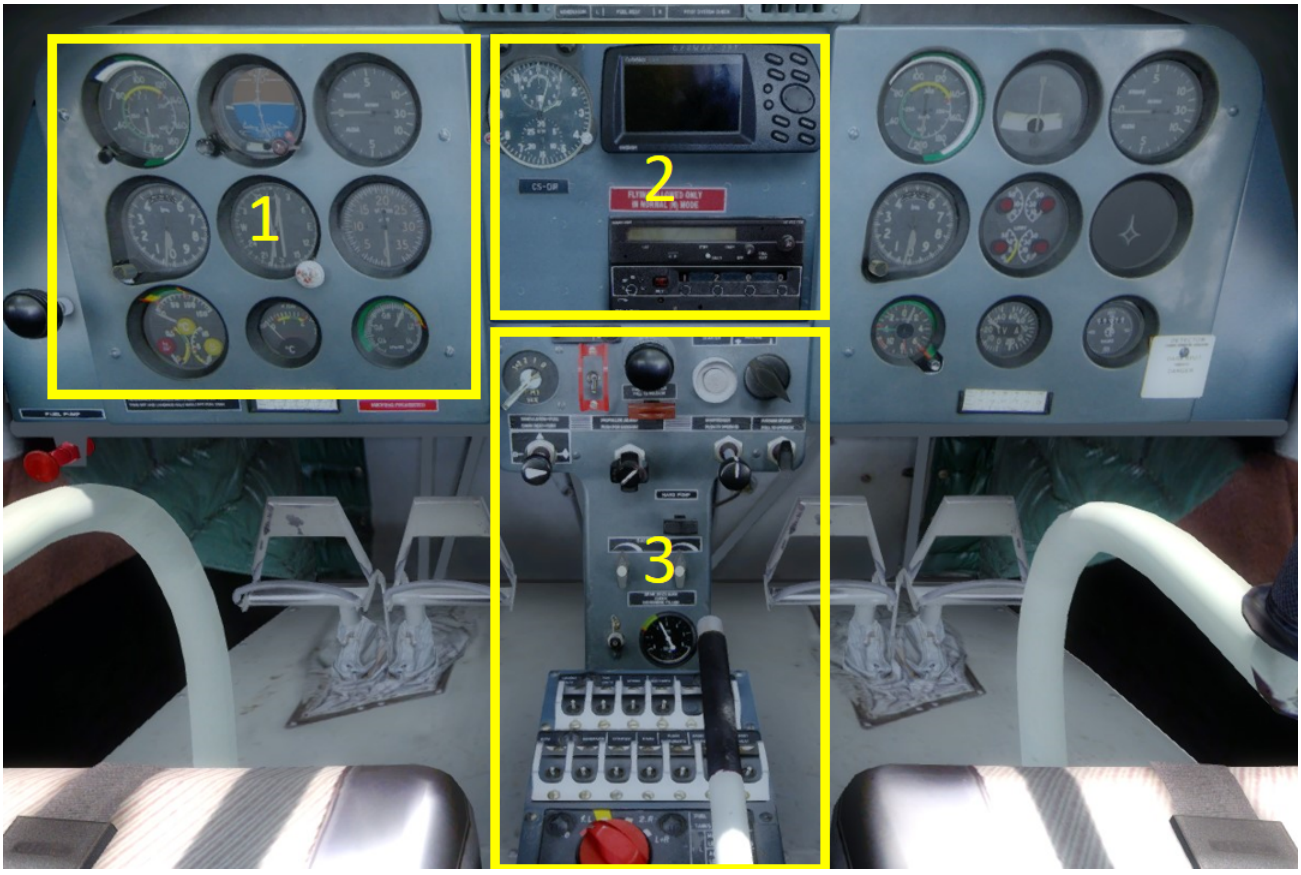


Figure 5.1 – Compartiments du cockpit

5.1.1 Instrument de vol, de navigation et de contrôle moteur

Figure 5.2 – Les indicateurs du bord

5.1.1.0.1 Anémometre(Air speed Indicator)

L'anémomètre fournit la vitesse de déplacement de l'avion par rapport à la masse d'air (Indicated Air Speed) en kilomètre par heure (km/h) ou en nœuds (kn). Il faut noter qu'il existe cinq vitesses pour un aéronef, on ne s'intéresse qu'à quatres d'entre elles :

Indicated Airspeed

Le système Pitot-statique comprend une ou plusieurs sondes Pitot faisant face au flux d'air entrant pour mesurer la pression de Pitot et un ou plusieurs orifices statiques pour mesurer la pression statique dans le flux d'air. La vitesse anémométrique est dérivée de la différence entre la pression d'air dynamique du tube de Pitot, ou pression de stagnation, et la pression statique. Un anémomètre est un manomètre différentiel dont la lecture de la pression est exprimée en unités de vitesse plutôt qu'en pression. IAS est simplement ce que l'indicateur de vitesse indique, elle est aussi le point de départ de tous les autres calculs [22].

True Airspeed

TAS est la vitesse de l'aéronef par rapport à l'atmosphère. La vitesse et le cap réel d'un aéronef constituent sa vitesse par rapport à l'atmosphère. La TAS est la véritable mesure des performances des avions en croisière, ainsi répertoriée dans les spécifications des avions, les manuels, les comparaisons de performances, les rapports des pilotes et toutes les situations où les performances réelles doivent être mesurées [22].

Calibrated Airspeed

CAS est la vitesse indiquée corrigée pour les erreurs d'instrument, les erreurs de position (dues à une pression incorrecte au port statique) et les erreurs d'installation. En vol propre, les erreurs de position et d'instrument sont généralement minimes. Cela signifie que la valeur CAS est presque égale à l'IAS.

Ground Speed

La GS est la vitesse de l'aéronef par rapport au sol. Cette vitesse est la combinaison du vecteur TAS de l'aéronef et du vecteur vitesse du vent à l'altitude de l'aéronef.

— GS = TAS + Vw

— GS = vitesse sol

— Vw = vecteur vitesse du vent

— TAS =True Airspeed

Cette vitesse est mesurée par le radar du contrôleur aérien [22].

5.1.1.0.2 Horizon artificiel(Gyro Horizon)

L'horizon artificiel ou indicateur d'assiette mesure l'assiette de l'aéronef par rapport à l'horizon c'est-à-dire les angles de tangage (pitch) et roulis (roll). 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.

5.1.1.0.3 Variomètre(Rate of Climb)

Le variomètre fournit la vitesse verticale Vz de l'avion, c'est-à-dire la composante verticale de la vitesse de l'avion par rapport à la masse d'air. Le variomètre est utile au pilotage

pour déterminer soit une vitesse ascensionnelle Vz supérieure à 0, soit un taux de descente Vz inférieure à 0, il est possible de l'utiliser également pour contrôler le vol en palier. L'unité la plus utilisée pour la vitesse verticale est les centaines de pieds par minute (100 Ft/min).

5.1.1.0.4 Altimètre

L'altimètre fournit une altitude-pression. Cette mesure est basée sur la décroissance de la pression atmosphérique lorsque l'altitude augmente. L'altimètre est en réalité un baromètre gradué en altitudes.

Il indique, soit l'altitude, en pieds (Ft), de l'avion (1m=3.3Ft ; 1000 Ft=910m) quand l'instrument est réglé sur la pression au niveau de la mer (QNH), ou bien la hauteur s'il est réglé sur la pression au sol (QFE) ou bien un niveau de vol standard quand il est réglé sur la pression standard de 1013 hPa.

5.1.1.0.5 Conservatreur de Cap (Directional Gyro)

Il indique le cap magnétique suivi par l'avion.

C'est un instrument gyroscopique, alimenté par la pompe à vide, qui permet de pallier à l'instabilité de la boussole de l'avion (au-dessus du cockpit). Le problème est que, comme tout gyroscope, cet instrument dérive dans le temps, et il faut donc le recaler régulièrement à l'aide du bouton en bas à droite dans une phase calme du vol en comparant avec le cap donné par la boussole.

5.1.1.0.6 Tachymètre(Tachometrer)

Il indique la vitesse de rotation du moteur (en tr/min)

5.1.1.0.7 Three Point Indicator

Il affiche 3 paramètres :

- Température de l'huile du moteur en celsius.
- Pression de l'huile du moteur en kg/cm^2
- Pression du carburant du moteur kg/cm^2.

.

5.1.1.0.8 Température des Têtes de Cylindre

Une jauge de température de culasse (CHT) mesure la température de la culasse d'un moteur. Couramment utilisée sur les moteurs refroidis par air, la jauge de température de la tête affiche le travail que le moteur effectue plus rapidement qu'une jauge de température d'huile ou d'eau. Lorsque le moteur fonctionne à haute vitesse ou en montée, la température de la tête augmente rapidement. Le compteur peut être numérique ou analogique.

5.1.1.0.9 Manomètre du Collecteur (Manifold Pressure)

Le manomètre du collecteur indique la pression de l'air dans le collecteur d'admission du moteur. Celui-ci est une indication de la puissance développée par le moteur. Plus la pression du mélange air-carburant entrant dans le moteur est élevée, plus il peut produire de puissance. Pour les moteurs à aspiration normale, cela signifie qu'une indication proche de la pression atmosphérique est le maximum. Les moteurs turbocompressés ou suralimentés pressurisant l'air mélangé au carburant, de sorte que les indications de pleine puissance sont supérieures à la pression atmosphérique. La plupart des manomètres de collecteur sont étalonnés en pouces de mercure.

5.1.1.1 Instrument de radionavigation

5.1.1.1.1 Radio Communication

Les radiocommunications aéronautiques sont utilisées pour les communications entre les pilotes et le personnel des stations au sol, la bande aéronautique VHF s'étend de 118 MHz à 136 MHz. L'instrument fonctionne avec deux fréquences, "COM1" et "COM1 Standby", on passe de l'une à l'autre en appuyant sur le bouton "Frequency Swap".

5.1.1.1.2 Transpondeur (Communication)

Un transpondeur est un dispositif électronique qui émet une réponse quand il reçoit une interrogation par radio. En aéronautique, les avions possèdent des transpondeurs pour aider à leur identification par les radars et aussi comme système anti-collision [23].

5.1.1.1.3 Radionavigation

Les informations des récepteurs de radionavigation sont affichées sur les instruments de radionavigation qui sont pour la plupart listés ci-dessous :

— ADI EADI,
— ADF,
— HSI & EHSI,
— RMI,
— CDI,
— EFIS,
— PFD,
— ND.

Les instruments de navigation sont :

— Récepteur ILS : Un système d'atterrissage aux instruments est une aide d'approche de piste de précision utilisant deux faisceaux radio pour fournir aux pilotes un guidage vertical et horizontal pendant l'approche à l'atterrissage. Le localisateur (LOC) fournit un guidage azimutal, tandis que la pente de descente (GS) définit le profil de descente verticale correcte. La fréquence ILS du Localizer est programmée sur la radio NAV1, tandis qu'on reçoit l'ILS sur le CDI, HSI, ND/EHSI, PFD/EADI.

— Récepteur VOR : Un récepteur VOR permet de déterminer son relèvement magnétique par rapport à une station au sol (balise émetteur VOR dont la position est connue), et donc le radial, gradué en degré d'arc, sur lequel le récepteur (donc l'avion) est situé. Par déduction, il permet de suivre n'importe quelle route passant par la station, que ce soit en rapprochement ou en éloignement de celle-ci, ou même de déterminer la position exacte de l'avion en utilisant deux balises VOR. La fréquence VOR est programmée sur deux fréquences radio NAV1 et NAV2, tandis qu'on reçoit les signaux VOR sur le CDI, HSI, ND/EHSI, RMI.

— Récepteur NDB : Une balise non directionnelle ou Non Directional Beacon (NDB) est une station radio localisée en un point identifié, et utilisée en tant qu'aide à la navigation aérienne ou maritime. Les NDBs sont exploités dans une plage de fréquences comprises entre 190 et 1 750 kHz (en Europe comprises entre 255 et 525 kHz). Elles sont situées aux abords des aérodromes et aéroports. La fréquence NDB est programmée sur le sélecteur de fréquence radio ADF, tandis qu'on reçoit les signaux NDB sur le ADF, HSI, ND/EHSI, RMI [24].

5.1.2 Commande moteur

(a) Magneto Swtich,
(b) Main Switch (Commutateur principal de la batterie),
(c) Throttle Control,
(d) Starter,
(e) Mixture Control,
(f) Propeller Control,
(g) Compressor,
(h) Parking Brakes,
(i) Landing Lights,
(j) Taxi Lights,
(k) Rotating Beacon Lights,

(l) Navigaion Lights,

(m) Inverter,

(n) Batery Switch (alimenter les instruments électrique),

(o) Genrator,

(p) Master Ignition,

(q) Master Avionic,

(r) Instruments (allumer les instruments électrique),

(s) Pitot Heat,

(t) Fuel Tank Selector,

(u) Flaps Switch.

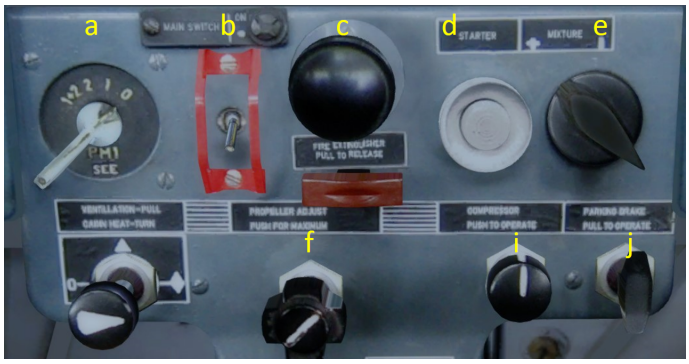


Figure 5.3 – Commande du moteur

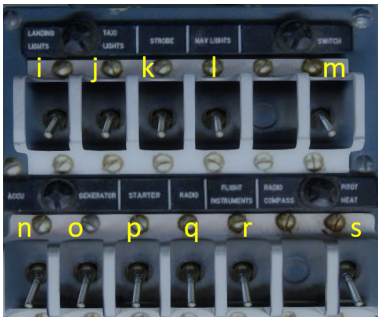


Figure 5.4 – Switch Panel

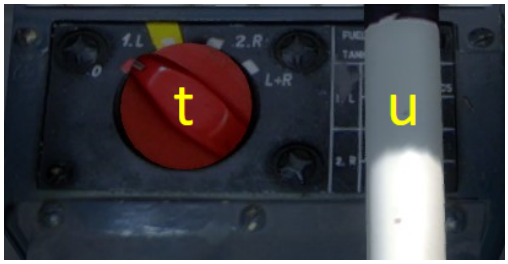


Figure 5.5 – Tank Switch et Flaps Switch

5.2 Périphérique CAN

5.2.1 Module USB-CAN

USB-CAN module est une solution d'interface de bus CAN très économique pour connecter un bus CAN à un PC via USB. Les messages CAN sont transmis de manière transparente, de sorte que le module peut prendre en charge tous les types de protocoles basés sur CAN de niveau supérieur, tels que CANopen, SDS, DeviceNet ou CAN-Avionic. Le cœur du module est un processeur 32 bits rapide avec tampon de messages intégré qui peut assurer une communication fiable même sous une charge élevée.AFMS

Figure 5.6 – Module USB-CAN Systec sysWorxx

5.2.2 CAN controller

Le MCP2515 est un contrôleur CAN qui implémente la spécification CAN, version 2.0B. Il est capable de gérer la transmission et la réception à la fois des données étendues et trames distantes. Le MCP2515 a deux masques d'acceptation et six filtres d'acceptation qui sont utilisés pour filtrer les messages indésirables et pour réduire la surcharge sur le MCU hôte. Le MCP2515 interfaces avec des microcontrôleurs (MCU) via une Interface série (SPI).

Figure 5.7 – CAN controller MCP2515

5.2.3 CAN transceiver

Le MCP2551 est un émetteur-récepteur CAN haute vitesse, tolérant aux pannes d'appareil servant d'interface entre un CAN contrôleur et le bus CAN physique.

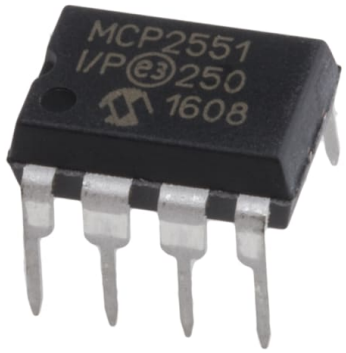


Figure 5.8 – CAN transceiver MCP2551

5.3 CANaerospace

5.3.1 Exemple de liste de distribution des identifiant CANaerospace

CAN ID	paramètre du système	Data Type	Unité	Notes
317	Air speed calibré	FLOAT/SHORT2	m/s	
321	angle du Heading	FLOAT/SHORT2	deg	+-180o
1070	Altitude Radio	FLOAT/SHORT2	m	

Table 5.1 – Exemple de liste de distribution des identifiant CANaerospace

Chapitre 6

Annexe B

6.1 Systémes temps-réels

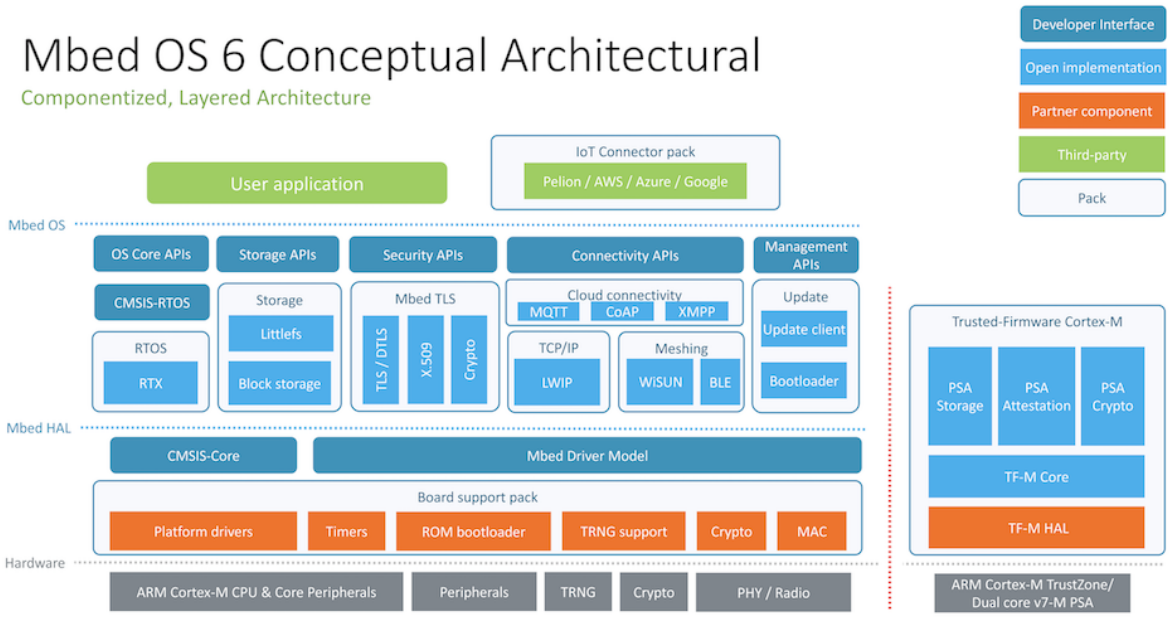


Figure 6.1 – Organigramme d'architecture de Mbed OS 6 [13]

Mbed OS est un système d'exploitation open source pour les plateformes utilisant des microcontrôleurs Arm conçu spécifiquement pour les appareils Internet des objets (IoT) : des appareils à faible puissance et contraints qui doivent se connecter à Internet. Mbed OS fournit une couche d'abstraction pour les microcontrôleurs sur lesquels il s'exécute, afin que les développeurs puissent se concentrer sur l'écriture d'applications C / C ++ qui appellent des fonctionnalités disponibles sur une gamme de matériels. Les applications Mbed OS peuvent être réutilisées sur n'importe quelle plateforme compatible Mbed.

Avec un cœur RTOS basé sur le CMSIS-RTOS RTX open-source largement utilisé, Mbed OS prend en charge l'exécution de logiciels en temps réel déterministes et multithread. Les primitives RTOS sont toujours disponibles, permettant aux pilotes et aux applications de s'appuyer sur des fonctionnalités telles que les threads, les sémaphores et les mutex [13].

La classe EventQueue fournit une file d'attente flexible pour la planification des événements. Elle peut être utilisée pour la synchronisation entre plusieurs threads ou pour déplacer des événements hors du contexte d'interruption (exécution différée d'opérations de sécurité chronophages ou non ISR) [13].

La classe EventQueue est sécurisée pour les threads et ISR [13].

On peut utiliser les API "dispatch" et "dispatch_forever" pour exécuter des événements en attente. "break_dispatch" consiste à mettre fin à l'exécution des événements dans la file d'attente d'événements spécifiée [13].

6.2 Table des offsets

Liste des offsets						
Nom de l'instrument	Offset Address	Taille (byte)	Nom de la Variable	Description et comportement	CAN ID	Intervale de transmission
Air Speed	0x02BC	4(uint)	airSpeedOffset	*0.0001313 to convert to gauge value	315	12.5ms
Gyro Horizon	0x2F70	8(double)	attitudePitchOffset		311	12.5ms
	0x2F78	8(double)	attitudeBankOffset		312	12.5ms
Rate Of Climb	0x02C8	4(int)	verticalSpeedOffset	*0.004 to convert to gauge value	1124	12.5ms
Altimeter	0x3324	4(int)	altimeterOffset	*0.003048 to convert to gauge value	332	100ms
Directional Gyro	0x2B00	8(double)	headingGyroOffset	North=0, E=90, South=180, W=270	315	12.5ms
Tachometer	0x0896	2(ushort)	engineRpmRatio-Offset	engineRPM= Max*Ratio/1638400 to convert to gauge value	504	100ms
	0x34E8	4(uint)	engineRpmMax-Offset			
Three Point Indicator	0x08B8	2(ushort)	oilTemperature-Offset	*140/16384 to convert to gauge value	536	100ms
	0x08BA	2(ushort)	oilPressureOffset	*3.866885/16384 to convert to gauge value	532	100ms
	0x08BA	4(uint)	fuelPressureOffset	-309)*0.00071845) to convert to gauge value	684	100ms
Cylindre Head Temperature (CHT)	0x08E8	8(double)	chtTemprature- Offset	- 98.85)*0.3292/100) to convert to gauge value	540	100ms
Manifold Pressure	0x08C0	2(ushort)	manifoldPressur-Offset	+5304)*0.000034316 to convert to gauge value	528	100ms

Radio Communication	0x311A	2(ushort)	com1FrqOffset	4 last digits of the freaquency, coded in BCD, first digit ”1” is assumed	1100	1s event
	0x311A	2(ushort)	com1StandbyFrq-Offset		1101	1s event
	0x3123	1(byte)	comSwapOffset	bit numéro 4 bascule entre COM1 et COM1Standby 0b00001000	1103	event
Transponder	0x0354	2(ushort)	transponderFrq-Offset	4 digits of freaquency coded in BCD	1116	1s event
Magneto Switch	0x0892	2(ushort)	magnetosSwitch-Offset	0=Off, 1=Start, 2=Gen/Alt, 4=Start	1300	1s event
Main Switch	0x281C	4(uint)	mainSwitchOffset	0=Off, 1=ON	1301	1s event
Throttle Control	0x088C	2(short)	throttleControl- Offset	0 to +16384 (100%)	1302	1s event
Starter	0x3B00	4(int)	starterEngineOffset	Read Only,use magneto switch = 4 to write	1303	1s event
Mixture Control	0x0890	2(ushort)	mixtureControl-Offset	Engine 1 Mixture lever, 0 – 16384 (100%)	1304	1s event
Parking Brakes	0x0BC8	2(ushort)	parkingBrakeOffset	0=Off, 32767=On	1305	1s event
Propeller Control	0x088E	2(short)	propellerControl-Offset	–4096(0%) to +16384(100%)	1306	1s event

Lights switch	0x0D0C	2(ushort)	lightsSwitches- Offset	Bits from low to high :0 Navigation,1 Beacon,2 Landing,3 Taxi	1307	1s event
Generator switch	0x3B78	4(uint)	generatorSwitch-Offset	A 32-bit BOOL (0 = Off, 1= On)	1308	1s event
Avionic Master	0x2E80	4(uint)	avionicsMasterSwitch-Offset	A 32-bit BOOL (0 = Off, 1= On)	1309	1s event
Pitot Heat	0x029C	1(byte)	pitotHeatSwitch-Offset	(0 = Off, 1= On)	1310	1s event
Fuel Tank Selector	0x0AF8	2(ushort)	fuelTankSelector-Offset	0=None, 1=All, 2=Left, 3=Right. Read Only, use events to write	1311	1s event
Flaps Swtich	0x0BDC	4(uint)	flapsSwitchOffset	0 = 0%, 16383 = 100%	1312	1s event

Table 6.1 – Liste des offsets (complète)

1s event : envoi de la donnée chaque 1 seconde ou lors d'un changement d'état.

6.3 Les circuits des nœuds

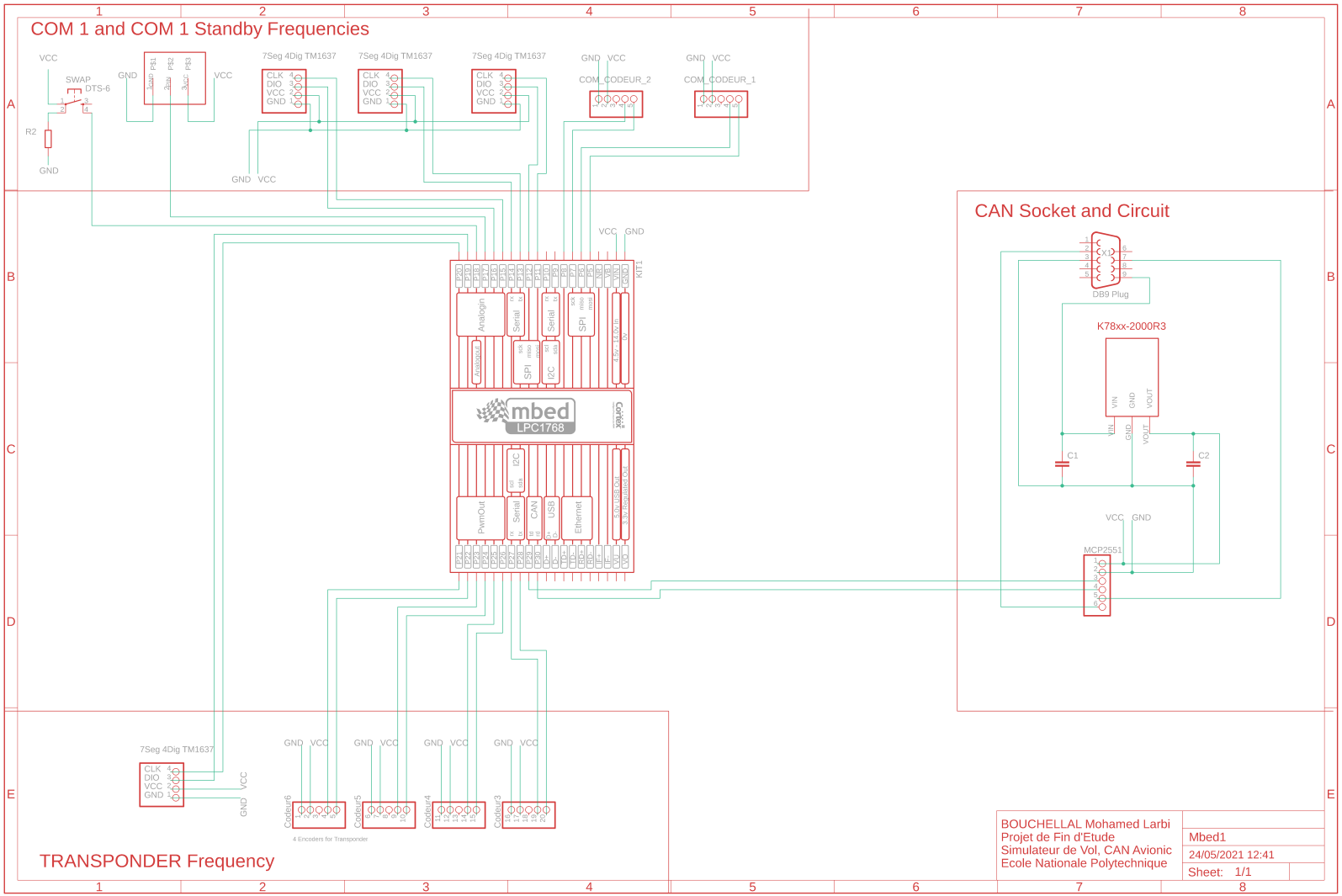


Figure 6.2 – Schématique du circuit Mbed 1

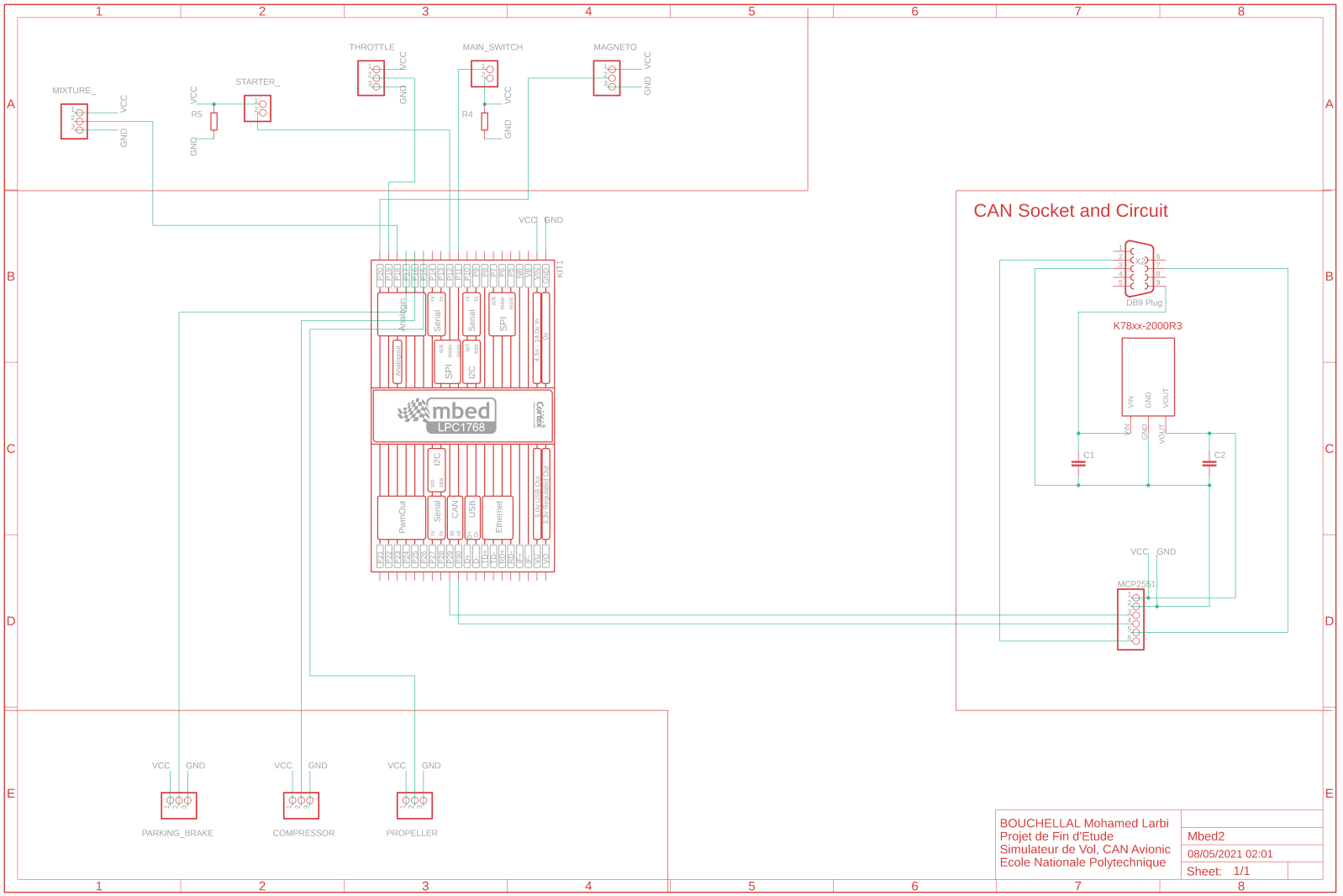


Figure 6.3 – Schématique du circuit Mbed 2

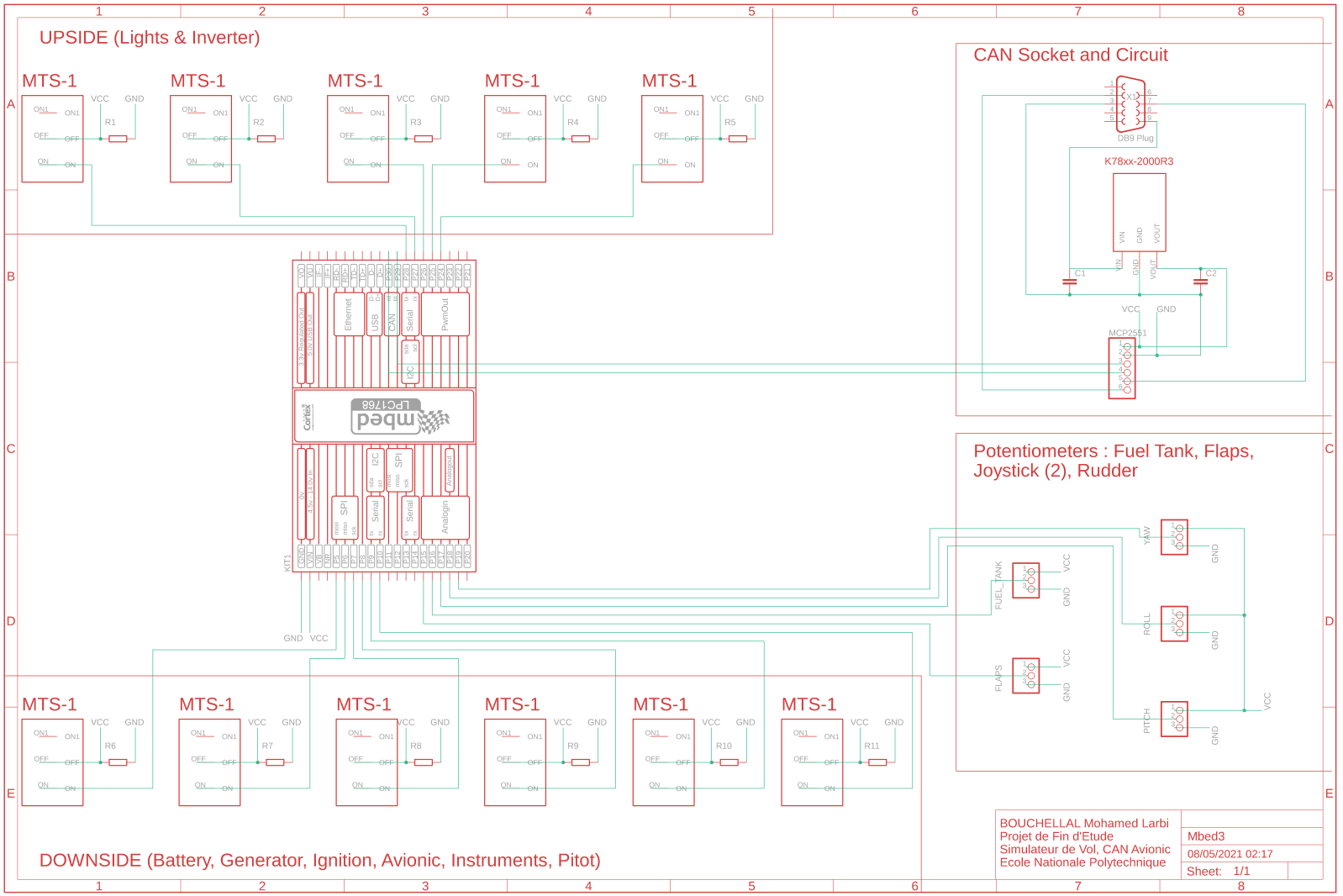


Figure 6.4 – Schématique du circuit Mbed 3