Julien CARCAU
Alicia FALCON
Arthur CHENU
Thomas GUILLEMOT
Tuteur : Denis MONNERAT

Février 2020

Projet Quadrora

Rapport de projet

Sommaire

Synthèse

Ce rapport décrit la réalisation d'un drone et de sa télécommande.
Ils ont été réalisés dans le cadre du projet tutoré du DUT Informatique de l'IUT de Sénart/Fontainebleau.
Le but recherché était d'arriver à assembler un drone et le programmer afin de le faire décoller, changer de direction (haut, bas, gauche, droite), atterrir en toute sécurité et se stabiliser en l'air.
De plus, une télécommande permettant de le piloter devait également être conçue, tout comme le drone.

Actuellement, la télécommande est entièrement fonctionnelle mis à part un problème matériel concernant un joystick.
Du côté du drone, il est possible de le piloter de manière assez basique, un moteur ayant subitement arrêté de fonctionner correctement. Il a visiblement été endommagé suite à un crash.

Vous pouvez retrouver l'ensemble du code source ainsi que des documents annexes sur notre GitHub :
https://github.com/MicrowavedCat/LoRaDrone

Sujet

Le but de ce projet était de réaliser un drone et sa télécommande, tant sur l'assemblage que sur la programmation.
Cette dernière communique avec le drone par ondes radio en utilisant la technologie LoRa sur une fréquence de 868 MHz, celle qui est utilisée en Europe (https://fr.wikipedia.or/wiki/LoRaWAN#Modulation_LoRa).
Parmi les nombreuses technologies de télécommunications disponibles sur le marché, le LoRa était la plus intéressante en raison de sa portée, qui est théoriquement de 8 km maximum, de sa consommation électrique, assez basse et de sa complexité d'intégration.
Ce projet à été réalisé en suivant le cahier des charges disponible en annexe.

Réalisation

Télécommande

Conception

La télécommande est constituée d’un PCB qui permet d'y disposer les composants. Tous ceux-ci sont reliés à un microcontrôleur (un ESP32). Cette télécommande est alimentée par une batterie Li-Po de 4000 mAh qui lui donne une autonomie de plusieurs heures. Afin de transmettre les données de pilotage des deux joysticks, elle communique avec le drone grâce à la technologie LoRa qu'un module (un E32-868T30S*) gère.

Elle dispose également de boutons, un bouton :

- ON/OFF qui, comme son nom l’indique, permet d’éteindre ou d’allumer la télécommande ;
- d’arrêt d'urgence permettant d'arrêter les hélices le plus rapidement possible ;
- sous chaque joystick qui, appuyés ensemble, permettent d'activer ou de désactiver la sécurité.

Quatre LED donnent des indications au pilote sur l'état de la télécommande.

Un ventilateur permet de refroidir les différents composants, en particulier le module LoRa qui, émettant une grande quantité de données, a tendance à beaucoup s'échauffer.

Logiciel

La partie logicielle de la télécommande à été réalisée par Julien et Alicia.
Le fonctionnement de la télécommande repose sur six threads qui permettent d’effectuer plusieurs traitements en parallèle qui sont :

- la réception de données ;
- l'envoi de données ;
- le contrôle des LED ;
- le contrôle de l’état des boutons gérant la sécurité ;
- le contrôle de l'état du bouton d’arrêt d’urgence ;
- la gestion des messages à envoyer au drone.

Contrôle des LED

Au total, il y a 4 LED : jaune, rouge, verte, et bleue. Chacune d'elles est contrôlée pour convenir à une certaine situation :

- jaune fixe : le drone et la télécommande sont allumés, mais la connexion n'a pas été établie ;
- vert clignotant rapidement (cinq fois par seconde) : le drone et la télécommande viennent de se connecter ;
- vert clignotant lentement : la connexion est établie mais la sécurité est enclenchée ;
- bleu fixe : la sécurité est enlevée, le drone peut décoller ;
- rouge clignotant : l’arrêt d’urgence a été enclenché.

Contrôle de la sécurité
On vérifie continuellement l'état de la sécurité. Son état change lorsque les deux joysticks de la télécommande sont enfoncés. Ainsi, le drone ne peut pas décoller si la sécurité est enclenchée.
Si celle-ci est activée alors que le drone est en l'air, il y aura deux situations possibles selon son altitude :
- à moins d’1 mètre, il se pose en douceur sans attendre ;
- à plus d’1 mètre, il se pose doucement après un certain temps.

Gestion de l'arrêt d'urgence
On vérifie également continûment si le bouton d’arrêt d’urgence a été enclenché. Lorsque celui-ci l'est, on envoie un message spécifique au drone : « STOP » pour qu’il coupe tous ses moteurs.

Gestion des messages
Selon les situations, on va envoyer différents messages pour le drone :
- "STOP" : bouton d'arrêt d’urgence enclenché ;
- "SECURITE" : envoyé si la connexion est établie et que la sécurité est activée ;
- "PAIR" : prévient le drone que la connexion est établie.

Communication

Conception

La communication entre le drone et la télécommande est réalise à l'aide du protocole LoRa. Le drone et la télécommande disposent chacun d'un module. Ce système permet à la télécommande d'envoyer les informations de pilotage au drone.

Partie logicielle

La communication entre un module LoRa et l'ESP32 ou un Raspberry est effectuée en UART*, à une vitesse de 9 600 bauds.

Drone

Conception

Le drone possède une plaquette PCB sur laquelle est disposé un Raspberry Pi Zero* (mini ordinateur disposant à peu près de la puissance de calcul d'un smartphone). Chacun des quatre ESC* est relié à un GPIO* du Raspberry contrôle la vitesse du moteur auquel il est rattaché. Ces derniers permettent de faire tourner les hélices.

Un accéléromètre et un télémètre ultrason permettent au drone de se repérer dans l'espace. Comme la télécommande, le drone dispose d'un module LoRa pour échanger des données avec cette dernière. L'ensemble des composants du drone sont alimentés par une batterie au lithium de 7 200 mAh.

Chaque ESC est relié, d'un côté, au Raspberry, et de l'autre, à un moteur. Une hélice est aussi présente sur chaque moteur.
Le fonctionnement d'un moteur brushless (par les impulsions électriques envoyées depuis un ESC) ne sera pas expliqué ici puisque cela sort du cadre de ce projet. Voici tout de même une animation montrant de manière assez claire le principe : https://www.youtube.com/watch?v=pXj1cxFoQMk.

Problème

A l'origine les hélices était toutes orientées de la même manière et les moteurs ne pouvaient les faire tourner que dans un seul sens. Toutes les hélices n'ayant pas le même forme (les hélices allant par paire en diagonale, deux hélices de deux diagonales différentes tournent donc dans des sens opposés), deux hélices attiraient le drone vers le haut et deux autres, vers le bas. Afin de remédier à ce problème, l'ordre des trois fils reliés aux moteurs a été modifié sur deux des quatre hélices afin de modifier leur sens de rotation.

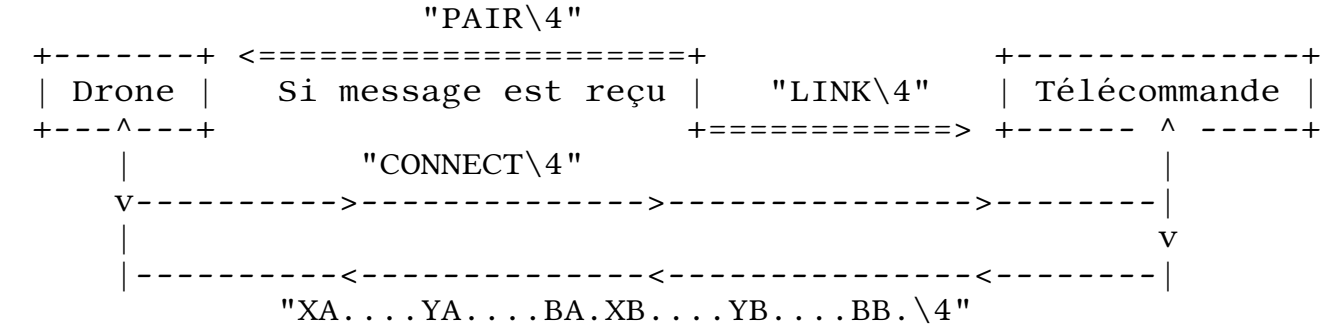
Logiciel

L'utilisation de la librairie WiringPi* a permis de réduire le volume du code concernant l'utilisation et la programmation des composants ci-dessous, évitant ainsi de définir par projection de la mémoire virtuelle de chaque élément branché sur les GPIO du Raspberry (ce principe utilise une partie de la mémoire, en l'occurrence sur Raspberry le fichier de chemin "/dev/mem", et permet d'effectuer des opérations bits à bits sur leur adresse en mémoire pour déterminer quel périphérique doit être utilisé et de quels sont les interactions à effectuer dessus) :

Le code du drone est séparé en quatre parties principales, la gestion :
- de la communication entre le drone et la télécommande ;
- des moteurs ;
- de l'accéléromètre ;
- du télémètre ultrason.

Concernant le LoRa, un flux de données (de chemin "/dev/ttyAMA0" sur le Raspberry) permet la réception et l'envoi de données avec lequel il est possible d'interagir en programmant un descripteur de fichier particulier, avec lequel peuvent interagir des fonctions de la librairie WiringPi, permettant l'échange de données en UART.
Les modules LoRa n'étant pas conçus pour fonctionner en full-duplex* mais seulement en half-duplex*, l'information ne peut circuler de manière simultanée du drone vers la télécommande et inversement.

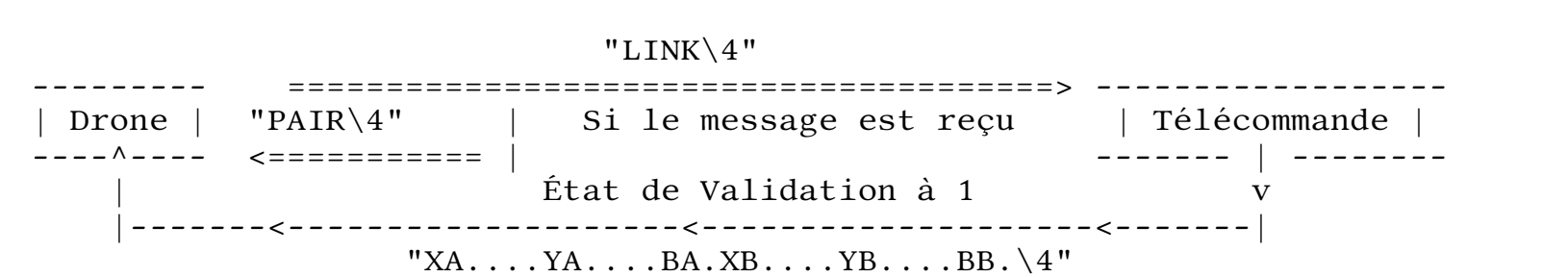
La communication était prévue de base pour que la télécommande demande de se connecter au drone dans un premier temps, le drone envoyait ensuite une réponse lorsque le message de connexion était correctement reçu.
Enfin le drone envoyait à rythme régulier (toutes les 2 secondes) des messages de type"keepalive" (afin de confirmer que la présence d'une connexion active), qui elle-même envoyait les messages de pilotage au format "XA....YA....BA.XB....YB....BB.\4", de manière simultanée.
Cela aurait permis de vérifier l'état de connexion entre le drone et la télécommande afin, par exemple, de prévenir l'utilisateur d'une éventuelle déconnexion.
(voir schéma ci-dessous)



Tous les messages reçus et envoyés sont des chaînes de caractères terminées par le caractère de fin de transmission (EOT) '\4', particulièrement adapté pour définir la fin des messages dans ce cas.

Pour remédier à ce problème, des demandes de connexion sont envoyées par le drone à la télécommande (définit par la chaîne de caractère "LINK\4"), si celle-ci lui retourne un message d'acceptation de connexion (défini par la chaîne de caractère "PAIR\4"), la communication quitte le mode écriture, ce qui permet d'activer le mode de réception de messages de pilotage (envoyés par la télécommande).
Il est possible de recevoir le message d'arrêt d'urgence "STOP\4", qui ordonne à tous les moteurs de se couper instantanément : il leur est simplement envoyé une valeur de "0".

Pour sécuriser un minimum les transmissions, les messages reçus par le drone sont filtrés par une fonction vérifiant si le message est bien du format "XA....YA....BA.XB....YB....BB.\4" (où .... est un nombre compris entre 0 et 4095, et .est un chiffre étant soit 0 ou 1).
(voir schéma ci-dessous)



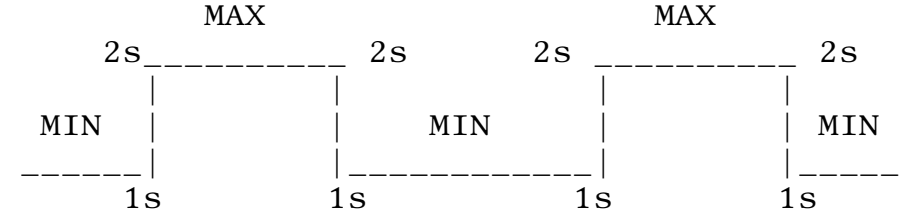
Par la suite cette chaîne de caractère est convertie en un tableau d'entiers évoqués ci-dessus.

Moteurs

Les hélices tournent grâce aux moteurs, commandés eux-mêmes par des impulsions électriques (dans notre cas, d'une amplitude d'environ 11 volts, dépendant de la charge de la batterie), il s'agit de l'utilisation de la technique PWM (Pulse Width Modulation, ou en français, Modulation par Largeur d'Implusion, soit MLI)
Dans des circuits analogiques, l'information peut être communiquée en faisant varier la tension électrique, par exemple, de 0 à 12 volts. Dans le cas d'un circuit numérique, ces variations ne sont pas réalisables "en l'état" et seulement deux états sont possibles : allumé (dans notre cas, 12 volts) ou éteint (0 V). La technique PWM* consiste donc à faire varier ces états, généralement très rapidement, afin de simuler un signal analogique. Ce phénomène est particulièrement visible sur une LED : plus la quantité de signaux envoyés est nulle, moins la LED brille.

Quatre GPIO (dont le mode est défini sur "sortie") sont chacun associés à un ESC, puis est écrite une puissance maximale suivit

d'une puissance minimale, séparés par un temps d'attente qui met à jour ces deux états de transition.
Il s'agit de la calibration des moteurs, permettant de définir la vitesse maximale de rotation qu'ils peuvent atteindre. La valeur minimale est à 0 et la valeur maximale a été fixée à 511 car 2^9 = 512 valeurs. La puissance dans les ESC étant gérés par des registres sur 32 bits, il y a donc une différence de 32 valeurs entre la puissance maximale et la puissance minimale à laquelle toutes les hélices tournent à la vitesse minimale, soit 511 - 32 = 479. Autrement, les moteurs sont coupés à une valeur inférieurs à 479, (ou peine à tourner sur eux-même). Les valeurs 511 et 479 ont été trouvée après de nombreux tests sur les moteurs en leur écrivant différentes vitesse manuellement.
(Voir le schéma ci-dessous)



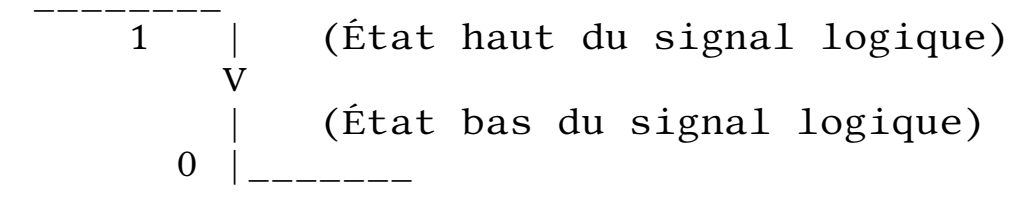
La création de threads, prenant en paramètre la fonction définissant un moteur et une structure concernant cette même fonction, permet de différencier la vitesse et le moteur lui-même, ce qui est nécessaire aux manœuvres directionnelles du drone. Lire une vitesse de rotation plus élevée sur deux moteurs et une vitesse moins élevée sur les deux autres permet d'orienter le drone lors du pilotage.

Une partie des données de pilotage, passées par la télécommande, et stockée dans un tableau puis convertie en données directionnelles interprétables par le drone, à l’échelle de vitesse de rotation des moteurs du drone. Les joysticks transmettent chacun deux valeurs allant de 0 à 4095, et une autre valeur pouvant être 0 ou 1. Il faut donc adapter celles comprises entre 0 et 4095, en valeurs comprises entre 479 à 511.

Télémètre à ultrasons

En ce qui concerne le télémètre à ultrasons, il est bien sûr nécessaire de définir puis de configurer les GPIO sur lesquels il est branché (en utilisant ici la librairie WiringPi).
Il est ensuite possible de déterminer lequel sera utilisé comme récepteur ou émetteur d'ondes ultrasons, en définissant, pour celui de la réception, le mode en tant que "sortie" et en mode "entrée" celui de l'émission.

Par la suite est effectué un "front descendant" sur le récepteur, ce qui correspond au passage de la mise à l'état haut à bas de son signal logique (signal physique pouvant prendre deux valeurs booléennes : "haut" ou "bas"). Cela permet de déterminer et lancer le signal d'horloge interne par une table de vérité (ou "haut" et "bas" correspondent respectivement à 1 et 0).
(Voir schéma ci-dessous)



Anisi, à la valeur 0, le signal d'horloge interne s'arrête et le temps est relevé par une fonction.
Celle-ci utilise des structures de temps et minutage, tel que "timeval", adaptées à cet effet, permettant ainsi de calculer le temps entre l'émission et la réception d'une onde.

Le temps précédemment calculé et l'état du signal logique permettent donc de déterminer quand l'onde a été émise et quand elle est revenu au récepteur, ce qui abouti au calcul de la distance d'un objet, sous le drone, par rapport au télémètre : distance = (réception - émission) / 58.
Cette donnée est stockée dans une variable réutilisée pour l’atterrissage automatique du drone.

Accéléromètre

L'accéléromètre est le seul composant à avoir été programmé sans la librairie WiringPi, mais en I²C (Inter-Integrated Circuit),car le modèle ADXL345* est le seul composant du projet à fonctionner en I²C , c'est à dire qu'il se connecte aux GPIO du Raspberry nommés SDA et SCL. Il s'agit d'une méthode encore plus optimisée et très peu coûteuse en instructions (proche de l'assembleur ARM), consistant à utiliser un flux de données "/dev/i2c-1", contenant les informations d'un composant branché, dont notamment son adresse I²C qui est 53 (Cela se vérifie en faisant "sudo i2cdetect -y 1").

L'accéléromètre est ensuite passé en mode "esclave" : le système maître-esclave permet, par son adresse, de déterminer un périphérique comme étant récepteur à son maître, qui lui est émetteur (le récepteur étant en l'occurrence l'ADXL345, et l'émetteur étant le Raspberry Pi Zero).

En écrivant dans le flux "/dev/i2c-1", dit le "bus I²C", il est possible de configurer, par création de tableaux servant de registres, les paramètres de notre accéléromètre sous forme d'adresses hexadécimales que l'on écrit dans le bus.
Après écriture la configuration, les informations que relève l'accéléromètre 3 axes paramétré sont stockées sous forme de valeurs entières, allant de -511 à 511 et prenant en compte l'espace réservé de 256 (pour d'éventuels autres composants), dans un tableau à trois dimensions, pour les données d'accélération linéaire sur les axes x, y et z.

Test de vol

Le 1er test du drone en conditions réelles a servi d'expérimentation, ayant abouti à un problème mais également à un résultat concluant. Le montage du drone n'étant pas terminé, un installation de fortune avec du scotch a été réalisée, afin de fixer à l'armature la batterie et le Raspberry contrôlé par un clavier sans fil. Très peu équilibré de cette manière, le drone s'est rapidement retourné et plaqué au sol après décollage.
Toutefois, ce test a permis de s'assurer que le drone pouvait largement supporter son propre poids et décoller sans encombre à la condition qu'il ne soit pas déstabilisé par le placement de ses composants lourds comme la batterie, le vent ou qu'un utilisateur le stabilise manuellement avec la télécommande.

Tableaux des notions utilisées

Notions	Julien	Arthur	Alicia	Thomas
Codage de l'information : nombres et caractères. Arithmétique et traitements associés	x	x	x	x
Architecture générale d'un système informatique	x	x	x	x
Types et caractéristiques des systèmes d'exploitation	x	x	x	x
Utilisation d'applications clientes réseau : messagerie, transfert de fichiers, terminal virtuel, répertoires partagés	x	x	x	x
Gestion des processus (création, destruction, suivi, etc.), des fichiers (types, droits, etc.) et des utilisateurs (caractéristiques, création, suppression, etc.)	x	x	x	x
Principes de l'installation et de la configuration d'un système	x	x	x	x
Notion de sous-programmes : nommage des variables, assertions, pré- et post-condition	x	x	x	x
Notion de types et de données, définitions de types simples, structures séquentielles à accès direct	x	x	x	x
Implantation des algorithmes dans un langage de programmation	x	x	x	x
Gestion des cas d'erreurs	x	x	x	x
Notion d'accès séquentiel et d'accès direct	x	x	x	
Connaître la performances des algorithmes utilisés	x	x	x	x
Écriture et lecture dans des fichiers	x	x	x	x
Études et analyses documentaires	x	x	x	x
Étude d'un système à microprocesseur ou microcontrôleur (réel ou simulé) avec ses composants (mémoires, interfaces, périphériques, etc.)	x	x	x	x
Utilisation de briques logicielles, d'interfaces de programmation (API : Application Programming Interface), de bibliothèques	x	x	x	x
Modélisation objet pour l'analyse et la conception détaillée par exemple en UML (Unified Modeling Language) : diagramme de séquences,de cas d'utilisation,d'activité)	x	x	x	x
Gestion des versions dans le développement	x	x	x	x
Réalisation de la documentation utilisateur	x	x	x	x

Documentation du code	x	x	x	x
Constitution d'une équipe	x	x	x	x
Gestion du temps et des délais	x	x	x	x
Utilisation d'outils de suivi de version	x	x	x	x
Approche du calcul des coûts	x			
La démarche projet	x	x	x	x
L'équipe projet : répartition des rôles	x	x	x	x
La définition des tâches, planification et enchaînement, attribution des ressources	x	x	x	x
Décrire le matériel informatique, son fonctionnement et ses applications	x	x	x	x
S'exprimer sur l'informatique en général	x	x	x	x
Utiliser la terminologie adéquate et les structures grammaticales adaptées	x	x	x	x
Approfondir sa culture générale et scientifique	x	x	x	x
Mise en œuvre des tâches : processus lourds et légers	x	x	x	x
Systèmes d'entrée-sortie	x	x	x	x
Rédaction précise d'un cahier des charges	x	x	x	x
Réalisation de la solution technique retenue	x	x	x	x
Synthèse, explication, reformulation	x	x	x	x
Présentation orale du projet	x	x	x	x
Affirmer ses choix et les argumenter	x	x	x	x

Résultat

L’objectif était de réaliser un drone et sa télécommande, comme indiqué dans le cahier des charges , les tâches ont été hiérarchisées par ordre d'importance grâce à la méthode MoSCoW.

Must :
communications entre les modules LoRa ;
montage de la télécommande et du drone ;
communication avec les ESC (rotation des hélices) ;
arrêt d’urgence ;
récupération et transmission des données à tous les périphériques de la télécommande.
Should :
atterrissage "intelligent" (détection de la distance au sol).
Could :
rotation à 360°.
Would :
Caméra (encodage vidéo et transmission du flux pour affichage dans une page HTML);
GPS (retour au point de départ en cas de déconnexion de la télécommande).

Vous trouverez ci dessous l'avancement des ces différentes tâches

Communications entre les modules LoRa :

La communication entre les deux appareils est à présent parfaitement fonctionnelle, mais a dû subir des modifications par rapport à ce qui était prévu initialement. Par le manque de documentations sur les module LoRa*, nous nous sommes rendus compte assez tard que ces-derniers ne fonctionnaient qu'en Half-Duplex, c'est à dire que la transmission d'information ne pouvait s'effectuer que dans un sens, et non dans les deux simultanément.

A présent, le drone envoie un 1er message reçu par la télécommande, celle-ci renvoie un message de confirmation et cela met le drone dans un état de validation, d'acceptation de messages de pilotage. Ces derniers étant traduite en coordonnée de pilotage, si le format évoqué précédemment est respecté.

Montage de la télécommande et du drone :

Télécommande :
La télécommande est complètement montée et est opérationnelle.

Drone :
Malgré un problème sur un des moteurs le montage complet du drone est réalisé. Problème majeur que nous tentons de régler.

Communication avec les ESC (rotation des hélices) :

La calibration des moteurs a été effectué correctement, chaque moteur peut recevoir une puissance traduite par la vitesse de rotation de ces derniers allant de 479 à 511.
Le problème matériel du moteur enrayé nous bloque actuellement, et nous tentons d'y remédier par tous les moyens. Des tests de vol en conditions réel sont actuellement complexes avec 3 moteurs sur 4.

Arrêt d'urgence :

L'arrêt total des moteurs à la réception du message d'arrêt d'urgence est fonctionnel.
Lors du codage la calibration, essentielle au fonctionnement des impulsions par largeurs de modulation dans les moteurs par l'intermédiaire des ESC, une fonction transmettant une valeur de vitesse de rotation au moteur a était mise en place, et a donc été réutilisée pour transmettre la valeur 0, coupant ainsi tous les moteurs instantanément.

Récupération et transmission des données à tous les périphériques de la télécommande :

<Analyse des résultats>

Atterrissage intelligent (détection de la distance au sol)

L'atterrissage automatique est en cours de finition, malgré le problème matériel du moteur enrayé, qui a été rencontré.

Le télémètre une fois programmé relève des données de proximité par rapport au sol, car ce dernier est placé sous le drone. Ces données sont transmises ensuite à une fonction qui relève la vitesse actuelle et la décrémente de 1 toutes les 2 secondes, pour atterrir en douceur au sol, et coupe toutes les moteurs lorsqu'il est à 20 cm du sol.

Il reste à faire en sorte que les données de proximité se relèvent en permanence, dans une boucle, car celles-ci ne se relèvent qu'une fois après exécution. Cela est due au structure de timer défini avec l'horloge interne.

Rotation à 360°

La rotation à 360° est en cours de développement mais a été retardée à cause du moteur enrayé. Le fonctionnement pour y parvenir sera basé sur le même principe que l'orientation directionnelle du drone, avec plus ou moins de puissance dans tel moteur par modification de la tension.

Caméra (encodage vidéo et transmission du flux pour affichage dans une page HTML)

Aucune caméra n'a été ajouté au drone, par manque de temps et de la priorisation des tâches plus importantes pour le fonctionnement du drone.

GPS (retour au point de départ en cas de déconnexion de la télécommande)

Aucun GPS n'a été ajouté au drone, par manque de temps et de la priorisation des tâches plus importantes pour le fonctionnement du drone.

Conclusions

Conclusion Générale

Le projet Quadrora nous a apporté une expérience inédite et des connaissances dans le domaine de la programmation impérative appliquée aux systèmes embarqués, ainsi qu'en électronique et en hardware.
Chacun avait des affinités dans un domaine ou plus, concernant soit le drone soit la télécommande. La synergie de nos compétences nous a permis de réaliser les composants essentiels au projet.
Le code a été conçu de manière particulièrement modulaire sur le drone et la télécommande et adapté en fonction de l'architecture système disponible de chaque.
La télécommande étant gérée par un microcontrôleur en Arduino, il ne dispose pas d'un réel système de plusieurs fichier mais d'un seul fichier binaire à compiler.
Il était possible de l'agencer sous forme de Makefile.(fichier dictant l’ordre de compilation)
De cette manière il a été possible de structurer proprement et efficacement notre code, en rappelant plusieurs fichiers gérant chacun un composant du drone. Ces derniers sont composés des fonctions principales toutes rappelées dans un fichier principal "main", lançant chacune de celles-ci simultanément.

Nous avons cherché à concevoir le code le plus optimisé possible, pour lui permettre une exécution plus rapide, moins laborieuse, prenant le moins d'espace mémoire possible et le moins défaillant pendant son lancement.
Nous avons beaucoup appris de ce projet et a été ravi de pouvoir y participer.

Conclusion de Julien CARCAU

Ce projet m'a permis de mettre en application des méthodes que j'avais vues dans la théorie mais pas encore ou insuffisamment dans la pratique. C'est ainsi que la partie électronique du projet, l'idée de l'assemblage d'une télécommande, a pu voir le jour, au prix néanmoins de défauts : j'estime par exemple que le problème concernant les joysticks est dû à des interférences dont j'avais largement sous-estimé l'importance.
Concernant la partie logicielle de ce projet, utiliser le langage C m'a permis de revoir et d'approfondir des notions que j'avais déjà rencontrées ainsi que d'en explorer de nouvelles. De plus, les programmes des 2 appareils (Raspberry et ESP32) n'étant pas exactement écrits dans le même langage (C pour le premier, Arduino pour l'autre), j'ai pu observer des différences, intéressantes et utiles, dans la façon de les écrire.
Enfin, travailler en pair-programming a été une expérience très intéressante, en particulier dans la façon de voir le projet au fil de son développement.

Conclusion d'Alicia FALCON

Ce projet était particulièrement intéressant pour moi car il m’a permis d’acquérir des compétences dans un domaine qui n’est pas au programme du DUT informatique. Comme l’assemblage de la télécommande avec tous ses composants et la communication entre elle et le drone, car pour ma part, je me suis principalement chargée de la télécommande.
Bien que je ne compte pas forcément aller dans cette branche plus tard, j’ai tout de même aimé faire ce projet en équipe. De plus, cela a été une expérience fructueuse qui me servira sans aucun doute.
Bien que nous n’ayons pas pu atteindre tous nos objectifs, à cause de plusieurs problèmes rencontrés, je suis contente de ce qu’on a pu réaliser.

Conclusion d'Arthur CHENU

Ce projet a été extrêmement intéressant pour moi, et je compte toujours travailler dessus même après la présentation orale, pour continuer d'y ajouter de nouvelles fonctionnalités (comme une Intelligence Artificielle auto-directionnelle ainsi qu'une application mobile de pilotage).

Appréciant, de surcroît, utiliser le langage C, j'ai ainsi pu l'appliquer concrètement dans la manipulation d'objets physiques, et particulièrement dans les systèmes embarqués, domaine qui me passionne énormément.
J'ai appris le fonctionnement électronique de certains composants essentiels dans les systèmes embarqués, et les différentes manières de les programmer grâce aux nombreux protocoles et techniques existants (PWM, SPI, UART, I²C, LoRa, Projection d'adressage virtuelle, librairie WiringPi).

L'intérêt principal d'avoir utilisé la librairie WiringPi a été de nous initier aux fonctions présentes en Arduino, sur microcontrôleur, un autre support de programmation bien plus souvent utilisés pour réaliser un système embarqué, et donc par extension nous former à la programmation sur microcontrôleur.

J'ai appris, notamment au contact de l'expertise de Julien dans le domaine de l'électronique, comment faire correctement une soudure, des branchements sur les GPIO d'un Raspberry ou faire des assemblages sur plaquette PCB, et quel matériel est nécessaire dans telle situation.

Ce projet est une ébauche de ce que je souhaite poursuivre comme étude, et une expérience significative dans ce domaine.

Conclusion de Thomas GUILLEMOT

Cette expérience été pour moi très enrichissante, en effet travailler au côté de Julien ma ouvert les portes de l'électronique,(sujet très peu abordé en DUT Informatique). J'ai assimilé de nouvelles compétences comme la soudure, le montage des composants électronique, la communication par ondes radio,qu'un étudiant classique sortant de DUT n'aurais pas forcément.
Sortir de sa zone de confiance n'est pas chose facile mais je suis satisfait d'avoir franchis le pas.
Au lieu de mettre à profit des compétences que je maîtrisais pour un projet qui ressemblerais à quelque chose de vu et revu, j'ai choisi de suivre Julien et Arthur dans leur idée originale, et je n'en suis pas déçu.
Concevoir un drone change d'une simple application sur un ordinateur, le moindre défaut peut porter préjudice à l'ensemble de la structure du projet et être très problématique (crash, perte de contrôle, accident de personne) on en a fait l'expérience, heureusement sans soucis majeur. Prendre en compte

le risque du projet ma permis de le contextualiser et de comprendre les enjeux d’une telle réalisation.
Grâce à la modularité de notre projet chaque partie peut être réutiliser indépendamment et savoir que notre travail sera peut être réutiliser pour d’autre application me réjouit. Pourquoi faire un projet pour l’oublier ensuite? Comprendre comment fonctionne un Raspberry ma également préparé pour mon stage dans lequel je devrais utiliser 96 Raspberry.
Concevoir et réaliser ce drone et cette télécommande ma permis de confirmer mon goût pour l'informatique ainsi quz de ma poursuite d’études dans cette même voie.

Glossaire

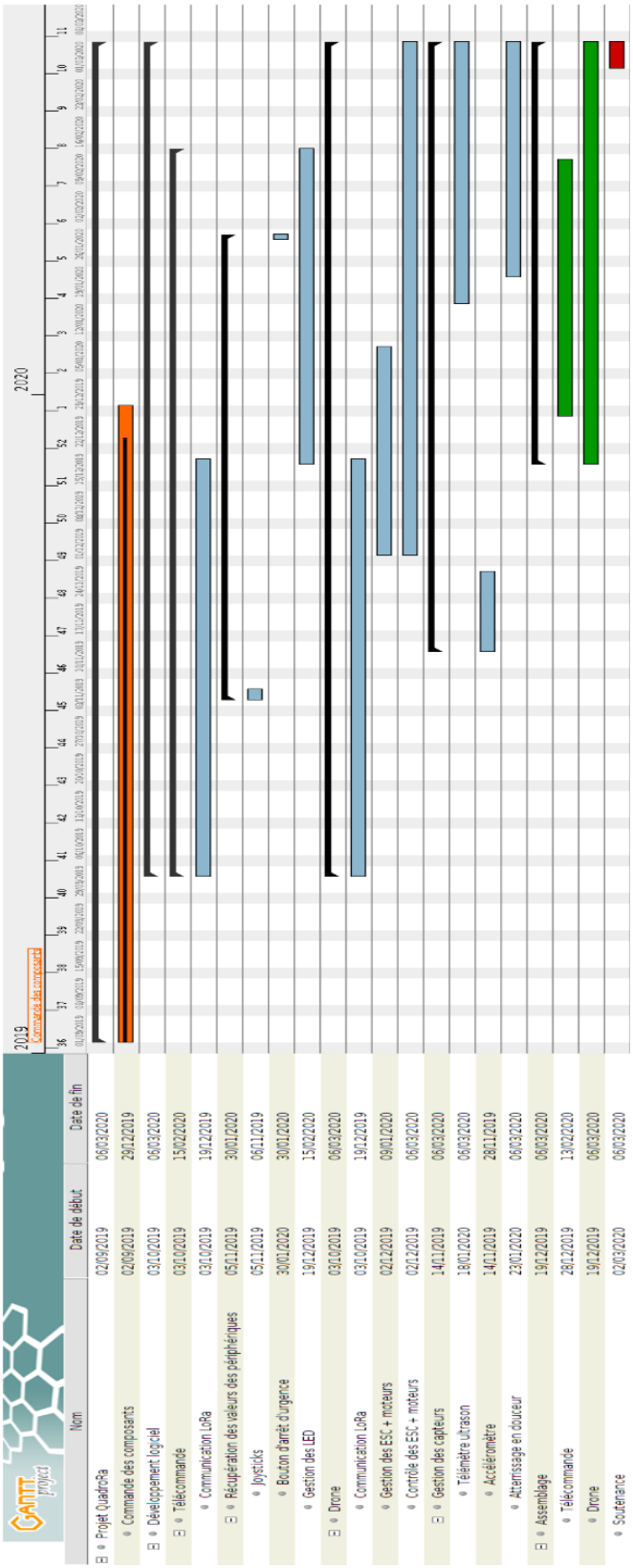
- Raspberry Pi : mini ordinateur disposant environ de la puissance de calcul d’un smartphone.
- ESP32 : micro-contrôleur.
- ADXL345: accéléromètre
- ESC (Electronic Speed Controller) : permet de contrôler un moteur.
- E32 : abréviation utilisée ici pour désigner un "E32-868T30S" qui est un module de communication LoRa.
- PCB (Printed Circuit Board, circuit imprimé en français) : plaquette servant à relier plusieurs composants électroniques entre eux.
- GPIO (General Purpose Input/Output) : ports d'entrées-sorties, petites broches permettant d'alimenter des composants ou de transmettre des données.
- LoRa : technologie permettant une communication longue portée et bas débit par ondes radio (868 MHz en Europe).
- UART : protocole de communication basique permettant d’échanger des données entre deux appareils uniquement.
- I²C : protocole de communication plus évolué permettant d’échanger des données entre deux appareils ou plus.
- PWM (Pulse Width Modulation, ou, en français, Modulation par Largeur d'Implusion)
- WiringPi : bibliothèque en C permettant l'utilisation des ports GPIO d'un micro-contrôleur http://wiringpi.com/.

Sources

La libraire WiringPi : http://wiringpi.com/

Documents Annexes

Diagramme de Gantt final



Cahier des charges
https://github.com/MicrowavedCat/LoRaDrone/blob/master/Cahier_des_charges.pdf

Guide de l'utilisateur
https://github.com/MicrowavedCat/LoRaDrone/blob/master/Guide_de_l_utilisateur.pdf