Rapport du projet d'IMA4

P17 Drone autonome

Parrot AR.Drone

When video games become reality

Encadré par M.REDON Xavier

Réalisé par COLAUTTI Kevin
LEFORT Benjamin

Polytech Lille Département Informatique Microélectronique Automatique

Remerciements

Nous tenons d'abord à exprimer notre reconnaissance et nos profonds remerciements à toute personne nous ayant aidé de près ou de loin à la réalisation de ce projet à ses différentes étapes.

Nous tenons à remercier plus particulièrement M REDON Xavier pour la confiance qui nous a accordée, ainsi que sa disponibilité et le temps qu'il nous a consacré tout au long du projet.

Sommaire

1 Introduction

Les drones sont déjà en phase de test pour la livraison de colis. Pour cela, il leur faut être autonome d'un point A à un point B en s'adaptant à leur environnement. Nous avons pour but d'étudier un tel système et de le rendre autonome à l'aide d'un système embarqué.

A terme, un drone autonome permet de nombreuses applications telles que la surveillance de lieux publics, ou hautement sécurisés. L'observation des animaux sauvages dans leur état naturel sans interférer leur environnement. Ou encore la livraison de colis de façon totalement autonome et écologique.

2 Présentation du projet

2.1 Système étudié

Nous avons à disposition des drones de la marque Parrot. Ils sont actuellement télécommandés uniquement via leur application et tant qu'ils sont à porté d'un point d'accès Wifi.

Nous allons donc l'équiper d'un système embarqué de type pcDuino pour le rendre autonome et pouvoir naviguer hors de porté du Wifi.

Pour cela, nous devons alimenter le pcDuino grâce à l'alimentation déjà présente sur le drone, à savoir la batterie.

De plus, un système avec deux interfaces Wifi doit ensuite être étudié, une première pour la connexion pcDuino/drone et la seconde pour la communication avec le pcDuino.

Enfin, on implante deux modes de fonctionnement : un premier permettant de contrôler le drone avec une interface Web simple et efficace qui est accessible via tous supports dotés du WIFI, un second permettant de spécifier un trajet à plus longue distance grâce au GPS ajouté au pcDuino.

2.2 Objectifs du projet

Le but est d'équiper un drone de type AR Drone de chez Parrot pour lui faire effectuer une tâche de façon autonome.

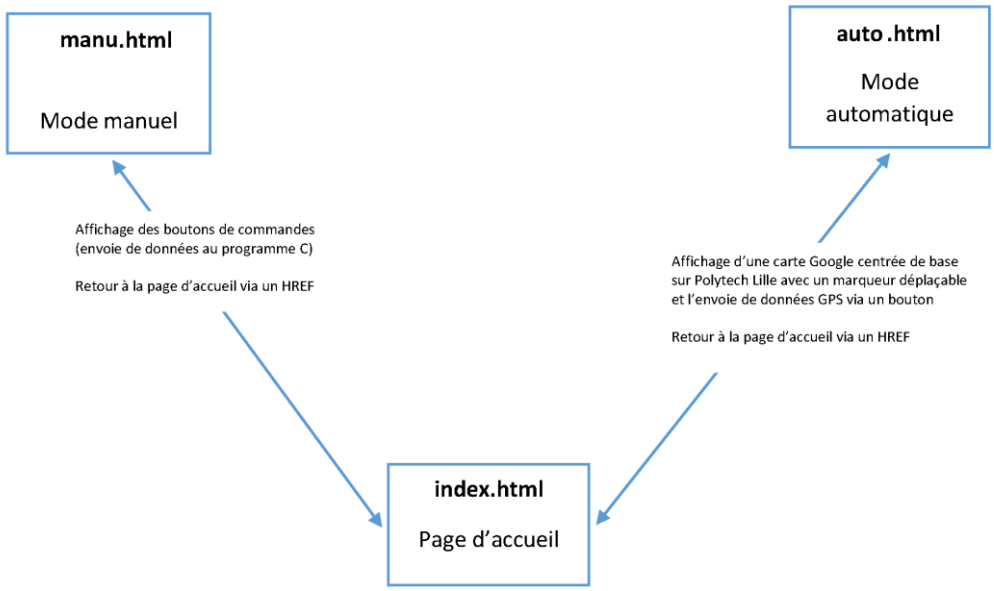
Nous devons donc étudier une interface web permettant d'envoyer les commandes du drone ou des coordonnées qui seront choisies par l'utilisateur.

De plus, il faut configurer le pcDuino à notre application et à l'utilisation du GPS.

Enfin, il faut étudier le système embarqué dans son ensemble. A savoir, son système d'alimentation, son autonomie et sa robustesse.

3 Interface web

Schéma de structure du site :



3.1 Index

L'index qui est donc la page d'accueil de l'interface homme/machine permet de choisir entre les deux modes de fonctionnement :

- Manuel
- Automatique

Nous avons une courte description du projet avec un lien vers le wiki pour de plus amples informations si nécessaires à l'aide du bouton « Plus d'info ». Les différents liens sont repris dans un menu en haut à droite de la page. (cf Annexe 1.1)

3.2 Mode manuel

Le mode manuel consiste à envoyer une commande à la fois à l'aide d'un bouton. (cf Annexe 1.2.1)

Pour cela, chaque bouton permet d'exécuter la fonction JavaScript « envoyer() » à l'aide de « onclick ».

Pour une commande plus intuitive du drone, nous avons intégré les évènements clavier. C'est la fonction « traitement() » qui permet de traiter quel évènement a eu lieu au clavier et donc exécute la fonction « envoyer() » correspondante.

Cette fonction « envoyer() » permet l'envoie d'une donnée à un cgi-bin. Elle prend en paramètre une lettre, celle-ci à l'aide d'un « post » en Ajax est transmise au cgi-bin. (cf Annexe 1.2.2)

Le cgi-bin permet donc de commander l'AR Drone en lui envoyant des paquets UDP (User Datagram Protocol). Il envoie donc des commandes AT (commandes de Hayes) qui sont listées dans le guide du développeur (SDK) édité par Parrot.

L'UDP est un protocole de télécommunication d'internet. Il ne propose aucune phase de connexion préalable à l'envoie des données. Il est donc très simple, mais aussi peu fiable. Cependant, dans notre application le chemin client/serveur (pcDuino/drone) est direct. Il n'est donc pas possible de « perdre » des paquets.

Toujours à l'aide du SDK, nous savons que l'adresse IP du drone sera 192.168.1.1 sur le port 5556.

Aussi, nous avons repris les différentes commentes AT que l'on termine par un « \r » qui symbolise le retour chariot. En effet, ce caractère va permettre au drone de comprendre que c'est la fin de la commande AT.

Nous avons également définie la taille maximale du paquet UDP à 256 caractères. En sachant que le maximum est de 1024.

Nous avons donc définie cela dès le début.

Ensuite, nous réalisons un simple client UDP à l'aide de socket qui utilise cette adresse IP sur ce port définie précédemment. Et grâce à la fonction « envoyer_AT() » nous envoyons le paquet UDP à l'aide d'un simple « sendto ». Nous avons ajouté un délai à la fin de cette fonction pour permettre d'avoir un petit temps d'attente entre deux commandes AT.

Cette fonction est appelée dans les différents cas d'un switch case qui différencie les commandes AT disponibles. Le switch case se base sur la lettre qui a été envoyée depuis la page web et qui est récupérée grâce à un « cgiGetValue() ». (cf Annexe 1.2.3)

3.3 Mode automatique

Le mode automatique consiste à choisir une adresse ou de déplacer le marqueur sur la carte GoogleMaps à l'endroit désiré. Les coordonnées s'affichent sur le site et à l'aide du bouton envoyer, l'utilisateur confirme le lieu de destination du drone. De base la carte est centrée sur Polytech Lille. (cf Annexe 1.3.1)

Sur le principe, ce mode ne diffère pas beaucoup du mode manuel. En effet, on récupère juste deux données au lieu d'une. Nous avons la latitude et la longitude à envoyer au cgi-bin, nous avons donc deux données dans la variable « parametres ».

Par contre, nous avons l'intégration d'une GoogleMaps au site. Pour cela, nous avons la fonction « load » qui permet de l'initialiser et d'avoir un marqueur déplaçable (draggable : true).

La fonction « showAddress » permet à l'utilisateur de choisir une adresse à sa convenance. Un formulaire lui permet de rentrer cette adresse qui est ensuite transmise à cette fonction grâce au bouton « Cherchez ! ». Le marqueur se déplace en conséquence.

A l'aide du bouton « envoyer », nous transmettons les coordonnées qui sont récupérées de cette carte par : document.getElementById("").innerHTML; (cf Annexe 1.3.2)

Cette fois le cgi-bin va écrire les deux données dans un fichier car il lui est impossible de fonctionner durant le déplacement du drone du point A au point B.

Nous avons un programme .c sensiblement identique au cgi-bin du mode manuel pour la partie client UDP (cf Annexe 1.3.3). Ce programme attend que le fichier avec les coordonnées finales soit non vide avant de lancer le traitement GPS.

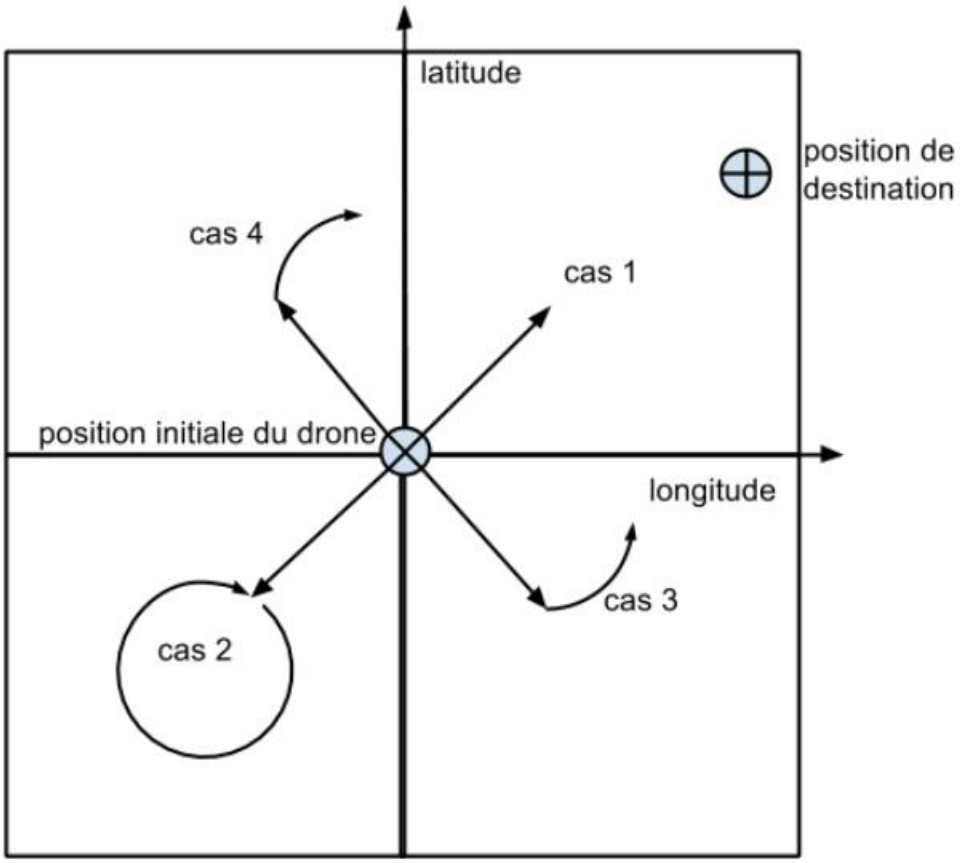
Pour ce traitement, nous avons deux petites fonctions à ajouter :

- La fonction « comparaison() » qui permet d'obtenir la valeur absolue de la soustraction de deux coordonnées.
- La fonction « test_arrivee() » qui est une fonction booléenne qui utilise la fonction précédente et qui renvoie « true » si nous sommes aux coordonnées finales à epsilon près (définie en variable globale) et renvoie « false » dans le cas contraire.

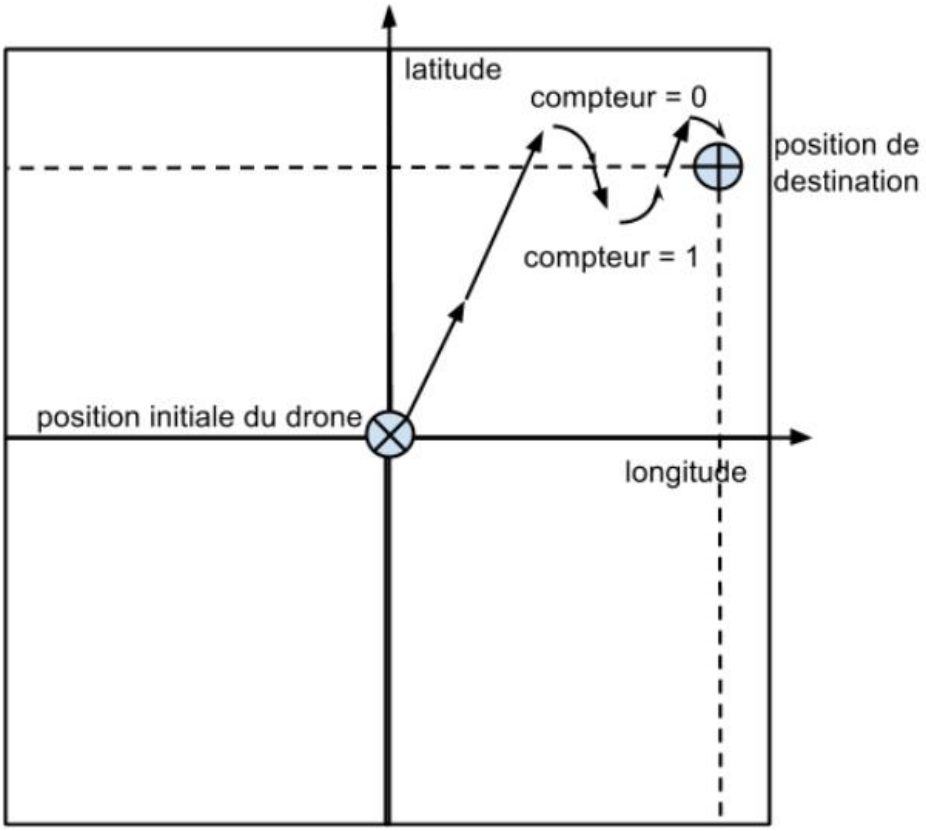
Nous n'avons plus le switch case mais on peut découper le « main() » en deux « tests » :

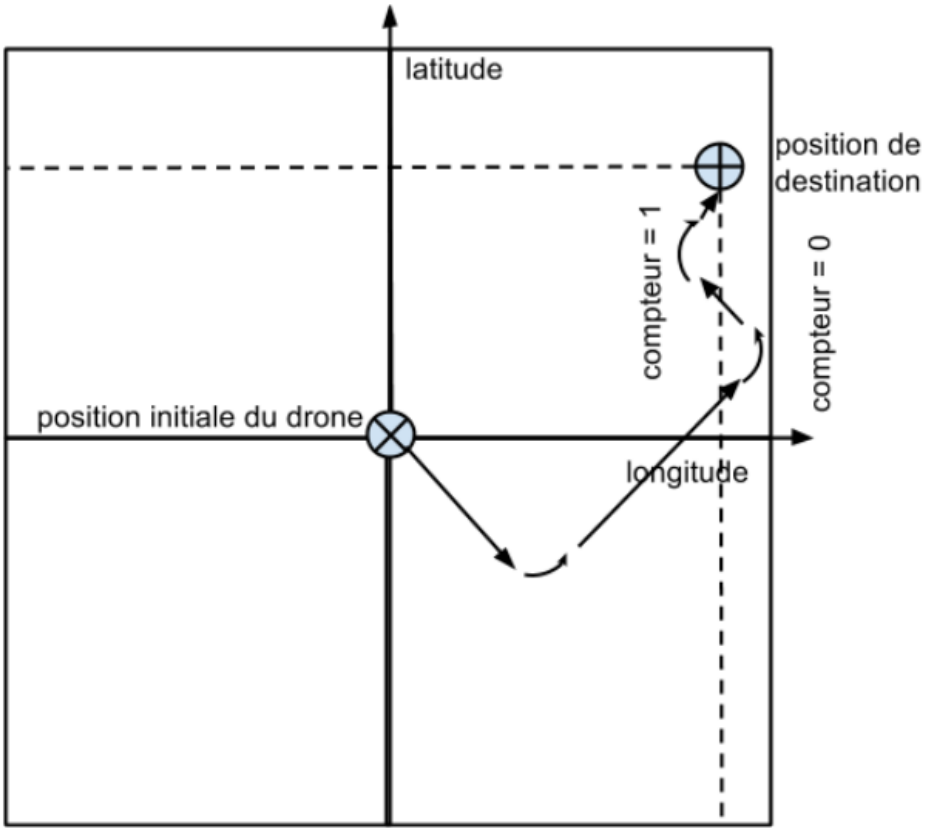
- Le premier consiste à connaître l'orientation du drone. En effet, on ne peut pas connaître à l'avance si le drone est bien dans la direction de la destination finale. Pour cela, nous pouvons imaginer que le drone se situe dans un quadrillage à uniquement quatre carreaux.
  Nous commençons par le faire décoller puis avancer durant quelques secondes tout droit. Ensuite, nous avons quatre cas :
  - Cas 1 : Si la distance entre la latitude à l'instant t et la latitude de destination est inférieure à la distance entre la latitude à l'origine et la latitude de destination et qu'il en est de même pour la longitude alors nous sommes dans le bon quart et donc bien orienté.
  - Cas 2 : Si ces deux tests sont faux, nous sommes dans la direction opposée, nous demandons donc au drone de faire un 180° en mode stationnaire.
  - Cas 3 : S'il n'y a que la distance entre les différentes latitudes qui est fausse, alors il fait une rotation de 90° vers la gauche en mode stationnaire.
  - Cas 4 : S'il n'y a que la distance entre les différentes longitudes qui est fausse, alors il fait une rotation de 90° vers la droite en mode stationnaire.

Ce test reste approximatif mais il permet tout de même d'orienter le drone dans la bonne direction, les figures ci-dessous illustrent les quatre cas :



- Le second test permet d'atteindre la destination finale. Nous entrons dans une boucle « while » tant que la fonction « test_arrivee() » renvoie « false ».
  Dans cette boucle, on demande au drone d'avancer tant que nous sommes dans le Cas 1. Sauf que l'on compare désormais la coordonnée à l'instant t0 et t1 par rapport à la coordonnée finale.
  Ensuite, il faut imaginer qu'il y a des perpendiculaires aux axes qui passent toutes les deux aux coordonnées finales. Lorsque le drone franchit l'une des deux droites, il va la longer jusqu'à atteindre sa destination.
  Pour cela, on commence par vérifier laquelle vient d'être franchie pour savoir si l'on doit tourner à droite ou à gauche. Puis grâce à un compteur binaire, le drone va tourner à droite ou à gauche en fonction de sa position par rapport à cette droite tout en avançant entre chaque rotation si elle a lieu. Ce schéma illustre les deux cas :





Lorsque le test d'arrivée est vérifié, on atterrit.

4 pcDuino et GPS

4.1 pcDuino

Les pcDuino sont des minis PC créés par LinkSprite, ils sont basés sur Linaro, un OS Linux fait pour les processeurs d'architecture ARM. Nous avons commencés par travailler principalement sur la connexion Wifi avec la version 1 de ces contrôleurs, cependant pour un gain de place, un pcDuino Nano v3 nous a été fourni.

Il est plus petit, plus puissant et ces GPIO sont placés de la même manière qu'un Arduino.

Afin de maximiser la puissance de fonctionnement du système d'exploitation, le pcDuino a été passé en Ubuntu 14.04 et les programmes gestion d'affichage (serveur X) ainsi que network manager ont été supprimés. Cependant, nous sommes rapidement arrivés aux limitations du système, une charge de travail faible peut rapidement le mettre à mal (liaison ssh, serveur apache et programme C)

4.2 Programme GPS

Le GPS fonctionnant au travers du port série, il a été nécessaire de créer un programme C pour gérer les trames et analyser la syntaxe de celle-ci pour récupérer les coordonnés GPS sous la bonne forme.

Le programme se décompose en cinq parties (cf Annexe 2) :

- La configuration des entrées/sorties pour utiliser le port série. Pour choisir si une broche fonctionne en entrée ou en sortie, il est nécessaire de configurer le fichier propre à chaque GPIO. Ici ce sont les GPIO 0 et 1 qui sont utilisés en mode 3.

- L'initialisation du port série. Dans cette fonction la configuration actuelle est enregistrée pour être restaurée à la fermeture du port. Les nouveaux paramétres sont également appliqués.

- La mise en forme des données. Le GPS fournissant quatre trames différentes, il est nécessaire de faire une analyse syntaxique de celle-ci. Cette analyse est faite à l'aide d'un parser provenant de la bibliothèque pour Arduino de notre GPS.
  Les coordonnées récupérées de ces trames sont ensuite remises en forme pour correspondre à une norme que nous utilisons dans tous les programmes. Ces données sont ensuite mises à disposition dans deux fichiers, un premier qui indique la position de départ et un second qui est mis à jour avec la position actuelle. Ces fichiers sont visibles dans le dossier tmp.

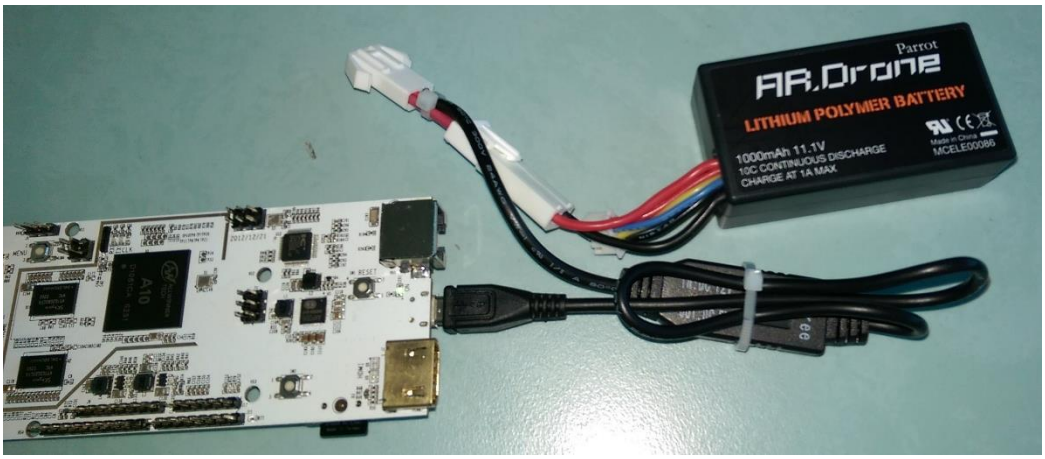
- Une sous-routine gérant les signaux de fin de processus. Celle-ci permet de récupérer le signal terminal du processus pour fermer correctement le port série et éviter des soucis lors de la réutilisation du programme.

- Le main. Il permet d'appeler les différentes fonctions et c'est lui qui récupère les trames provenant du port série.

5 Le système embarqué et ses solutions

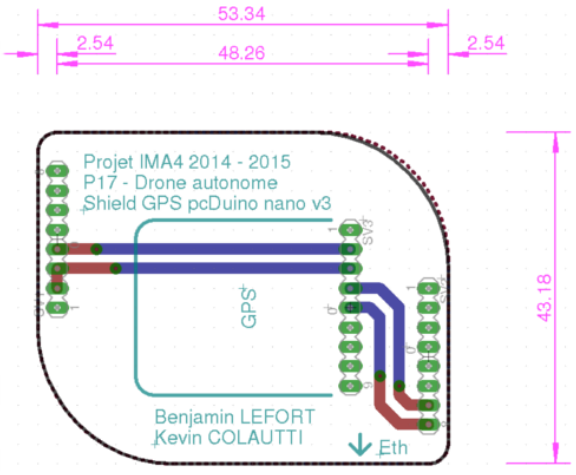
5.1 Alimentation

Afin de pouvoir alimenter le pcDuino à l'aide de la batterie du drone, nous avons acheté des connecteurs adaptés à ceux de la batterie et un convertisseur12V/5V avec un embout adapté au pcDuino. De l'autre nous l'avons dénudé afin de mettre le connecteur de batterie. Nous avons réduit également la longueur du fil du convertisseur afin de garder de la place et surtout du poids, tout en gardant une marge de sécurité et de modularité.



5.2 Shield GPS

Le drone devant se localiser, il a été nécessaire de créer un shield pour simplifier l'utilisation du module GPS. Un premier a été réalisé à l'aide de plaque de développement et de fils pour réaliser les connexions, puis un second a été réalisé sur PCB à l'aide du service électronique de l'école. L'une des particularités du pcDuino nano v3 est d'avoir un brochage de GPIO identique à un Arduino, ce qui permet d'utiliser différents supports ou bien d'utiliser celui créer sur un Arduino.



5.3 Coque

Pour assurer un vol en extérieur, il est primordial de pouvoir mettre le système embarqué sur le drone et de protéger l'ensemble pendant le vol. Pour cela, nous nous sommes servis de la coque extérieure comme patron.

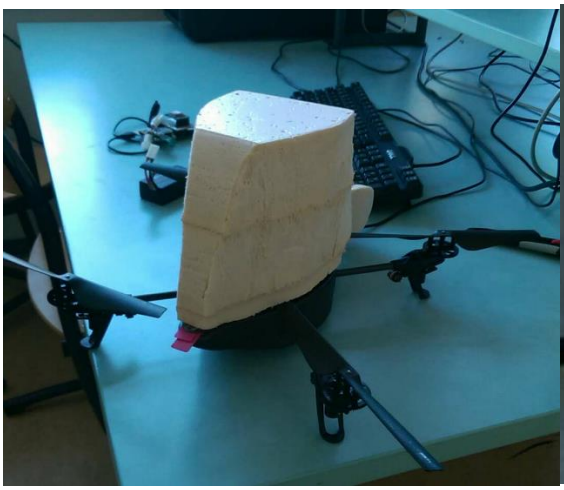
Nous avons vérifié si le système pouvait passer à plat sur la batterie malheureusement, les moteurs sont trop proches de la base et le système dépasse légèrement de celle-ci. Nous sommes contraints de mettre le pcDuino à la verticale.

Nous sommes allés au FabLab de Polytech Lille où nous avons pu utiliser des plaques d'isolation. Celles-ci sont très légères et très résistantes. Nous en avons collées trois ensemble grâce à du double face tissé. Grâce au menuisier de l'école nous avons pu obtenir le contour de toute la coque.

Les étapes suivantes de découpe se sont faites à l'aide d'une scie pour les gros œuvres, puis au papier à poncer.

Enfin, pour accueillir la caméra et le système embarqué sur le drone, nous avons fini le travail à la Dremel.

Voici le résultat :




5.4 Antennes Wifi

Deux antennes Wifi sont nécessaires dans ce projet, une première qui permet la liaison avec le drone et la seconde qui se connecte à un réseau autre. Il est également possible d'utiliser le pcDuino comme hotspot, mais cette solution n'a pas été retenue dû à la non-nécessité d'avoir une liaison Wifi quand le drone vole en autonomie.

Attention cependant à la provenance des clefs Wifi, à cause d'un problème logiciel, il a été nécessaire d'acheter des clefs officielles.

6 Limites hardware et software

6.1 Hardware

L'autonomie du drone n'est déjà pas très élevée, nous avions peur que le fait d'ajouter un système embarqué puisse la réduire fortement. Nous avons réalisé des mesures afin de vérifier cela.

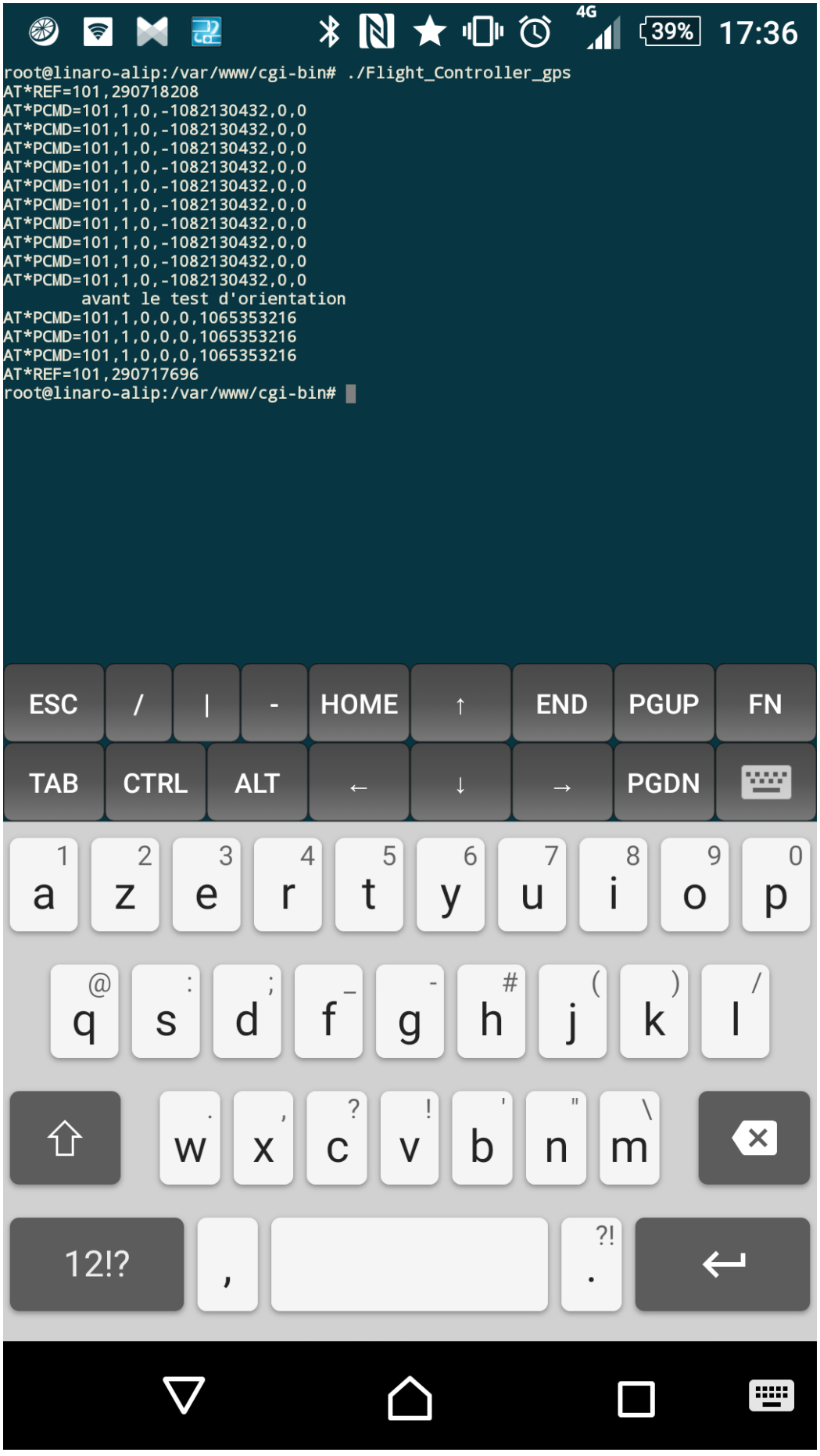
Point de mesure	Convertisseur 12V/5V	pcDuino	GPS	WiFi (x1)	Drone
Documentation	???	700mA	20mA	???	???
"Repos"	16,3mA	110mA	10 - 15mA	13mA	220 mA

On se rend compte que l'autonomie va diminuer mais raisonnablement. De plus, nous avons supprimé tous les services inutiles dans le pcDuino tel que le moteur graphique qui consomme beaucoup sans être utilisé.

Nous avons ensuite réalisé des tests de vol avec la coque que nous avions fabriqué. En effet, celle-ci est nettement plus grande et elle aurait pu déséquilibrer le drone ou être trop lourde. Avec une certaine fierté, le drone vole parfaitement avec la coque. Il ne tangue pas et réagit toujours aussi bien à toutes les commandes envoyées en manuel.

Notre déception fut lors du test en mode automatique. En effet, nous pensions avoir un problème logiciel car le drone ne répondait plus ou très mal. De plus, il finissait au ras du sol alors qu'il y a une sécurité avec un minimum d'un mètre d'altitude au décollage. Ne comprenant pas, nous avons réessayé avec le mode manuel. Malheureusement, nous avions le même résultat, idem avec l'application officielle. Nous en concluons que les moteurs ne sont pas assez puissants pour supporter le pcDuino en plus. D'où la baisse progressive du drone après le décollage.

Afin de vérifier le programme du GPS, nous nous sommes déplacés avec le pcDuino sur nous. On remarque effectivement que les commandes sont bien envoyées et de façon correcte, voici une capture d'écran du Smartphone durant l'exécution du programme :


39%
17:36
root@linaro-alip:/var/www/cgi-bin# ./Flight_Controller_gps
AT*REF=101,290718208
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
AT*PCMD=101,1,0,-1082130432,0,0
avant le test d'orientation
AT*PCMD=101,1,0,0,0,1065353216
AT*PCMD=101,1,0,0,0,1065353216
AT*PCMD=101,1,0,0,0,1065353216
AT*REF=101,290717696
root@linaro-alip:/var/www/cgi-bin#
ESC
/
|
-
HOME
↑
END
PGUP
FN
TAB
CTRL
ALT
←
↓
→
PGDN
12!?

6.2 Software

Nous rencontrons également quelques problèmes du côté logiciel. En effet, le pcDuino n'est pas assez puissant pour supporter le côté graphique et ergonomique du site. En effet, si l'on essaye d'y accéder le pcDuino ne répond plus.

Nous sommes donc contraints de revenir à la phase primaire de l'interface web. (cf Annexe 3)

7 Conclusion

Ce projet est pour nous une très bonne expérience personnelle et pédagogique. En effet, nous avons pu construire un projet en autonomie. Nous nous sommes fixés notre propre cahier des charges à partir de résultats souhaités, ainsi qu'un planning à respecter. Cette situation simule très bien une future expérience en entreprise.

Aux yeux du cahier des charges initial, nous avons plutôt bien réussi notre travail, mais vis à vis de nos ambitions nous sommes un peu en deçà des objectifs prévus. En effet, nous avons rencontré quelques difficultés, en partie dues à la partie mécanique du drone. Nous n'avons donc pas un système autonome qui fonctionne sur le drone.

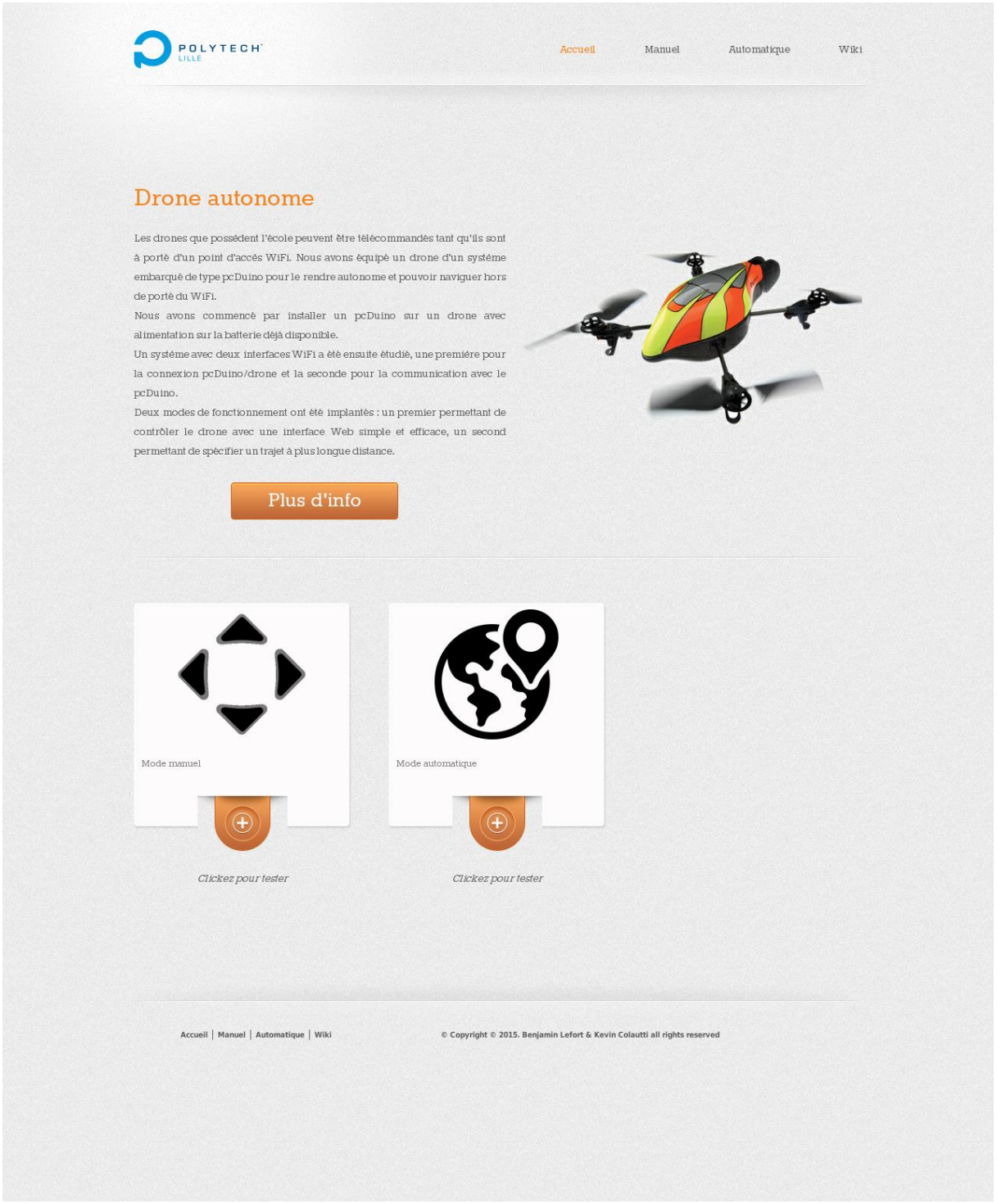
Cela a été finalement très enrichissant, puisque nous avons pu perfectionner de nombreux points de cours qui sont la base et de les approfondir avec un peu de recherche.

Nous avons aussi rencontré beaucoup de difficultés avec le temps. Le drone ne réagissait jamais de la même manière au décollage. Il fallait donc à chaque fois réaliser des allers-retours dans des endroits plus ouverts. Aussi, le pcDuino a des performances très réduites lorsqu'il était sur batterie. Il nous fallait parfois vingt minutes avant de lancer le programme. Nous ne sommes pas toujours maître du temps à notre disposition, ce qui compte c'est d'en être conscient et d'agir en conséquence.

D'autres points comme la récupération vidéo, la gestion de l'altitude et d'autres paramètres intéressants pour rendre le drone totalement autonome à son environnement peuvent approfondir notre projet mais pour cela il faudrait partir sur un autre drone qui supporte un peu plus de poids que son poids initial.

8 Annexes

Annexe 1.1



Code de l’index: index.html

```
<!DOCTYPE html>
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>
<title>Drone autonome</title>
<linkrel="stylesheet"href="css/style.css"type="text/css"/>
</head>
<body>
<divclass="page">
<divclass="header">
<ahref="index.html"id="logo"><imgsrc="images/polytech.png"height="49"width="166"alt=""/></a>
<ul>
<liclass="selected"><ahref="index.html">Accueil</a></li>
<li><ahref="manu.html">Manuel</a></li>
<li><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
</div>
<divclass="body">
<divid="featured">
<h3>Drone autonome</h3>
<p>Les drones que possèdent l'école peuvent être télécommandés tant qu'ils sont à porté d'un point d'accès WiFi. Nous avons équipé un drone d'un système embarqué de type pcDuino pour le rendre autonome et pouvoir naviguer hors de porté du WiFi.</p>

<p>Nous avons commencé par installer un pcDuino sur un drone avec alimentation sur la batterie déjà disponible.</p>

<p>Un système avec deux interfaces WiFi a été ensuite étudié, une première pour la connexion pcDuino/drone et la seconde pour la communication avec le pcDuino.</p>

<p>Deux modes de fonctionnement ont été implantés : un premier permettant de contrôler le drone avec une interface Web simple et efficace, un second permettant de spécifier un trajet à plus longue distance.</p>
<inputtype="button"value="Plus d'info"onClick="parent.location='http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome'"/>
</div>
<center>
<ulclass="blog">
<li>
<div>
<ahref="manu.html"><imgsrc="images/manu.png"height="168"width="168"alt=""/></a>
<p>Mode manuel</p>
<ahref="manu.html">Clickez pour tester</a>
</div>
</li>
<li>
<div>
<ahref="auto.html"><imgsrc="images/world.png"height="168"width="168"alt=""/></a>
<p>Mode automatique</p>
<ahref="auto.html">Clickez pour tester</a>
</div>
</li>
</ul>
</center>
</div>
<divclass="footer">
<ul>
<li><ahref="index.html">Accueil</a></li>
```

```
<li><ahref="manu.html">Manuel</a></li>
<li><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
<p>&#169; Copyright &#169; 2015. Benjamin Lefort&Kevin Colautti all rights reserved</p>
</div>
</div>
</div>
</body>
</html>
```

Annexe 1.2.1

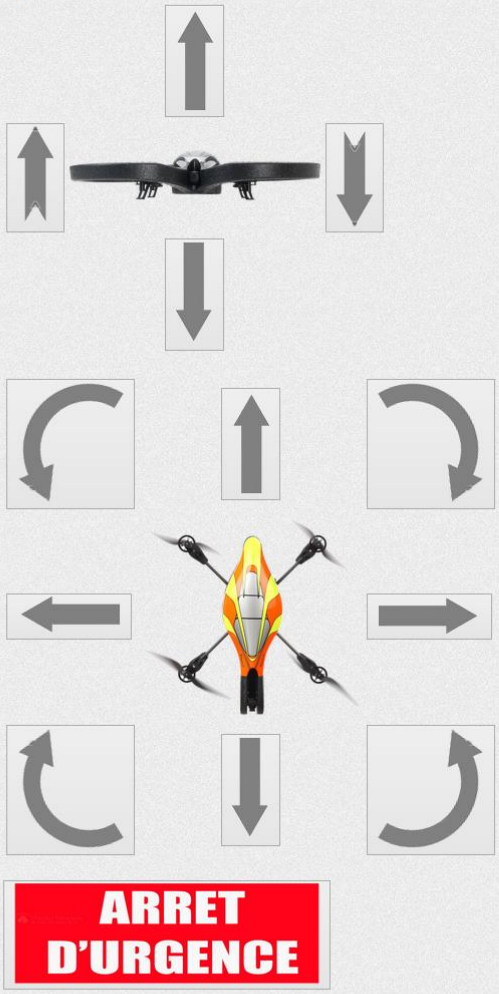
POLYTECH LILLE

Accueil Manuel Automatique Wiki

Mode manuel

Le mode manuel permet le contrôle du drone a vu, de part son fonctionnement il est utilisable sur n'importe quel système d'exploitation ou support. De plus, il ne nécessite pas la présence on-board du pcDuino.

Déplacement :



Raccourci clavier :

v : Atterissage
c : Decollage
z : Monter
s : Descendre
d : Rotation droite
q : Rotation gauche
o : Avancer
l : Reculer
m : Virer à droite
k : Virer à gauche
u : Urgence

Accueil | Manuel | Automatique | Wiki

Annexe 1.2.2

Code du mode manuel : manu.html

```
<!DOCTYPE html>
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>
<title>Drone autonome</title>
<linkrel="stylesheet"href="css/style.css"type="text/css"/>

<scripttype="text/javascript"src="jquery.js"></script>
<scriptlanguage="javascript">
functionenvoyer(lettre){
var mot=lettre;
varparametres={donnees: mot};
$.ajax({url:'/cgi-bin/Flight_Controller', data: parametres, type: 'post'});
    }

functiontraitement(evenement){
if(evenement.which == 86){envoyer('v');}
if(evenement.which == 67){envoyer('c');}
if(evenement.which == 90){envoyer('z');}
if(evenement.which == 83){envoyer('s');}
if(evenement.which == 68){envoyer('d');}
if(evenement.which == 81){envoyer('q');}
if(evenement.which == 79){envoyer('o');}
if(evenement.which == 76){envoyer('l');}
if(evenement.which == 77){envoyer('m');}
if(evenement.which == 75){envoyer('k');}
if(evenement.which == 85){envoyer('u');}
    }

    $(function(){
      $(document).keydown(traitement);
    });
</script>


</head>
<body>
<divclass="page">
<divclass="header">
<ahref="index.html"id="logo"><imgsrc="images/polytech.png"height="49"width="166"alt=""/></a>
<ul>
<li><ahref="index.html">Accueil</a></li>
<liclass="selected"><ahref="manu.html">Manuel</a></li>
<li><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
</div>
<divclass="body">


<h3>Mode manuel</h3>
<p>Le mode manuel permet le contrôle du drone a vu, de part son fonctionnement il est utilisable sur n'importe quel système d'exploitation ou support. De plus, il ne nécessite pas la présence on-board du pcDuino.</p>

<fieldset>
<h3>Déplacement :</h3>
```

```
<table>
<tr>
<td></td>
<td><center><buttononclick="envoyer('z')"title="Monter"><imgsrc="images/arrow1.png"alt="Monter"/></button></center></td>
<td></td>
</tr>
<tr>
<td><buttononclick="envoyer('c')"title="Decollage"><imgsrc="images/arrow2.png"alt="Decollage"style="image-orientation: 90deg;"/></button></td>
<td><imgsrc="images/cont1.png"height="50"alt="Ctrl hauteur"/></td>
<td><buttononclick="envoyer('v')"title="Atterissage"><imgsrc="images/arrow2.png"alt="Atterissage"style="image-orientation: 270deg;"/></button></td>
</tr>
<tr>
<td></td>
<td><center><buttononclick="envoyer('s')"title="Descendre"><imgsrc="images/arrow1.png"alt="Descendre"style="image-orientation: 180deg;"/></button></center></td>
<td></td>
</tr>

</table>
<br>
<table>
<tr>
<td><buttononclick="envoyer('q')"title="Rotation gauche"><imgsrc="images/arrow4.png"alt="Rotation gauche"style="image-orientation: 180deg;"/></button></td>
<td><center><buttononclick="envoyer('o')"title="Avancer"><imgsrc="images/arrow1.png"alt="Avancer"/></button></center></td>
<td><buttononclick="envoyer('d')"title="Rotation droite"><imgsrc="images/arrow3.png"alt="Rotation droite"/></button></td>
</tr>
<tr>
<td><buttononclick="envoyer('k')"title="Virer à gauche"><imgsrc="images/arrow1.png"alt="Virer à gauche"style="image-orientation: 270deg;"/></button></td>
<td><imgsrc="images/cont2.png"height="200"alt="Ctrl mouvement"/></td>
<td><buttononclick="envoyer('m')"title="Virer à droite"><imgsrc="images/arrow1.png"alt="Virer à droite"style="image-orientation: 90deg;"/></button></td>
</tr>
<tr>
<td><buttononclick="envoyer('d')"title="Rotation droite"><imgsrc="images/arrow3.png"alt="Rotation droite"style="image-orientation: 180deg;"/></button></td>
<td><center><buttononclick="envoyer('l')"title="Reculer"><imgsrc="images/arrow1.png"alt="Reculer"style="image-orientation: 180deg;"/></button></center></td>
<td><buttononclick="envoyer('q')"title="Rotation gauche"><imgsrc="images/arrow4.png"alt="Rotation gauche"/></button></td>
</tr>
</table>
<br>
<buttononclick="envoyer('u')"title="Arreturgence"><imgsrc="images/urg.jpg"height="100"alt="Arreturgence"/></button>
</fieldset>
<br>
<fieldset>
<h3>Raccourci clavier :</h3>
<p>v : Atterissage<br>
c : Decollage<br>
z : Monter<br>
s : Descendre<br>
d : Rotation droite<br>
```

```
q : Rotation gauche<br>
o : Avancer<br>
l : Reculer<br>
m : Virer à droite<br>
k : Virer à gauche<br>
u :Urgence</p>
</fieldset>


</div>
<divclass="footer">
<ul>
<li><ahref="index.html">Accueil</a></li>
<li><ahref="manu.html">Manuel</a></li>
<li><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
<p>&#169; Copyright &#169; 2015. Benjamin Lefort&Kevin Colautti all rights reserved</p>
</div>
</div>
</div>
</body>
</html>
```

Annexe 1.2.3

Code du cgi-bin : Flight_Controller.c

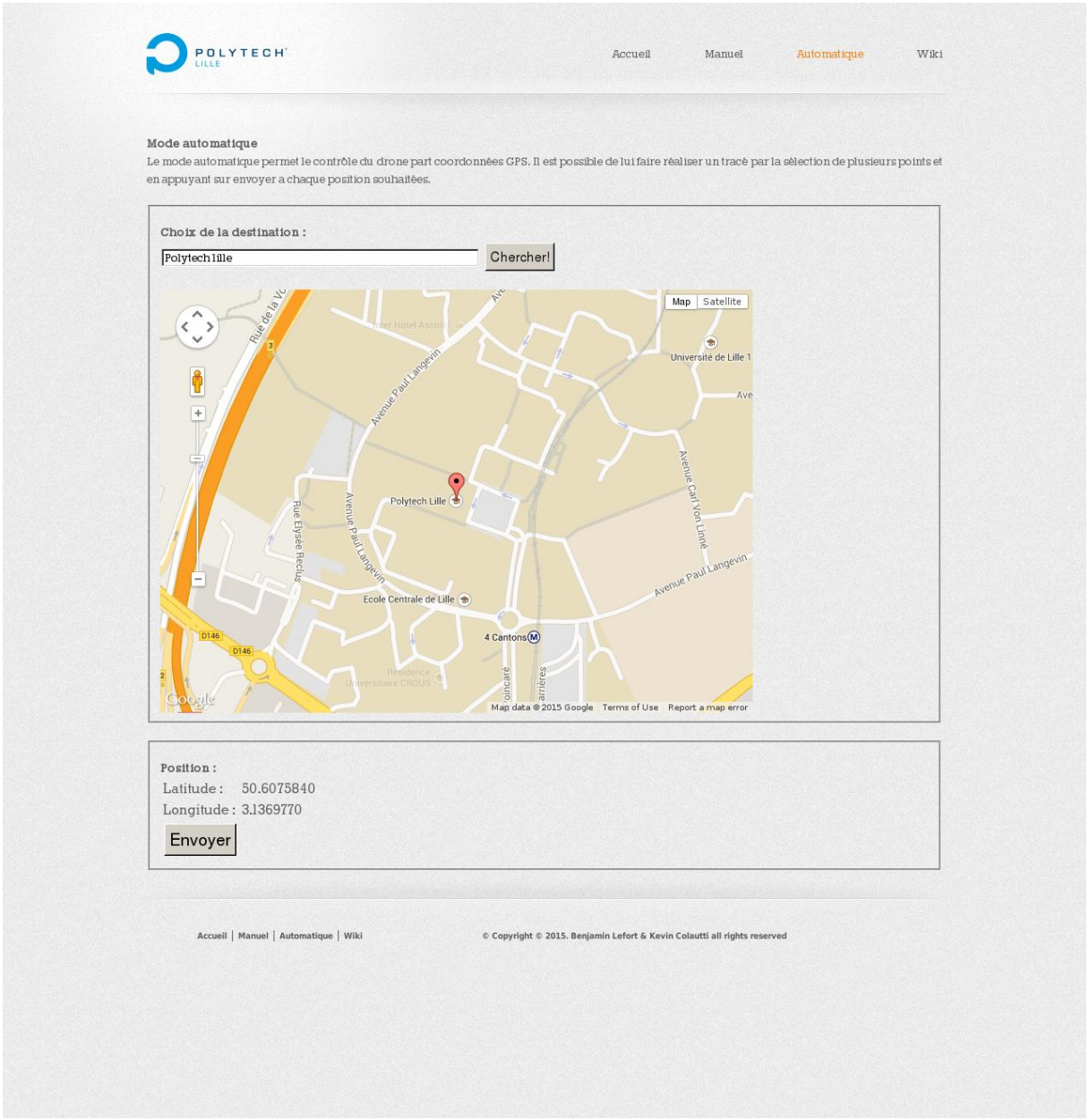
```
// Commande AR.DRONE

// BIBLIOTHÈQUES
//#include<SDL/SDL.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <cgi.h>

// ADRESSE IP AR.DRONE
#define ADRESSEIP "192.168.1.1"
#define PORT 5556
#define BUFLEN 256

// COMMANDES AT
#define COMMANDE_AT_DECOLLAGE    ",290718208\r" // c
#define COMMANDE_AT_ATTERISSAGE    ",290717696\r" // v
#define COMMANDE_AT_ARRET_URGENCE    ",290717952\r" // u
#define COMMANDE_AT_ANTI_ARRET_URGENCE    ",290717696\r" // u
#define COMMANDE_AT_AVANT    ",1,0,-1082130432,0,0\r" // o
#define COMMANDE_AT_ARRIERE  ",1,0,1065353216,0,0\r" // l
#define COMMANDE_AT_GAUCHE   ",1,-1082130432,0,0,0\r" // k
#define COMMANDE_AT_DROITE   ",1,1065353216,0,0,0\r" // m
#define COMMANDE_AT_HAUT ",1,0,0,1065353216,0\r" // z
#define COMMANDE_AT_BAS  ",1,0,0,-1082130432,0\r" // s
#define COMMANDE_AT_ROTATION_GAUCHE  ",1,0,0,0,-1082130432\r" // q
#define COMMANDE_AT_ROTATION_DROITE  ",1,0,0,0,1065353216\r" // d

#define DEBUG

// PROTOTYPES DES FONCTIONS
void err(char*s);
voidenvoyer_AT(char* commande,unsignedint compteur,char* argument);

// VARIABLES GLOBALES
structsockaddr_inserv_addr;
intsockfd,slen;

// FONCTION MAIN
int main(void)
{
slen=sizeof(serv_addr);
if((sockfd= socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1)
{
err("socket");
}
bzero(&serv_addr,sizeof(serv_addr));
serv_addr.sin_family= AF_INET;
serv_addr.sin_port=htons(PORT);
if(inet_aton(ADRESSEIP,&serv_addr.sin_addr)==0)
{
```

```
fprintf(stderr,"inet_aton() erreur\n");
exit(1);
}

intstatus=0;// Status = 0 : Le drone est au sol
int urgence =0;// Mode urgence désactivé
unsignedint compteur =0;// Initialisation du compteur

//GESTION RECEPTION CGI
s_cgi*cgi;
  cgi=cgiInit();
cgiHeader();
char*donnees=cgiGetValue(cgi,"donnees");

//CONTROLE A LA CONSOLE

//Affichage de l'interface
printf("\r\n\r\ndonnes=%s\r\n",donnees);
printf("\nBienvenue dans l'interface de contrôle de l'ARDrone\n\n");
printf("\tVoici les touches de commande :\n\n");
printf("\t\t- Pause : p\n\t\t- Atterissage : v\n\t\t- Decollage : c\n\t\t- Arret d'urgence : u\n\t\t- En avant : o\n\t\t- En arriere : l\n\t\t- Virer a gauche : k\n\t\t- Virer a droite : m\n\t\t- Prendre de l'altitude : z\n\t\t- Descendre en altitude : s\n\t\t- Pivoter sur la gauche : q\n\t\t- Pivoter sur la droite : d\n");

switch(*donnees)
{
case'v':
#ifdef DEBUG
printf("\nAtterissage !\n");
#endif
                compteur++;
envoyer_AT("AT*REF=",compteur,COMMANDE_AT_ATTERISSAGE);
break;

case'c':
#ifdef DEBUG
printf("\nDécollage !\n");
#endif
                compteur++;
envoyer_AT("AT*REF=",compteur,COMMANDE_AT_DECOLLAGE);
break;

case'u':
compteur++;
#ifdef DEBUG
printf("ARRET_URGENCE\n");
#endif
envoyer_AT("AT*REF=",compteur,COMMANDE_AT_ARRET_URGENCE);
break;

case'o':
compteur++;
#ifdef DEBUG
printf("AVANT\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_AVANT);
break;

case'l':
compteur++;
```

```
#ifdef DEBUG
printf("ARRIERE\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ARRIERE);
break;

case'k':
compteur++;
#ifdef DEBUG
printf("GAUCHE\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_GAUCHE);
break;

case'm':
compteur++;
#ifdef DEBUG
printf("DROITE\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_DROITE);
break;

case'z':
compteur++;
#ifdef DEBUG
printf("HAUT\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_HAUT);
break;

case's':
compteur++;
#ifdef DEBUG
printf("BAS\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_BAS);
break;

case'q':
compteur++;
#ifdef DEBUG
printf("ROTATION_GAUCHE\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_GAUCHE);
break;

case'd':
                compteur++;
#ifdef DEBUG
printf("ROTATION_DROITE\n");
#endif
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_DROITE);
break;
}

cgiFree(cgi);
//close(sockfd);
//return EXIT_SUCCESS;
}
```

```
// DÉFINITION DES FONCTIONS

voiderr(char*signal)
{
perror(signal);
//exit(1); // Décommentez pour quitter le programme en cas d'erreur
}

voidenvoyer_AT(char* commande,unsignedint compteur,char* argument)
{
charsequence_number[20];
charmessage[100];
sprintf(sequence_number,"%d",compteur);
strcpy(message,commande);
strcat(message,sequence_number);
strcat(message,argument);
#ifdef DEBUG
printf("%s\n",message);
#endif
if(sendto(sockfd, message, BUFLEN,0,(structsockaddr*)&serv_addr,slen)==-1)
{
err("sendto()");
}
usleep(50000);// Délai (approximatif) entre deux commandes AT
}
```

Annexe 1.3.1


POLYTECH LILLE
Accueil
Manuel
Automatique
Wiki
Mode automatique
Le mode automatique permet le contrôle du drone part coordonnées GPS. Il est possible de lui faire réaliser un tracé par la sélection de plusieurs points et en appuyant sur envoyer a chaque position souhaitées.
Choix de la destination :
Polytech lille
Chercher!
Map
Satellite
Université de Lille 1
Avenue Paul Langevin
Polytech Lille
Rue Elysée Reclus
Avenue Carl Von Linné
Ecole Centrale de Lille
4 Cantons
D146
Google
Map data ©2015 Google
Terms of Use
Report a map error
Position :
Latitude : 50.6075840
Longitude : 3.1369770
Envoyer
Accueil | Manuel | Automatique | Wiki
© Copyright © 2015. Benjamin Lefort & Kevin Colautti all rights reserved

Annexe 1.3.2

Code du mode auto : auto.html

```
<!DOCTYPE html>
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>
<title>Drone autonome - Mode automatique</title>
<linkrel="stylesheet"href="css/style.css"type="text/css"/>

<scriptsrc="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAgrj58PbXr2YriiRDqbnL1
RSqrCjdkglBijPNIIYrqkVvD1R4QxRl47Yh2D_0C1l5KXQJGrbkSDvXFA"
type="text/javascript">
</script>

<scripttype="text/javascript">

function load() {
if (GBrowserIsCompatible()) {
var map = new GMap2(document.getElementById("map"));
map.addControl(newGSmallMapControl());
map.addControl(newGMapTypeControl());
varcenter = newGLatLng(50.6075840, 3.1369770);
map.setCenter(center, 16);
geocoder = newGClientGeocoder();
var marker = newGMarker(center, {draggable: true});
map.addOverlay(marker);
document.getElementById("lat").innerHTML = center.lat().toFixed(7);
document.getElementById("lng").innerHTML = center.lng().toFixed(7);

GEvent.addListener(marker, "dragend", function() {
var point = marker.getPoint();
map.panTo(point);
document.getElementById("lat").innerHTML = point.lat().toFixed(7);
document.getElementById("lng").innerHTML = point.lng().toFixed(7);
        });

GEvent.addListener(map, "moveend", function() {
map.clearOverlays();
varcenter = map.getCenter();
var marker = newGMarker(center, {draggable: true});
map.addOverlay(marker);
document.getElementById("lat").innerHTML = center.lat().toFixed(7);
document.getElementById("lng").innerHTML = center.lng().toFixed(7);

GEvent.addListener(marker, "dragend", function() {
var point =marker.getPoint();
map.panTo(point);
document.getElementById("lat").innerHTML = point.lat().toFixed(7);
document.getElementById("lng").innerHTML = point.lng().toFixed(7);
          });
        });
      }
    }

functionshowAddress(address) {
var map = new GMap2(document.getElementById("map"));
map.addControl(newGSmallMapControl());
map.addControl(newGMapTypeControl());
```

```
if (geocoder) {
geocoder.getLatLng(address,function(point) {
if (!point) {
alert(address + " not found");}
else {
document.getElementById("lat").innerHTML = point.lat().toFixed(7);
document.getElementById("lng").innerHTML = point.lng().toFixed(7);
map.clearOverlays()
map.setCenter(point, 14);
var marker = newGMarker(point, {draggable: true});
map.addOverlay(marker);
GEvent.addListener(marker, "dragend", function() {
varpt = marker.getPoint();
map.panTo(pt);
document.getElementById("lat").innerHTML = pt.lat().toFixed(7);
document.getElementById("lng").innerHTML = pt.lng().toFixed(7);
            });

GEvent.addListener(map, "moveend", function() {
map.clearOverlays();
varcenter = map.getCenter();
var marker = newGMarker(center, {draggable: true});
map.addOverlay(marker);
document.getElementById("lat").innerHTML = center.lat().toFixed(7);
document.getElementById("lng").innerHTML = center.lng().toFixed(7);

GEvent.addListener(marker, "dragend", function() {
varpt = marker.getPoint();
map.panTo(pt);
document.getElementById("lat").innerHTML = pt.lat().toFixed(7);
document.getElementById("lng").innerHTML = pt.lng().toFixed(7);
              });
            });
          }
        });
       }
     }

</script>

<scripttype="text/javascript">
//<![CDATA[
//vargs_d=new Date,DoW=gs_d.getDay();gs_d.setDate(gs_d.getDate()-(DoW+6)%7+3);
//varms=gs_d.valueOf();gs_d.setMonth(0);gs_d.setDate(4);
//var gs_r=(Math.round((ms-gs_d.valueOf())/6048E5)+1)*gs_d.getFullYear();
//var gs_p = (("https:" == document.location.protocol) ? "https://" : "http://");
//document.write(unescape("%3Cscript src='" + gs_p + "s.gstat.orange.fr/lib/gs.js?"+gs_r+"'
type='text/javascript'%3E%3C/script%3E"));
////]]>
</script>

<scripttype="text/javascript"src="jquery.js"></script>
<scriptlanguage="javascript">
functionenvoyer()
    {
var latitude=document.getElementById("lat").innerHTML;
var longitude=document.getElementById("lng").innerHTML;
varparametres={donneesX: longitude, donneesY: latitude};
    $.ajax({url:'/cgi-bin/Flight_Controller_auto', data: parametres, type: 'post'});
}
```

```
</script>


</head>

<bodyonload="load()"onunload="GUnload()">
<divclass="page">
<divclass="header">
<ahref="index.html"id="logo"><imgsrc="images/polytech.png"height="49"width="166"alt=""/></a>
<ul>
<li><ahref="index.html">Accueil</a></li>
<li><ahref="manu.html">Manuel</a></li>
<liclass="selected"><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
</div>
<divclass="body">

<h3>Mode automatique</h3>
<p>Le mode automatique permet le contrôle du drone part coordonnées GPS. Il est possible de lui faire réaliser un tracé par la sélection de plusieurs points et en appuyant sur envoyer a chaque position souhaitées.</p>

<fieldset>
<h3>Choix de la destination :</h3>
<formaction="#"onsubmit="showAddress(this.address.value); return false">
<p>
<inputtype="text"size="60"name="address"value="Polytech lille"/>
<inputtype="submit"value="Chercher!"/>
</p>
</form>
<divalign="center"id="map"style="width: 700px; height: 500px"><br/></div>
</fieldset>
<br>
<fieldset>
<h3>Position :</h3>
<table>
<tr>
<td>Latitude : </td>
<tdid="lat"></td>

</tr>
<tr>
<td>Longitude : </td>
<tdid="lng"></td>
</tr>
<tr>
<td><buttononclick="envoyer()">Envoyer</button></td>
</tr>
</table>
</fieldset>
</div>
<divclass="footer">
<ul>
<li><ahref="index.html">Accueil</a></li>
<li><ahref="manu.html">Manuel</a></li>
<li><ahref="auto.html">Automatique</a></li>
<li><ahref="http://projets-imasc.plil.net/mediawiki/index.php?title=Drone_autonome">Wiki</a></li>
</ul>
<p>&#169; Copyright &#169; 2015. Benjamin Lefort&Kevin Colautti all rights reserved</p>
</div>
```

```
</div>
</div>
</body>
</html>
```

Annexe 1.3.3

Code du cgi-bin mode automatique : Flight_Controller_auto.c

```
#include <cgi.h>
#include <stdio.h>
#include <stdlib.h>

int main(void){
s_cgi*cgi;
cgi=cgiInit();
cgiHeader();
printf("\n\r\n\rtest\n\r");
char*donneesX=cgiGetValue(cgi,"donneesX");
char*donneesY=cgiGetValue(cgi,"donneesY");
printf("%s %s\n\r",donneesX,donneesY);
doublelng_desti=strtod(donneesX,NULL);
doublelat_desti=strtod(donneesY,NULL);
printf("%f %f\n\r",lng_desti,lat_desti);

FILE *fp;
fp=fopen("/var/www/cgi-bin/pos_final","a+");
if(fp==NULL){perror("pos_finale");exit(-1);}
else{
fprintf(fp,"%lf %lf\n",lat_desti,lng_desti);
fclose(fp);
}

cgiFree(cgi);
}
```

Code du programme gps : Flight_Controller_gps.c

```
// Commande AR.DRONE

// BIBLIOTHÈQUES
//#include<SDL/SDL.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>

// ADRESSE IP AR.DRONE
#define ADRESSEIP "192.168.1.1"
#define PORT 5556
#define BUFLEN 256

// COMMANDES AT
#define COMMANDE_AT_DECOLLAGE    ",290718208\r" // c
#define COMMANDE_AT_ATTERISSAGE    ",290717696\r" // v
#define COMMANDE_AT_ARRET_URGENCE    ",290717952\r" // u
#define COMMANDE_AT_ANTI_ARRET_URGENCE    ",290717696\r" // u
```

```
#define COMMANDE_AT_AVANT    ",1,0,-1082130432,0,0\r" // o
#define COMMANDE_AT_ARRIERE  ",1,0,1065353216,0,0\r" // l
#define COMMANDE_AT_GAUCHE   ",1,-1082130432,0,0,0\r" // k
#define COMMANDE_AT_DROITE   ",1,1065353216,0,0,0\r" // m
#define COMMANDE_AT_HAUT ",1,0,0,1065353216,0\r" // z
#define COMMANDE_AT_BAS  ",1,0,0,-1082130432,0\r" // s
#define COMMANDE_AT_ROTATION_GAUCHE  ",1,0,0,0,-1082130432\r" // q
#define COMMANDE_AT_ROTATION_DROITE  ",1,0,0,0,1065353216\r" // d

#define EPSI 0.0005

// PROTOTYPES DES FONCTIONS
void err(char*s);
voidenvoyer_AT(char* commande,unsignedint compteur,char* argument);
double comparaison (double coord1,double coord2);
booltest_arrivee(double coord1,double coord2,double coord3,double coord4);
//voiddelai( int secondes);

// VARIABLES GLOBALES
structsockaddr_inserv_addr;
intsockfd,slen;
//int tempo;

// FONCTION MAIN
intmain(void)
{

slen=sizeof(serv_addr);
if((sockfd= socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1)
{
err("socket");
}
bzero(&serv_addr,sizeof(serv_addr));
serv_addr.sin_family= AF_INET;
serv_addr.sin_port=htons(PORT);
if(inet_aton(ADRESSEIP,&serv_addr.sin_addr)==0)
{
fprintf(stderr,"inet_aton() erreur\n");
exit(1);
}

//GESTION RECEPTION DATA GPS
  FILE* final=NULL;
doublelng_desti,lat_desti;
lng_desti=lat_desti=-1;
while( final ==NULL){
final=fopen("/var/www/cgi-bin/pos_final","r");
sleep(2);
}
while(lng_desti==-1&&lat_desti==-1){
fscanf(final,"%lf %lf",&lat_desti,&lng_desti);
sleep(2);
}
//final = fopen("/var/www/cgi-bin/pos_final","r");
//fscanf(final,"%lf %lf",&lat_desti, &lng_desti);
fclose(final);

//printf("%f %f\n",lat_desti,lng_desti);

int cpt=0;
```

```
int compteur=100;
doublelat_origine,lng_origine,lat_actuelle,lng_actuelle, lat_actuelle_t0, lng_actuelle_t0, lat_actuelle_t1,
lng_actuelle_t1;

  FILE* origine;
  origine =fopen("/tmp/pos_init","r+");
if(origine !=NULL)
{
fscanf(origine,"%lf %lf",&lat_origine,&lng_origine);
fclose(origine);// On ferme le fichier qui a été ouvert
}

inti;
compteur++;
envoyer_AT("AT*REF=",compteur,COMMANDE_AT_DECOLLAGE);
sleep(3);
for(i=0;i<10;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_AVANT);
sleep(1);
}
//delai(3);


  FILE* actuelle =NULL;
  actuelle =fopen("/tmp/pos_actu","r+");
if(actuelle !=NULL)
{
fscanf(actuelle,"%lf %lf",&lat_actuelle,&lng_actuelle);
fclose(actuelle);// On ferme le fichier qui a été ouvert
}
printf("\tavant le test d'orientation\n");
//test pour connaitre approximativement la direction
if(comparaison(lat_actuelle,lat_desti)< comparaison(lat_origine,lat_desti)){
if(comparaison(lng_actuelle,lng_desti)< comparaison(lng_origine,lng_desti)){}
else{
for(i=0;i<3;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_DROITE);
sleep(1);}//90°
}
}
else{
printf("\taprès else\n");
printf("\t%f\n",comparaison(lng_actuelle,lng_desti));
printf("\t%f\n",comparaison(lng_origine,lng_desti));
if(comparaison(lng_actuelle,lng_desti)< comparaison(lng_origine,lng_desti)){
printf("\taprès if\n");
for(i=0;i<3;i++){
printf("\tboucle for\n");
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_GAUCHE);
sleep(1);}//90°
}
else{
for(i=0;i<6;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_GAUCHE);
sleep(1);}//180°
}
}

//test pour atteindre la destination
while(test_arrivee(lat_actuelle,lat_desti,lng_actuelle,lng_desti)== false){
```

```
printf("coucou\n");
    FILE* actuelle =NULL;
    actuelle =fopen("/tmp/pos_actu","r+");
if(actuelle !=NULL)
{
fscanf(actuelle,"%lf %lf",&lat_actuelle_t0,&lng_actuelle_t0);
fclose(actuelle);// On ferme le fichier qui a été ouvert
}
sleep(1);
    actuelle =fopen("/tmp/pos_actu","r+");
if(actuelle !=NULL)
{
fscanf(actuelle,"%lf %lf",&lat_actuelle_t1,&lng_actuelle_t1);
fclose(actuelle);// On ferme le fichier qui a été ouvert
}

envoyer_AT("AT*REF=",compteur,COMMANDE_AT_AVANT);
//delai(1);
if(comparaison(lat_actuelle_t1,lat_desti)< comparaison(lng_actuelle_t0,lng_desti)){
if(comparaison(lng_actuelle_t1,lng_desti)< comparaison(lng_actuelle_t0,lng_desti)){}
else{
if(cpt==0){
for(i=0;i<3;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_GAUCHE);
sleep(1);}
cpt++;}
else{
for(i=0;i<3;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_DROITE);
sleep(1);}
cpt--;}
}
}
else{
if(cpt==0){
for(i=0;i<3;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_DROITE);
sleep(1);}
cpt++;}
else{
for(i=0;i<3;i++){
envoyer_AT("AT*PCMD=",compteur,COMMANDE_AT_ROTATION_GAUCHE);
sleep(1);}
cpt++;}
}
}

envoyer_AT("AT*REF=",compteur,COMMANDE_AT_ATTERISSAGE);

final=fopen("/var/www/cgi-bin/pos_final","w+");
if(final==NULL){perror("pos_final rewrite"); exit(-1);}
else{
fprintf(final,"-1 -1");
fclose(final);
}

return0;
}
```

```
// DÉFINITION DES FONCTIONS

voiderr(char*signal)
{
perror(signal);
//exit(1); // Décommentez pour quitter le programme en cas d'erreur
}

voidenvoyer_AT(char* commande,unsignedint compteur,char* argument)
{
charsequence_number[20];
charmessage[100];
sprintf(sequence_number,"%d",compteur);
strcpy(message,commande);
strcat(message,sequence_number);
strcat(message,argument);
// #ifdef DEBUG
printf("%s\n",message);
// #endif
if(sendto(sockfd, message,strlen(message),0,(structsockaddr*)&serv_addr,slen)==-1)
{
err("sendto()");
}
//    usleep(50000); // Délai (approximatif) entre deux commandes AT
}

//voiddelai( intdelaiSecondes ){
//  while( clock()/CLOCKS_PER_SEC <delaiSecondes) {
//  envoyer_AT("AT*REF=",1,COMMANDE_AT_DECOLLAGE);
//  printf("coucou\n");
//  }
//}

double comparaison (double coord1,double coord2){
return(fabs( coord2 - coord1));
}

booltest_arrivee(double coord1,double coord2,double coord3,double coord4){
//  printf("%f %f\n",comparaison(coord1, coord2),comparaison(coord3, coord4));
if((comparaison(coord1, coord2)< EPSI)&&(comparaison(coord3, coord4)< EPSI))
return(true);
elsereturn(false);
}
```

Annexe 2 :

Code : serial_web.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include <strings.h>
#include "GPS.h"
#include <inttypes.h>
#include <signal.h>
#include <unistd.h>

struct termios saveterm;
struct sigaction action;
int fdserial;
int cpt_trame=0;

/* INIT GPIO */
void init_gpio(){
  FILE *fp0;
  char data;
       fp0=fopen("/sys/devices/virtual/misc/gpio/mode/gpio0","w");
       if(fp0==NULL){perror("gpio0"); exit(-1);}
       else {
    fgets(&data,1,fp0);
    if(data != '3')
           fprintf(fp0, "3");
              fclose(fp0);
       }
       FILE *fp1;
       fp1=fopen("/sys/devices/virtual/misc/gpio/mode/gpio1","w");
       if(fp1==NULL){perror("gpio1"); exit(-1);}
       else {
    fgets(&data,1,fp1);
    if(data != '3')
           fprintf(fp1, "3");
              fclose(fp1);
       }
}


/*INIT SERIAL*/
int init_serial(char *device, int speed){
    int fd=open(device,O_RDWR|O_NOCTTY);
    if(fd<0){perror(device); exit(-1);}
    tcgetattr(fd, &saveterm); /* save current port settings */
  struct termios newterm=saveterm;
    newterm.c_cflag=CLOCAL|CREAD|speed|CS8;
  newterm.c_iflag=0;
    newterm.c_oflag=0;
    newterm.c_lflag=0;  /* set input mode (non-canonical, no echo,...) */
    newterm.c_cc[VTIME]=0; /* inter-character timer unused */
    newterm.c_cc[VMIN]=1; /* blocking read until 1 char received */
    tcflush(fd, TCIFLUSH);
    tcsetattr(fd, TCSANOW, &newterm);
    return fd;
}
```

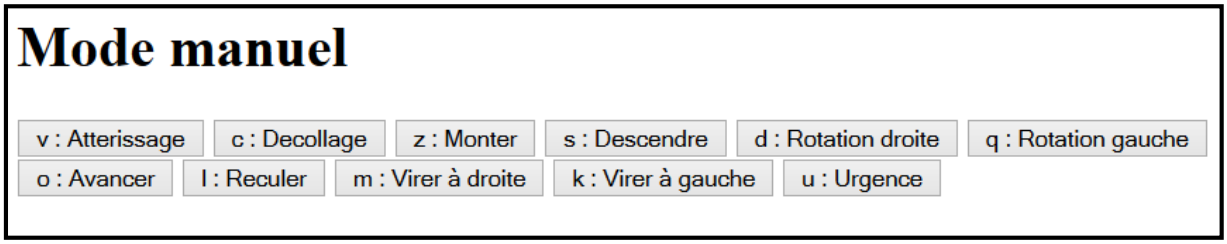
```
/*CLOSE SERIAL*/
void close_serial(int fd){
  tcsetattr(fd,TCSANOW, &saveterm);
    close(fd);
}

/* FORMATTING DATA */
void mad(char  *dataIn){
  parse(dataIn);
  if( fix){
    FILE *fp;
    double la= (double)latitude_fixed * 0.0000001;
    double lo= (double)longitude_fixed * 0.0000001;
    if ( cpt_trame == 0){
      fp=fopen("/tmp/pos_init","w+");
      if (fp==NULL){perror("pos_init");exit(-1);}
      else{
        fprintf(fp,"%lf %lf",la,lo);
        fclose(fp);
      }
    }
    else{
      fp=fopen("/tmp/pos_actu","w+");
      if (fp==NULL){perror("pos_actu");exit(-1);}
      else{
        fprintf(fp,"%lf %lf",la,lo);
        fclose(fp);
      }
    }
    cpt_trame++;
  }
}

void hand(int sig){
  if ( sig == SIGINT ){
    close_serial(fdserial);
    exit(0);
  }
}


#define MAX_CHAINE  126

/* MAIN */
int main (){
  char data[MAX_CHAINE];

  action.sa_handler = hand;
  sigaction(SIGINT, &action, NULL);

    init_gpio();
    fdserial = init_serial("/dev/ttyS1", B9600);
  if(fdserial>=0){
    FILE *entree=fdopen(fdserial,"r");
    if( entree!=NULL){
      while(fgets(data,MAX_CHAINE,entree)!=NULL)
        if(data[0]=='$') mad(data);
      fclose(entree);
    }
```

```
  }
  return 0;
}
```

Annexe 3 :

Index :

Mode manuel :



Mode automatique :