Projet SWARMON

Rapport de projet

ENSTA Bretagne
06 Juin 2014

BICHOUARINE Mouad
BOURDON Benoit
BOYE Jean-Jacques
DESCOURS Quentin
STEPHAN Simon

Introduction

La robotique en essaim (swarmrobotics) est une branche de la robotique appliquant les méthodes d'intelligence distribuée aux systèmes multi-robot. Cette technique consiste à utiliser des robots simples et peu coûteux, d'un intérêt individuel assez limité, mais qui ensemble forment un système complexe et robuste. L'emploi d'un tel nombre de robots simultanément implique le développement d'outils performants de localisation et de suivi en temps réel ou différé des éléments constituant ces meutes à des fins d'analyse des performances et des comportements.

Le but du projet SWARMON (swarm monitoring) est de créer un système permettant à un robot d'envoyer en temps réel sa position GPS vers un serveur qui centralise la position de l'ensemble des robots d'un essaim. Ces positions sont ensuite disponibles à l'affichage à partir d'un site web.

Remerciements

M. Olivier REYNET, pour sa disponibilité et toute l'assistance qu'il nous apporté tout au long de ce projet complexe.

M. Jean-Christophe LE LANN, pour ses conseils pertinents dans la gestion du projet.

M. Gilles LE MAILLOT, pour l'aide qu'il nous apporté dans la conception de notre carte électronique personnalisée.

PROTECNO, pour la réalisation des cartes électroniques personnalisées.

L'ENSTA Bretagne, pour les ressources et le budget qu'elle a accordé au projet et la visibilité qu'elle nous a donné à travers la Journée Portes Ouvertes.

stackoverflow.com, qui apporte chaque jour la lumière aux développeurs en herbe.

Table des matières

Introduction 1
Remerciements 2
Présentation du projet 5
État de l'art 6
1. Analyse de l'existant 6
2. Évaluation de l'existant par rapport à l'objectif du projet 6
3. Solutions possibles : méthodes, algorithmes, matériels, logiciels 6
4. Solutions retenues 7
Cahier des charges fonctionnel 8
1. Exigences négociées avec le client et fonctions du système 8
2. Quelles fonctions émergent par l'approche Top-Down?[9] 8
3. Quelles fonctions émergent de l'analyse des exigences ? 10
Architectures 11
1. Architecture fonctionnelle 11
2. Architecture physique 12
3. WBS 13
4. Diagramme de GANTT prévisionnel 14
5. Diagramme de GANTT réel 14
API 16
1. Base de données 16
2. Format des données GPS 17
Organisation 19
1. Identification/affectation des tâches 19
2. Méthodes de travail 19
Ruby on Rails 21
1. Présentation Rails 21
2. Patron de conception Modèle-Vue-Contrôleur 21
2.1 Modèle 21
2.2 Vue 22

2.3 Contrôleur 22
3. Développement dirigé par les tests 23
Présentation des réalisations 24
1. Initialisation du module SIM5218 24
2. Validation du GPS 25
3. Validation de l'envoi des coordonnées avec le SIM5218 27
4. Page Web 28
5. Authentification 30
6. Estimation du coût de développement du projet SWARMON : 32
7. Conception de la carte électronique 34
Conclusion 37
Références 38
Annexe 1 : Lexique 39
Annexe 2 : Extrait de documentation de la carte de développement 41
Annexe 3 : Configuration et utilisation d'un module SIMCOM 5218 43
Démarrage pour la configuration du SIMCOM avec la carte de développement 43
Accès au terminal série via PuTTY 43
Accès au terminal série via Python Serial (pas de lecture du port) 43
Configuration de l'accès internet 43
Commandes de test pour poster un visiteur sur le prototype CGI sur Haggis 44
Utilisation des scripts LUA 44
Annexe 4 : Script Lua - Envoi des coordonnées GPS du tracker au serveur. 45
Annexe 5 : Plan de validation fonctionnelle 46
Test unitaire 46
Test d'intégration 48

Présentation du projet

Ce travail vise à mettre au point un système performant de localisation en temps réel d'une meute de robots. Il s'inscrit dans un contexte relativement précis, l'organisation de la World Robotic Sailing Championship par notre école, l'ENSTA Bretagne.

Dans un tel contexte, il peut être intéressant pour un opérateur (concurrent, arbitre ou observateur) de pouvoir suivre le comportement des robots de l'essaim à des fins de contrôle et de sécurité.

La WRSC met en avant le besoin de connaitre en temps réel la position des différents drones marins, tout au long de l'épreuve. Ainsi, la connaissance permanente de la position de chaque robot participant à la compétition offre des garanties supplémentaires de sécurité tout en simplifiant la tâche du jury concernant la validation de l'épreuve. Cette solution de suivi permet une meilleure équité dans l'attribution des notes, offrant la possibilité de revivre la course en différé. De plus, une telle solution technique devrait permettre aux concurrents d'analyser leur prestation. Enfin, la retransmission en direct de l'évolution de la course est un atout non négligeable pour la médiatisation de cette épreuve.

L'objectif du projet SWARMON est de proposer une formule de tracking basée sur le positionnement GPS. S'il aboutit à une solution technique viable, ce procédé pourrait s'intégrer dans l'organisation et le déroulement des prochaines compétitions WRSC, et contribuerait ainsi au rayonnement de l'école. D'autre part, la solution proposée dépasse le contexte de la robotique marine et vise à s'adapter au suivi de meutes de drones terrestres ou aériens.

Le développement du projet SWARMON a été débuté par un élève de l'ENSTA Bretagne, dans le cadre de son PFE. La solution retenue pour cette première phase était basée sur un smartphone Android.

La seconde phase du développement consiste à remplacer le système Android par une solution de tracking indépendante. Une amélioration du serveur, de la base de données et du client web est également attendue.

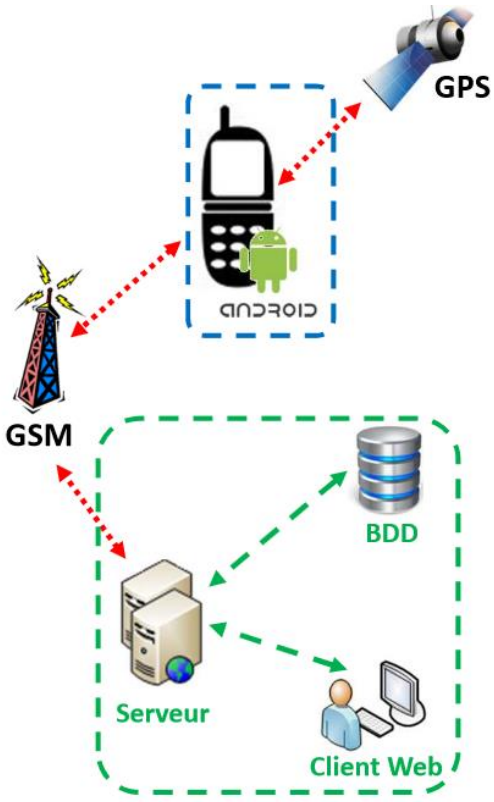


Figure 1. Architecture du projet existant

État de l'art

1. Analyse de l'existant

Aujourd'hui, on peut trouver pour une centaine d'euros des tracker[Lexique]GPS performants proposant un suivi par SMS (le très populaire TK-102 par exemple[1]).

Il existe aussi des services en ligne proposant ce type de tracker, mais, au lieu du suivi par SMS, on peut surveiller les déplacements du tracker via un site web.

Enfin, les smartphones, tous équipés d'un GPS et d'un module 3G, sont un moyen de plus en plus populaire pour suivre une personne ou un objet, via des applications disponibles sur les différents Store, ou en créant la sienne.

Actuellement, le système embarqué consiste en un smartphone Android dans un boîtier. Les positions sont envoyées au serveur par une application Android. Le serveur a exploité dans un premier temps une application codée en PHP et utilisant une base de données MySQL, avant de migrer vers une application utilisant le Framework Sinatra (un Framework web léger en Ruby). Les données sont transmises par les clients au serveur par SMS ou requête HTTP via le réseau 3G. Les positions des bateaux sont affichées sur le site web via l'API Google Maps.

2. Évaluation de l'existant par rapport à l'objectif du projet

L'architecture actuelle pose un certain nombre de problèmes :

- l'application Android est instable, problème en partie réglé par un usage en tant que daemon[Lexique];
- certains procédés (l'authentification par exemple) sont codés de manière minimale et peu robuste;
- pas de remontée d'informations vers le tracker qui pourrait, par exemple, prévenir le drone marin d'une possible collision avec un concurrent.

3. Solutions possibles : méthodes, algorithmes, matériels, logiciels

Pour adresser les problèmes détaillés ci-dessus, un certain nombre de décisions ont été prises :

- développement d'un démonstrateur matériel basé sur un microcontrôleur ou un Single Board Computer ARM [2] intégrant un GPS et un modem 3G dont le système embarqué sera plus stable qu'Android et qui permettra de s'adresser au serveur directement par des requêtes HTTP. Le démonstrateur aura en outre une interface de communication avec le drone marin (probablement via un lien série) pour lui transmettre des alertes.

Si le projet avance suffisamment vite, l'intégration d'un modem Xbee[Lexique] [3] [4] pour permettre des communications directes entre les drones ainsi que des capteurs de mouvement pour déterminer la direction du drone est envisagée.

- refonte complète de l'application serveur, qui sera développé en utilisant un Framework[Lexique]web complet qui permettra de s'affranchir du codage d'un certain nombre de fonctionnalités comme l'authentification.

Pour le démonstrateur matériel, un module comprenant un modem 3G [Lexique]et un GPS, avec lequel on communique "simplement" par des commandes série sera utilisé, après avoir hésité avec un modem 3G et un capteur GPS séparés (peut-être moins difficiles à prendre en main individuellement, mais plus encombrants et probablement plus énergivore).

Le choix d'une BeagleBone Black [Lexique] [5] avait été fait dans le cas de l'utilisation de plusieurs capteurs, la carte offrant une connectique complète et une puissance de calcul importante (utile pour le positionnement via des accéléromètres envisagés en objectif secondaire).

Cependant, l'utilisation d'un combo 3G/GPS a remis en cause ce choix, d'autant que l'intégration des objectifs secondaires semble difficile dans le temps imparti. Un simple microcontrôleur de type Arduino[Lexique] [3] [6] serait alors probablement suffisant pour contrôler le module 3G/GPS, et serait beaucoup plus économe en énergie que la BeagleBone Black [5] (important étant donné que le système devra fonctionner sur une source d'énergie embarquée pendant plusieurs heures).

L'application serveur utilisera le Framework web Ruby on Rails[Lexique] [8], qui nous dispense du codage d'un grand nombre de fonctions de base (interface avec la base de données, authentification via des Gem[Lexique]) et permettra probablement d'obtenir une application plus robuste.

4. Solutions retenues

Nous utiliserons un module 3G+GPS SIMCOM SIM5218 comme composant de base pour le tracker. Il nécessite une carte fille pour la connectique et l'alimentation. Tous les développements et tests ont jusqu'ici été faits avec la carte de développement fournie par le constructeur, le démonstrateur présenté en Juin utilisera cette carte. Une carte fille personnalisée est en cours de réalisation par la société Protecno et sera à priori utilisable en septembre pour la WRSC.

Le module fonctionne grâce à des scripts LUA qu'il est capable d'exécuter nativement.

Cahier des charges fonctionnel

1. Exigences négociées avec le client et fonctions du système

Le cahier des charges fonctionnel (CDCF) est un document formulant le besoin, au moyen de fonctions détaillant les services rendus par le produit et les contraintes auxquelles il est soumis.

Actuellement, le système embarqué est un Smartphone Android. Les données sont transmises par les clients au serveur par SMS ou requêtes HTTP. Les positions des bateaux sont affichées sur le site web via l'API [10] Google Maps.

Le client souhaite améliorer les capacités de rafraichissement du système embarqué. De plus, le client recommande l'abandon de la plateforme Android et du smartphone au profit d'un module 3G/GPS, afin de maitriser au mieux le système embarqué.Enfin, le système doit être positionné sur un drone marin. Ce qui implique une résistance au milieu marin, une certaine autonomie et un encombrement minimum.

2. Quelles fonctions émergent par l'approche Top-Down?[9]

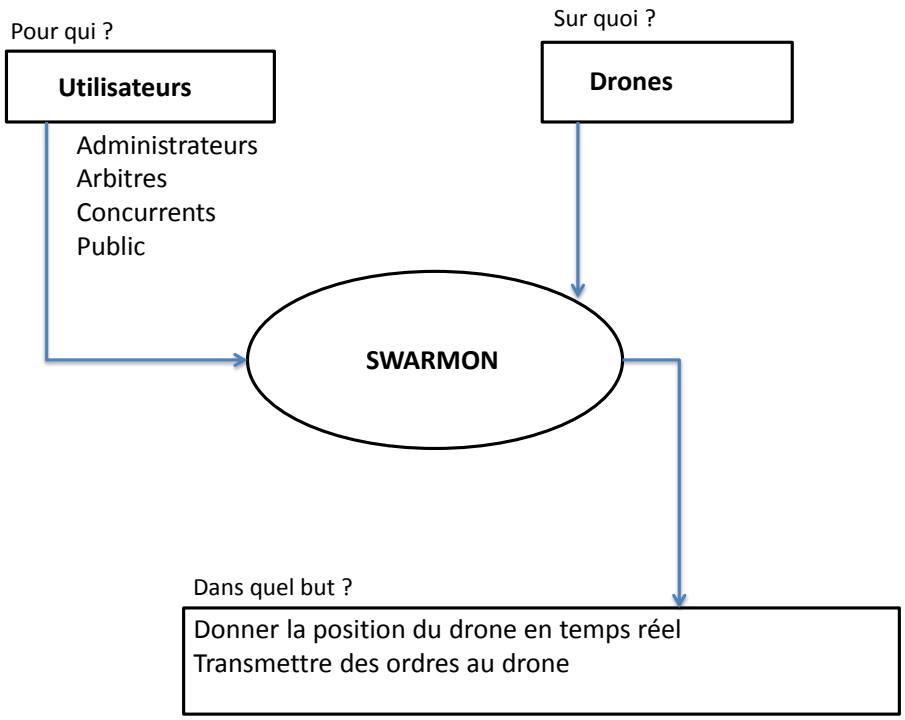
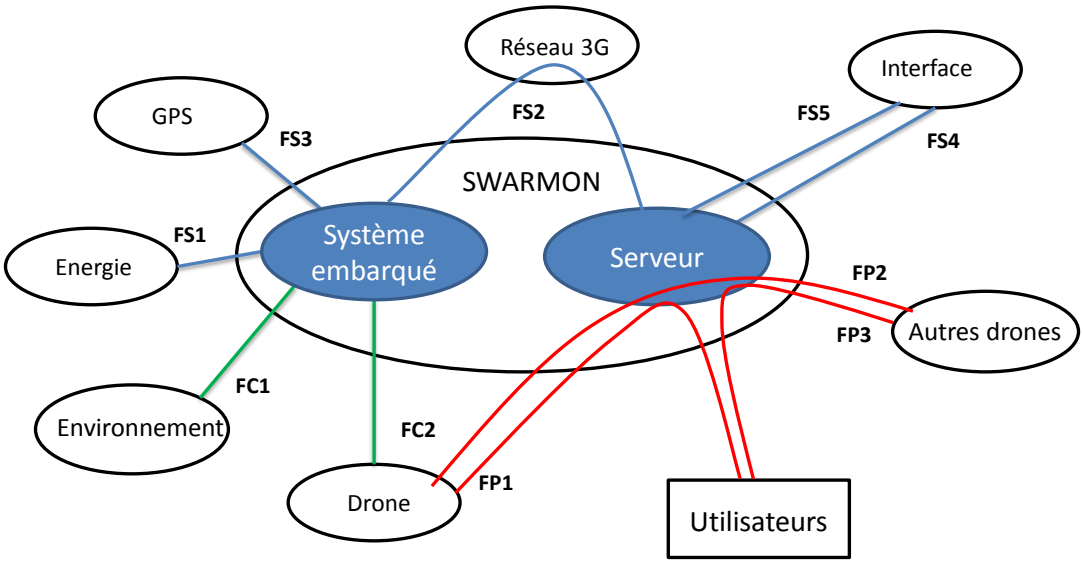


Figure 2. Synthèse de la méthode APTE : Diagramme de « la Bête à cornes »

Le système SWARMON (boitier+serveur) permet de donner la position du drone en temps réel. Le boitier fixé sur le drone envoit des informations au serveur pour qu'un utilisateur puisse accéder à la position du drone. Notre système a vocation à être utilisé lors de compétitions de robotique. Les utilisateurs sont donc les participants, l'arbitre, les organisateurs et le public.



FP1: Permettre à l'utilisateur de connaître la position du drone en temps réel
FP2: Etre capable d'éviter une collision avec d'autres drones
FP3: Permettre à l'utilisateur de donner un ordre à la meute de drones

FC1: Résister milieu marin
FC2: S'adapter au drone (porteur)

FS1: Etre autonome en énergie
FS2: Communiquer avec le serveur via le réseau 3G+
FS3: Calculer sa position avec le système GPS
FS4: Proposer un IHM graphique
FS5: Proposer un accès direct via l'API

Figure 3. Synthèse de la méthode APTE : Diagramme « Pieuvre »

Notre système est composé d'un boitier étanche (système embarqué) à fixer sur le drone ainsi que d'une partie serveur servant à traiter les informations émises par le drone.

Système embarqué :

Le système embarqué doit s'adapter au robot et à l'environnement. Pour cela il est étanche [FC1] et permet de communiquer avec le drone [FC2].Il reçoit de l'énergie provenant d'une batterie, reçoit des signaux GPS et communique via le réseau 3G.

Serveur :

Le serveur reçoit les informations provenant du réseau 3G. L'utilisateur peut consulter la position du drone soit via l'API soit via une interface graphique (page web).

Le dispositif permet également d'éviter des collisions en connaissant les trajectoires des autres drones. L'utilisateur peut aussi envoyer lui-même un signal d'alerte au drone.

3. Quelles fonctions émergent de l'analyse des exigences ?

Analyse Fonctionnelle *Bottom-up*.

Les exigences formulées par le client nous permettent de mettre en place ce tableau.

N°	Fonction	N°	Sous-fonction	Critère	Niveau
1	Normaliser les communications	1.1	Editer un API	Utilisable par le serveur, les utilisateurs et les boîtiers	/
2	Acquérir la position du drône	2.1	Localiser la position du drone par GPS	Précision	< 5 mètres
		2.2	Formater les données	Protocole NMEA	/
		2.3	Communiquer par le réseau 3G	/	/
		2.4	Stocker les données récoltées	Mémoire interne	> 50 Mo
		2.5	Sécuriser l'accès au données	Authentification	/
3	Fournir la position du drône à l'utilisateur via une interface	3.1	Remplir la BDD	/	/
		3.2	Permettre un affichage temps réel	Rafraichissement	< 10 s
		3.3	Editer un site Web en utilisant un Framework	Framework utilisé	RoR
		3.4	Afficher une IHM graphique	Design	/
4	Concevoir un démonstrateur physique permettant de localiser le drône	4.1	Maîtriser le coût	Prix	<300 € (primaire) <200€ (secondaire)
		4.2	Résister au milieu marin	Etanchéité	Eau salée
		4.3	S'adapter à l'énergie disponible	Autonomie	> 6 h
		4.4	S'adapter au drone	Poids	/
				Encombrement	/
5	Détecter les risques de collision et alerter le drône	5.1	Evaluer le risque de collision	distance et cap	à déterminer
		5.2	Alerter le boîtier	temps pour donner l'alerte	< 10s
		5.3	communiquer avec le drone	interface	lien série

Figure 4. Tableau des exigences

Architectures

La corrélation entre les analyses Top-Down et Bottom-Up nous permet de faire apparaître les différentes fonctions relatives au système. L'architecture fonctionnelle ci-dessous nous permet de faire apparaître les flux entre les différentes fonctions.

1. Architecture fonctionnelle

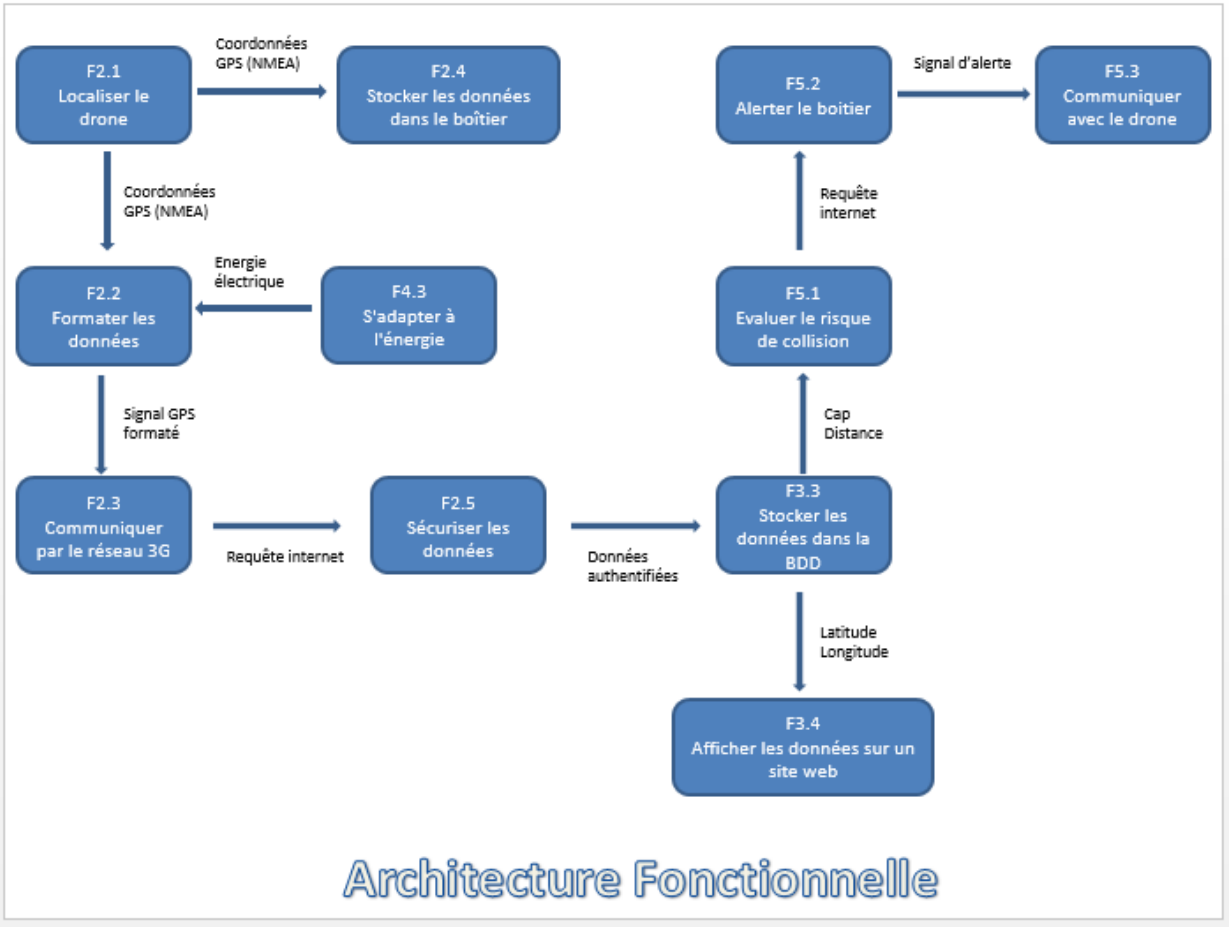


Figure 5. Architecture fonctionnelle

2. Architecture physique

L'architecture physique nous permet d'associer à chaque fonction le(s) composant(s) physique(s) ou logiciel(s) permettant de répondre aux besoins.

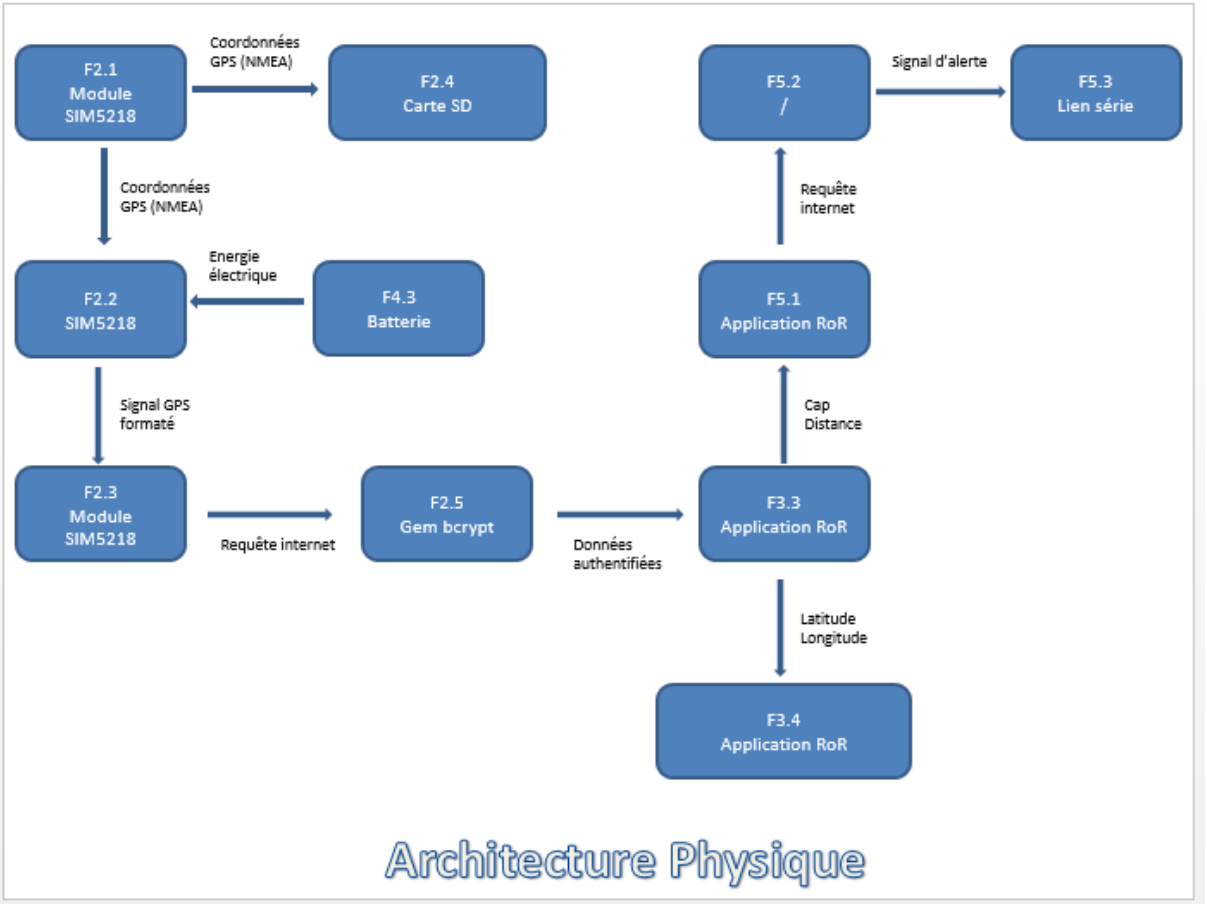


Figure 6. Architecture physique

On en déduit la matrice d'allocation suivante :

	SIM5218	Carte SD	Application RoR	Lien série	Circuit de charge + batterie
F2.1 Localiser le drône	X				
F2.2 Formater les données	X				
F2.3 Communiquer par le réseau 3G	X				
F2.4 Stocker les données dans le boîtier		X			
F2.5 Sécuriser les données			X		
F3.3 Stocker les données dans le BDD			X		
F3.4 Afficher les données sur un site web			X		
F5.1 évaluer le risque de collision			X		
F5.3 Communiquer avec le drône				X	
F4.3 S'adapter à l'énergie					X

Figure 7. Matrice d'allocation

3. WBS

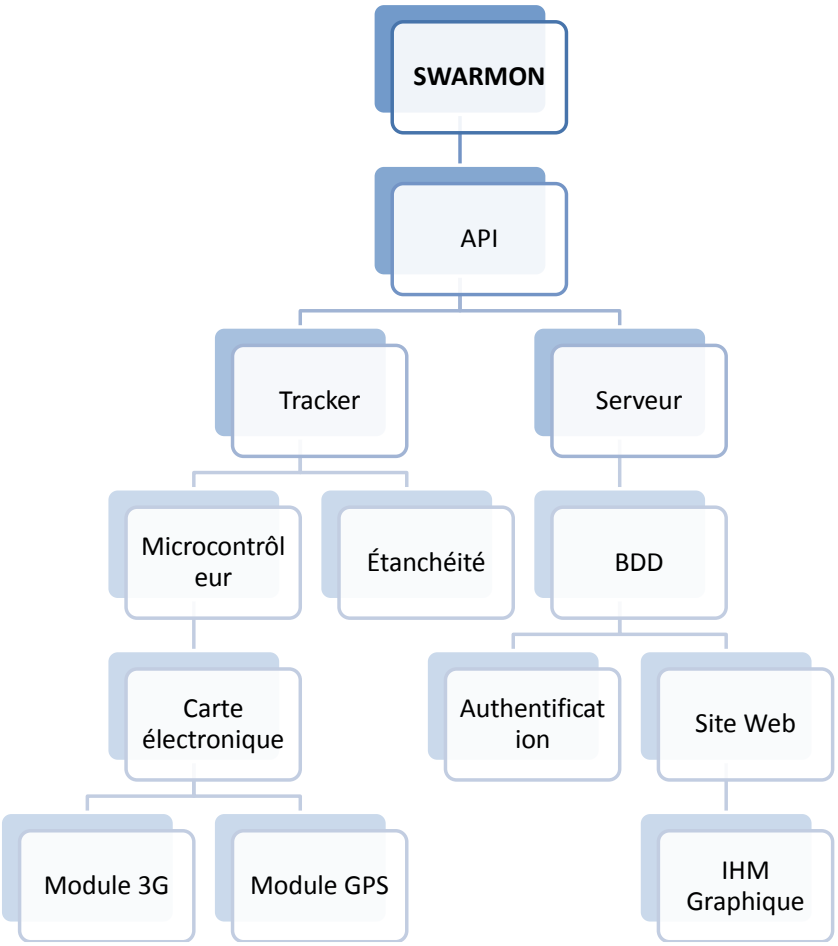


Figure 8. Diagramme WBS

Le diagramme WBS nous permet de commencer à paralléliser le travail en vue du diagramme Gantt. Nous voyons par exemple qu'une séparation des tâches sera possibles entre la partie software (logiciel) et la partie hardware (physique). Ceci est d'ailleurs en accord avec la répartition des ateliers techniques réalisés par les membres de l'équipe.

4. Diagramme de GANTT prévisionnel

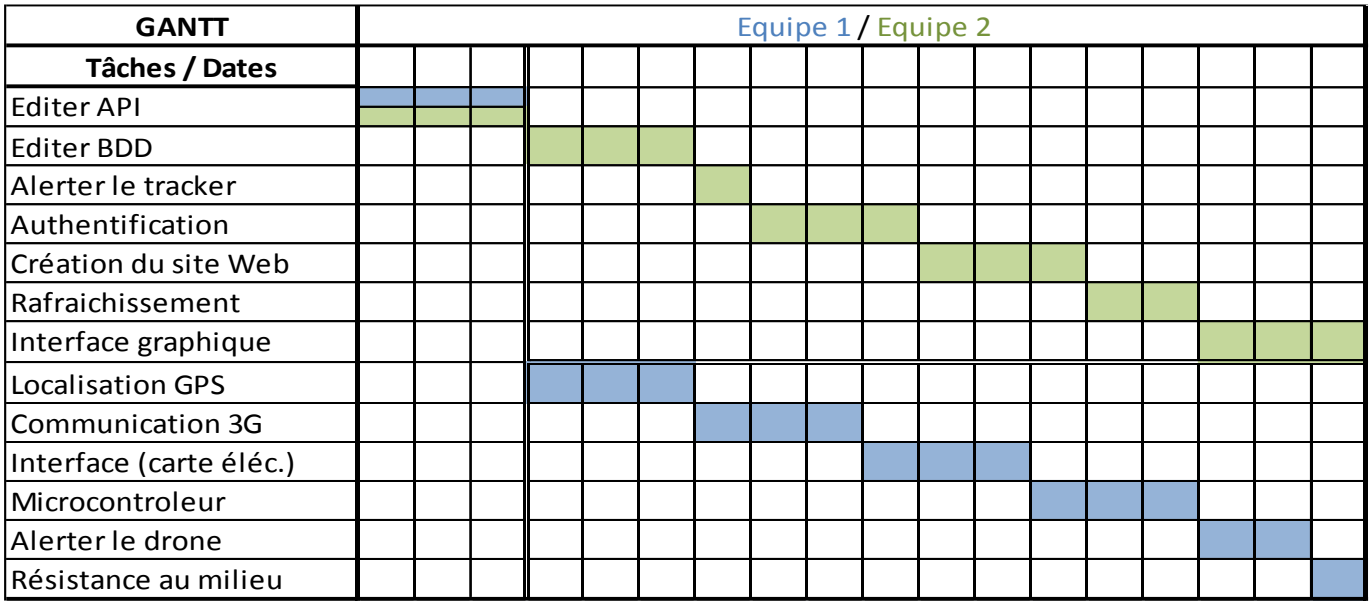


Figure 9. Diagramme de GANTT prévisionnel

Le diagramme de GANTT met en avant les différentes tâches à effectuer. On voit ici que la séparation du travail entre Hardware et Software est la plus logique.Les équipes 1 et 2 ne sont pas fixées à priori. Les personnes constituant le groupe peuvent changer d'équipe et la taille des équipes n'est pas fixe.

5. Diagramme de GANTT réel

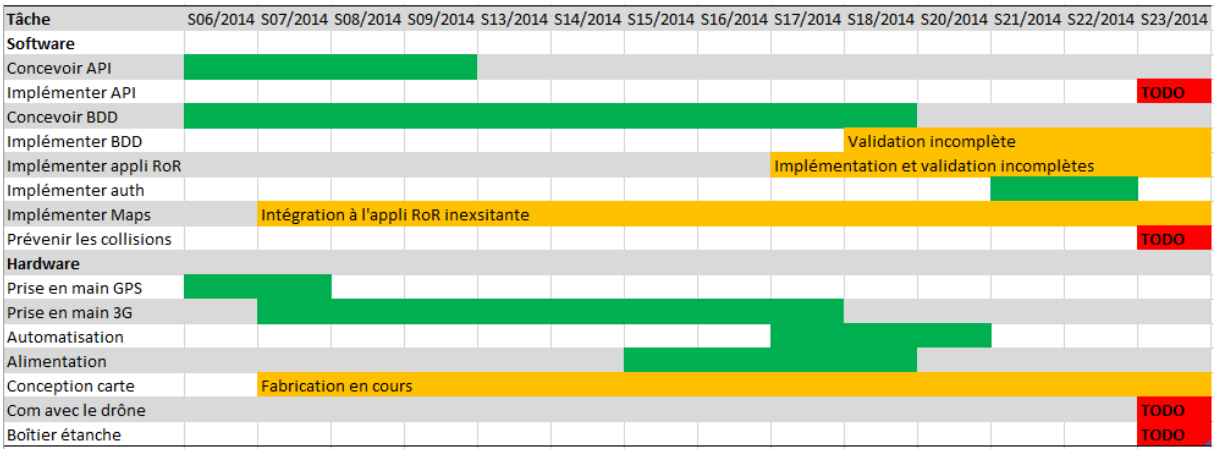


Figure 10. Diagramme de GANTT réel

On constate que la conception de la base de données a été extrêmement longue, et la création de l'application RoR n'est pas encore terminée notamment à cause de la complexité du framework.

Ce retard pris dans la partie software a entrainé nous a incité à tous nous y consacrer depuis 1 mois, d'où l'absence de certaines fonctions hardware à ce stade.

Enfin, la carte électronique personnalisée dont le design a été finalisé (après un travail de recherche de longue haleine) et commandée n'a pas encore été livrée, aucun test n'a donc encore été fait avec ce matériel.

API

1. Base de données

L'API : en informatique une interface de programmation (abr. API pour Application Programming Interface [11]) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur.

L'API est un des premiers travaux à effectuer car il permet de séparer le travail sur le tracker et le site web tout en conservant une cohérence entre les deux phases de notre projet.

L'API doit être lue en parallèle de la base de données pour être assimilée facilement [11].

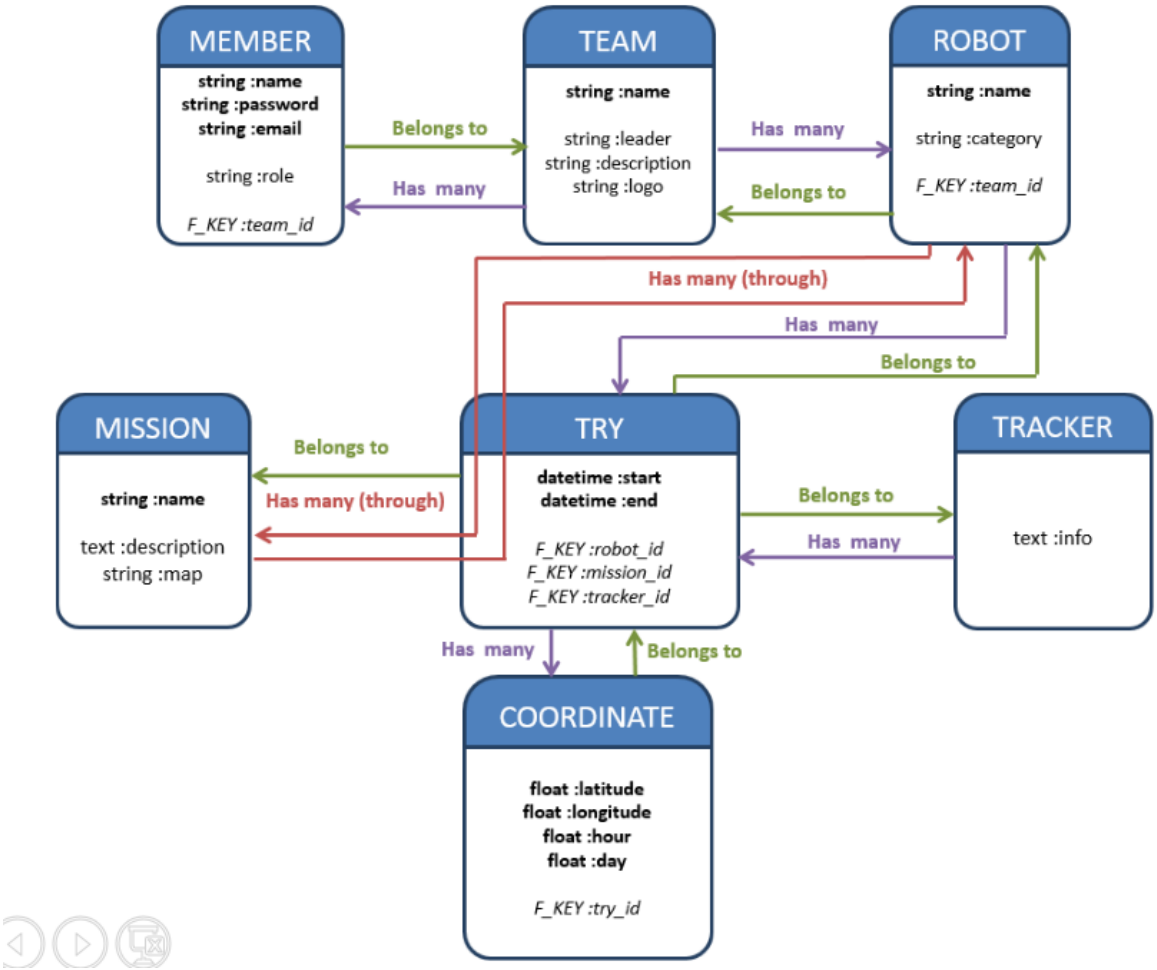


Figure 11. Base de données du site web

URL	GET	POST	DELETE	PUT
member	ME / A / S	S		
members	ME / A / S			
member/name	ME / A / S		M	M / CE / A
team	ME / A / S	A		
teams	ME / A / S			
team/name	ME / A / S		A	CE / A
robot	CE	CE		
robots	CE			
robot/id	CE		CE	CE
mission	A	A		
missions	A			
mission/name	A		A	A
try	ME / A / S	ME / A		
tries	ME / A / S			
try/number	ME / A / S		A	ME / A
tracker	A	A		
trackers	A			
tracker/id	A		A	A
coordinate	ME / A	T		
coordinates	ME / A			
coordinate/num	ME / A			

T = tracker
CE = chef d'équipe
ME = membre d'équipe
membre pour son compte
A = arbitre
S = spectateur

Figure 12. Schéma API

2. Format des données GPS

Il existe plusieurs formats disponibles pour les données GPS de la carte SIM5218.

Deux ports peuvent être utilisé, le port série qui renvoi les données lorsque que la commande AT+CGPSINFO est envoyé, ou le port NMEA qui diffuse constamment des données GPS lorsqu'il est activé.

Le port NMEA envoi les données en plusieurs formats différents, voici un exemple.

```
$GPGSV,3,1,11,02,19,142,20,08,16,045,21,09,19,180,22,10,41,063,29*7B
$GPGSV,3,2,11,15,77,317,46,18,09,286,31,21,28,315,43,24,73,310,47*7F
$GPGSV,3,3,11,26,53,241,31,27,39,158,20,29,30,250,41*45
$GPGGA,013925.0,3113.340070,N,12121.176208,E,1,08,2.9,164.0,M,,,,*08
$GPRMC,013925.0,A,3113.340070,N,12121.176208,E,,,150509,,,A*61
$GPGSA,A,3,08,10,15,18,21,24,26,29,,,,,4.4,2.9,3.3*37
$GPVTG,,T,,M,0.0,N,0.0,K*4E
```

La commande AT+CGPSINFO renvoi :

CGPSINFO:3113.343286,N,12121.234064,E,250311,072809.3,44.1,0.0,0

Les deux formats contiennent autant d'information. Il est plus simple d'utiliser la commande AT+CGPSINFO car le format est plus court, il n'y a pas de redondance.

Il est aussi plus facile d'enregistrer les données en utilisant la commande car dans le cas du port NMEA, les données arrivent en flux continu qui demande une synchronisation et une écoute constante du port. Ce problème est résolu par l'utilisation de la commande AT+CGPSINFO.

```
CGPSINFO: [<lat>],[<N/S>],[<log>],[<E/W>],[<date>],[<UTCtime>],[<alt>],[<speed>],[<course>]
Defined values
<lat>
Latitude of current position.
Output format is ddmm.mmmm
<N/S>
N/S Indicator, N=north or S=south
<log>
Longitude of current position. Output format is dddmm.mmmm
<E/W>
E/W Indicator, E=east or W=west
<date>
Date. Output format is ddmmyy
<UTC time>
UTC Time. Output format is hhmmss.s
<alt>
MSL Altitude. Unit is meters.
<speed>
Speed Over Ground. Unit is knots.
<course>
Course. Degrees.
```

Figure 13. Description de la commande AT+CGPSINFO

Organisation

1. Identification/affectation des tâches

L'analyse effectuée grâce aux diagrammes WBS et Gantt met en exergue deux fonctions principales:

- créer un boitier comportant l'électronique embarqué
- traiter les différents flux d'informations pour afficher la position du drone sur un site web

Nous avons "scindé" le groupe en deux sous-groupes selon l'appétence des différents membres envers le Hardware ou le Software.

En effet lors du premier atelier technique Quentin, Jean-Jacques et Benoit ont suivi la formation Rubi On Rails [7] alors que Simon et Mouad ont participé à la formation Android (pour comprendre le système déjà existant). Lors du second atelier technique Mouad et Benoit ont suivis la formation carte électronique alors que Simon, Jean-Jacques et Quentin ont suivis la formation Linux pour l'embarqué[Lexique] [7].

Cependant nous voulons également que chacun des membres de l'équipe soit aussi polyvalent que possible.Pour cela nous organisons à chaque début de séance un "point compétences acquises" (cf. méthodes de travail).

2. Méthodes de travail

Philosophie de travail :

Nous essayons dès que possible de travailler selon la philosophie "test driven" c'est-à-dire en procédant pas à pas, par petites incrémentations, en effectuant des tests unitaires dès que c'est possible.

Pour la partie software, le logiciel Ruby on Rails est dans cet état d'esprit car il permet rapidement et facilement d'effectuer des tests.
Dans la même idée la partie hardware il sera question d'acquérir le matériel au plus vite pour "le prendre en main" et faire dès que possible des tests simples (notion de "time to blink" qui consiste à s'intéresser au temps de développement nécessaire avant que le système fasse quoique ce soit, par exemple clignoter une LED)

Réunions :

Nous essayons de mettre en place des petites réunions (10min max) dès que possible pour maintenir la cohésion du groupe et optimiser le temps de travail.
Il y a les "mini-réunions de travail". En début de séance elles servent à repartir les taches (création et attribution de petites taches nommées "sprints"[Lexique]). En fin de séance on vérifie l'avancement ou la réalisation des différents sprints et on rédige la TODO-list (liste des taches a effectuées lors de la prochaine séance).

Nous avons également mis en place des "Points compétences acquises": réunion d'environ 10min lors de laquelle chaque membre de l'équipe fait part au groupe des compétences qu'il a acquis et les résume à l'essentiel (si besoin à l'aide de schémas).

Méthodes de travail supplémentaires:

Nous stockons tous nos fichiers sur le Skydrive fournit par l'école. Cela permet à plusieurs personnes de modifier en même temps le même document.
Nous mettons à disposition des encadrants sur le Wikiun maximum d'informations sur l'avancée du projet et les travaux réalisés.

Nous avons mis en place un système très simple pour savoir à tout moment quels sont les membres du groupe qui ont lus un document (en bas de page figurent les noms des membres du groupe, ils n'ont qu'à compléter si nécessaire).

Le développement de l'application RoR se fait de manière collaborative à travers la plateforme GitHub qui permet de gérer les versions du programme et de merger facilement des travaux sur des parties différentes de l'application.

Ruby on Rails

1. Présentation Rails

Ruby on Rails est un Framework écrit en Ruby avec un haut niveau d'abstraction permettant de faciliter le développement d'une application web avec un grand nombre de fonction et de conventions déjà établis. Il suit le patron de conception Modèle-Vue-Contrôleur facilitant la modification et l'ajout de nouvelles fonctionnalités.

2. Patron de conception Modèle-Vue-Contrôleur

Le patron de conception Modèle-Vue-Contrôleur ou Présentation-Abstraction-Contrôle sert à diviser une application en trois parties interconnectées afin de mieux organiser son code et faciliter l'implémentation de nouvelles fonctionnalités, règles ou visuels. Ci-dessous un schéma expliquant l'interaction que peut avoir un utilisateur avec une application de ce genre :

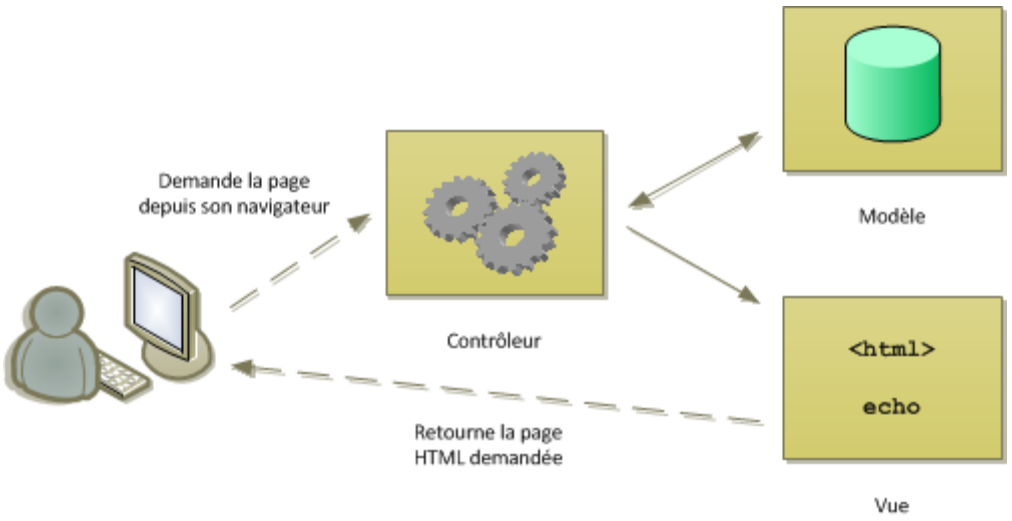


Figure 14. Schéma du modèle MVC

2.1 Modèle

Gère tout ce qui concerne la Base de données : traitement des données fournies par l'utilisateur, établissement des règles pour stocker ces données, les mettre à jour ou en récupérer d'autres de la BDD. Dans notre cas, la partie Modèle du Ruby on Rails hérite de la classe ActiveRecord qui se chargent de faire la liaison avec la BDD. Ci-dessous un extrait de code d'un des modèles de notre application Rails :

```
class Member<ActiveRecord::Base
  belongs_to:team

  validates:name, presence: true, uniqueness: true, length: { in: 3..30,
```

```
    too_long: "%{count} charactersis the maximum allowed",
    too_short:"%{count} charactersis the minimum allowed"  }
  validates:password, presence: true, length: { in: 6..30,
    too_long: "%{count} charactersis the maximum allowed",
    too_short:"%{count} charactersis the minimum allowed"  }
  validates:role, inclusion: { in: %w(administratorvisitorplayer),
    message: "%{value} is not a validrole" }, allow_nil: true
  validates:email, presence: true, uniqueness: true, on: :create
  validates:team,  presence: {if: :player?}

  defplayer?
    role == "player"
  end
end
```

Ce modèle instaure des règle quant à l'ajout d'un nouveau membre comme par exemple le fait que le nom d'un nouveau membre devrait impérativement être spécifié, qu'il ne soit pas déjà existant dans la BDD et qu'il contient un nombre limité de caractères.

2.2 Vue

La vue établit l'interface avec l'utilisateur et traduit ses choix en événements envoyés au contrôleur. Elle hérite de la classe ActionView implémentée en ERb (Embedded Ruby) qui gère les templates de présentation. Le code ci-dessous présente un exemple de vue correspondant au modèle précédant :

```
. . .

  <div class="field">
    <%= f.label:email %><br>
    <%= f.text_field:email %>
  </div>
  <div class="field">
    <%= f.label:role %><br>
    <%= f.select:role, ['visitor','administrator','player'] %>
  </div>
  <div class="field">
    <%= f.label :team_id %><br>
    <%= f.collection_select(:team_id, Team.all, :id, :name) %>
  </div>
. . .
```

2.3 Contrôleur

Permet de recevoir les événements depuis les vues, les traiter avec les actions appropriées, envoyer les données traitées au modèle puis notifier l'utilisateur de l'avancement de son opérateur dans la vue appropriée. Ci-dessous un extrait du contrôleur correspondant à la vue et au modèle précédant :

```
. . .
defedit
  end

  defcreate
```

```
    @member = Member.new(member_params)

    respond_to do |format|
      if @member.save
        format.html { redirect_to @member, notice:
'Memberwassuccessfullycreated.' }
        format.json{ render :show, status: :created, location: @member }
      else
        format.html { render :new }
        format.json{ renderjson: @member.errors, status:
:unprocessable_entity }
      end
    end
  end
. . .
```

On voit dans l'exemple précédant que la méthode « edit » ne contient pas de code, c'est parce que notre contrôleur hérite de la classe ActionController où cette fonction est déjà définie.

3. Développement dirigé par les tests

Le développement dirigé par les tests est une technique de développement qui consiste en l'écriture de tests unitaires au fur et à mesure que le code progresse. L'écriture du test d'un modèle consiste par exemple en l'ajout d'un nouvel élément dans la BDD de tel sorte que ce test échoue au début pour être sûr que les règles instaurées dans le modèle sont bien respectées. Ces tests peuvent s'avérer incontournables si on veut effectuer un changement radical dans notre application et être sûr que cela ne nuira pas au reste du code. Ci-dessous un extrait d'un des tests de notre application :

```
test "Member has a name" do
    m=Member.new
    assert_notm.save, "Membershould not beempty..."
    m.email="okok@okok.fr"
    m.password="okokokokoko"
    m.team=teams(:Team1)
    #Name...
    m.name="o"
    assert_notm.save, "Membershould have a name... but length>=3"
    m.name="ok"
    assert_notm.save, "Membershould have a name... but length>=3"
    m.name="oko"
    assertm.save, "Membershould have a name... but length>=3
==>okoshouldbetrue"
    m.name="okok"
    assertm.save, "Membershould have a name... but length>=3
==>okokshouldbetrue"
  end
```

Présentation des réalisations

Cette partie présente les différents travaux que nous avons effectués pour remplir les fonction de notre système. Le plan de validation fonctionnel en Annexe 5 résume notre travail.

1. Initialisation du module SIM5218

Le SIMCOM SIM5218 est un module 3G/GPS. Les capacités de ce module correspondent parfaitement aux fonctions que nous souhaitons implémenter. Il est capable de se connecter à un réseau 3G, envoyer des requêtes HTTP, repérer sa position GPS, enregistrer des informations et envoyer des commandes via un lien série. Le module est aussi capable d'exécuter des scripts écrits dans le langage Lua. Cela nous permet de nous passer d'un microcontrôleur additionnel car le SIM5218 possède son propre système embarqué.

Figure 15. Module SIMCOM 5218

Afin d'effectuer notre travail, nous utilisons la carte de développement vendu par le fabriquant du module. Un extrait de la documentation de la carte de développement se trouve en annexe 2.

Pour communiquer avec le module, nous utilisons un terminal série (minicom ou puTTY par exemple) communiquant par le port série de la carte de développement. Grace à cette interface, nous pouvons lancer directement des commandes sur le module.

Pour fonctionner, le module requiert la présence d'une carte SIM. L'annexe 3 explique comment configurer la carte SIM et communiquer avec le module via puTTY.

2. Validation du GPS

Afin de valider la fonction FP2.1, *localiser la position du drone par GPS* avec un critère de précision inférieur à 5 mètres, nous avons réalisé un comparatif avec un GPS différentiel utilisé en hydrographie.

Le test s'est déroulé le 22 avril 2014, à 14h, sur la piste d'athlétisme de l'ENSTA Bretagne. La météo était favorable à ce type de test.

Pour ce test, nous avons démarré les deux GPS en même temps. Après avoir obtenu un fixe sur chaque appareil, nous avons entamé un tour de la piste. Après traitement des données, nous pouvons observer les résultats ci-dessous.

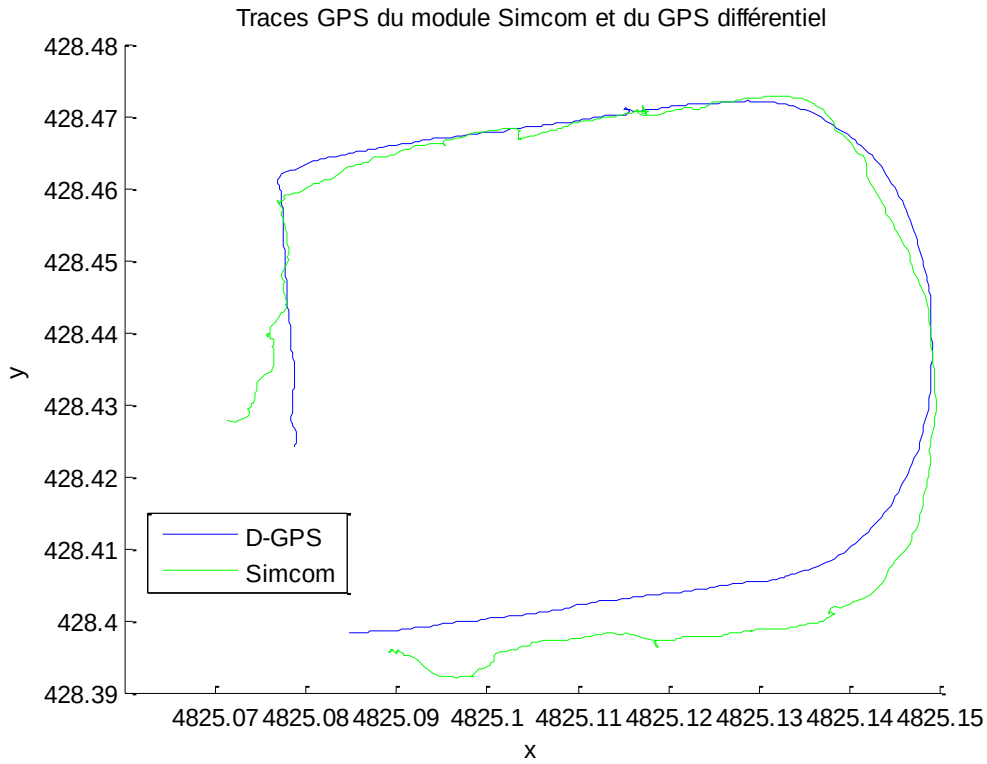


Figure 16.Traces GPS du module SIM5218 et du GPS différentiel

La trace bleue du GPS différentiel sera notre référence. Nous pouvons remarquer que le module SIM5218 est capable de fournir un position correcte une fois en mouvement. On observe un décrochage de sur la seconde partie du trajet. Cette zone se situe à proximité des ateliers de l'ENSTA Bretagne, peut-être responsables de perturbations via des réflections.

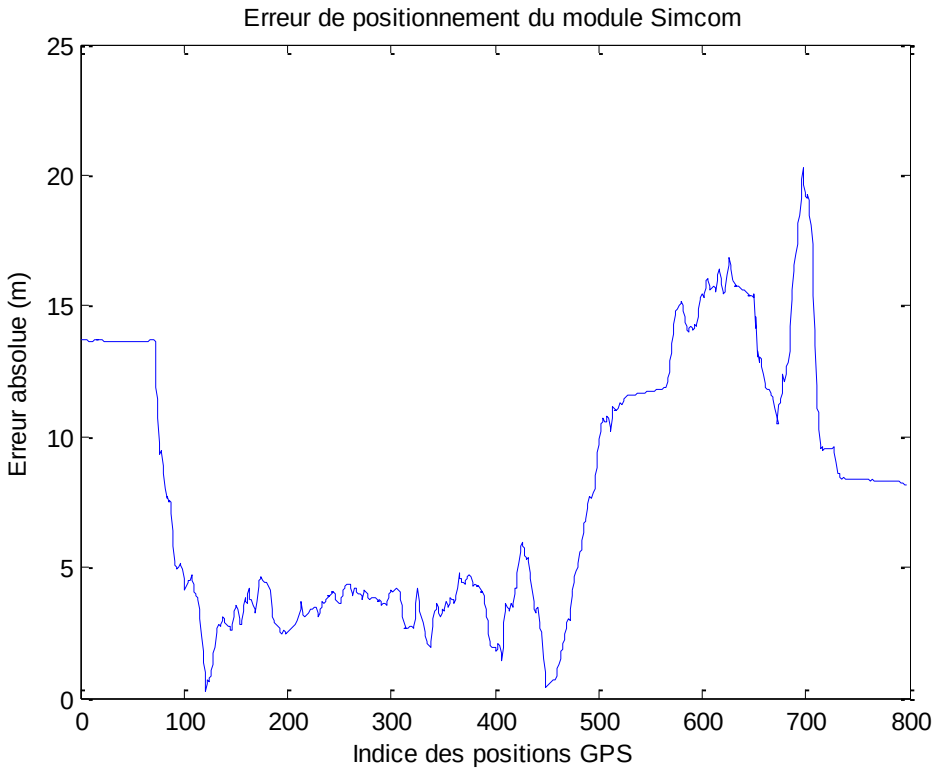


Figure 17. Erreur de positionnement du module SIM5218

La figure ci-dessus représente l'erreur absolue de positin entre les deux GPS. A partir des 800 échantillons mesurés, nous obtenons une moyenne de 7.68 mètres et un écart-type de 4.93.

Ces résultats ne permettent pas, à priori, de valider la fonction d'acquisition de la postion GPS. Néanmoins, dans des conditions normales d'utilisation, nous estimons que notre module de positionnement apporte une réelle plus-value vis-à-vis de la précédente solution basée sur un smartphone.

3. Validation de l'envoi des coordonnées avec le SIM5218

Pour travailler de pair avec le script fourni en annexe 4, nous avons créé un prototype d'application CGI en Python qui récupère les coordonnées GPS envoyées automatiquement par le SIM5218 à travers une requête HTTP GET, les formate, et les affiche sur une carte Google Maps.

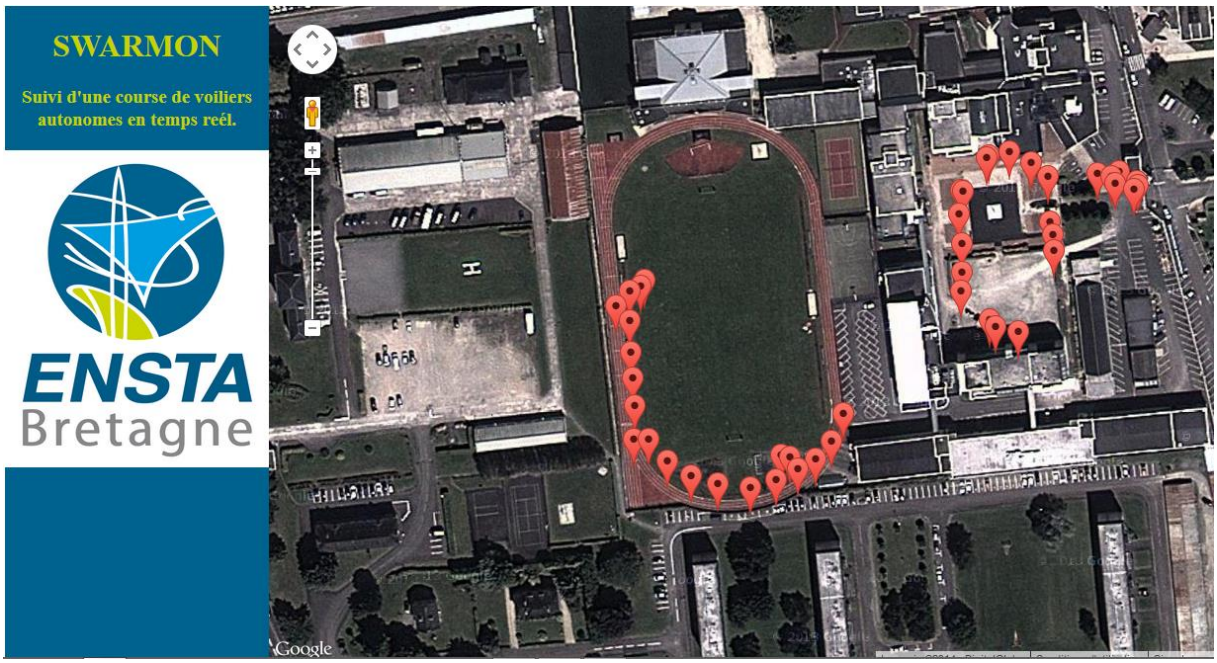


Figure 18. Coordonnées envoyées et affichées automatiquement lors des tests sur le campus

Ce script a permis de valider différents points :

- Lecture de la position GPS par le SIM5218
- Envoi d'une requête HTTP GET avec le SIM5218
- Exécution d'un script LUA exploitant le GPS et la 3G sur le SIM5218
- Automatisation de l'initialisation et du fonctionnement du SIM5218
- Capacité à transformer les données GPS fournies par le SIM5218 en points Google Maps

En dehors de l'authentification du tracker et de l'envoi d'alertes au drone, ce prototype a permis de valider toutes les fonctions de communication entre le SIM5218 et le serveur.

4. Page Web

Dans le cadre des fonctions *FP3.2 - Permettre un affichage temps réel* et *FP3.4 – Afficher une IHM graphique*, nous avons créer un page web permettant de visualiser les positions du tracker sur une carte générée par ©Google Maps.

Après avoir élaboré l'arborescence du site web et la structure des pages, nous avons décidé d'utiliser un design libre, disponible sur un site en ligne et de le modifier pour répondre à nos besoins. Voici un aperçu de la page web (.html) et de la feuille de style en cascade (.css) à l'aide d'un navigateur web.

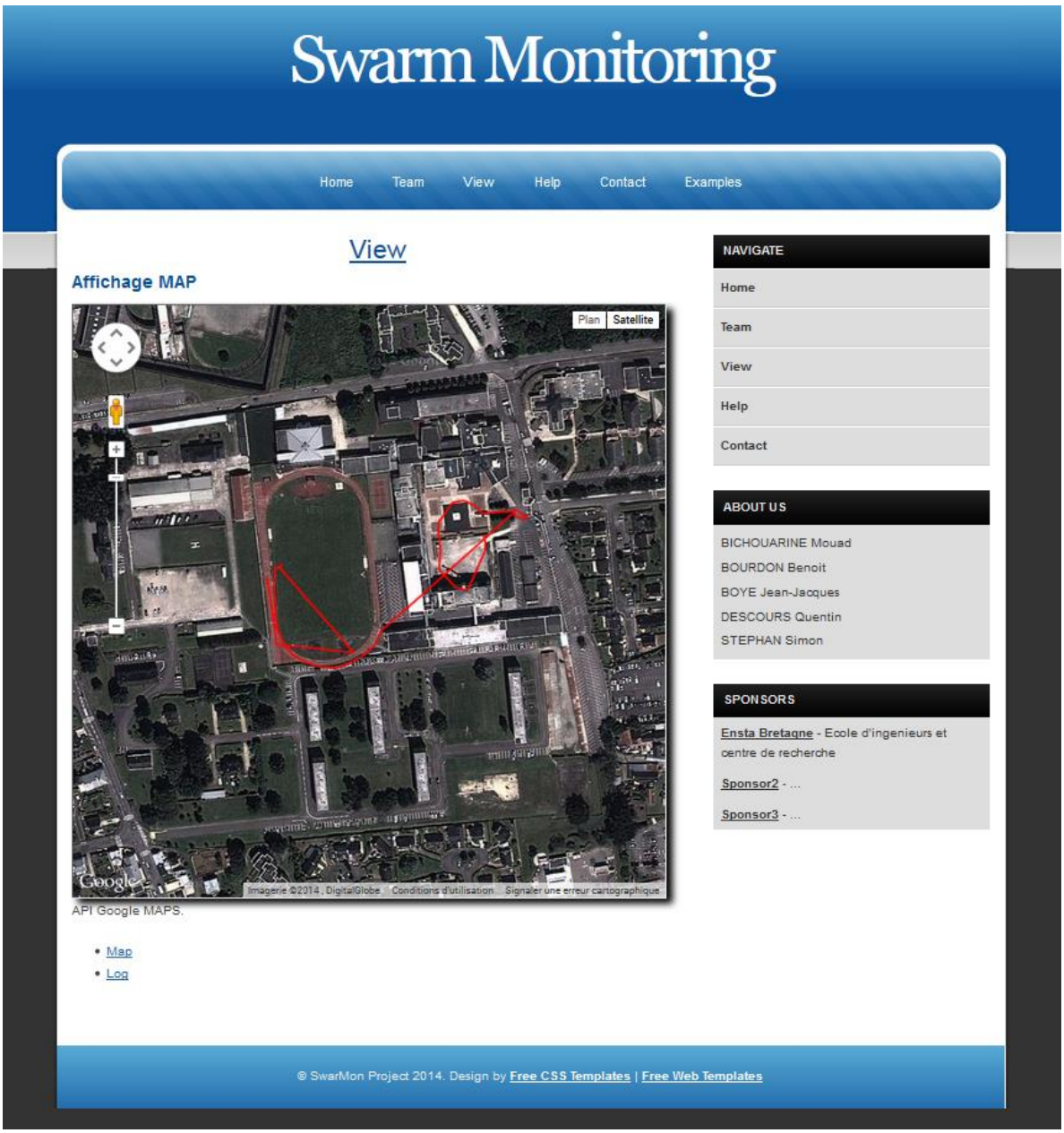


Figure 19. Aperçu de la page web

Au centre, nous avons inséré une carte qui affiche automatiquement les données issues des différents tests effectués tout au long du développement.

A l'aide d'un script python générant des positions aléatoires, nous avons également pu tester le rafraichissement automatique de la carte. Le code Javascript présent permet de générer en temps réel le trajet emprunté par le tracker sans recharger la carte ©Google Maps. Cette astuce permet de rafraichir automatiquement la page web toutes les 10 secondes. Le script est capable de rafraichir la carte toutes les 2 secondes sans dégrader l'expérience utilisateur.

5. Authentification

Le site étant accessible à tous, il semble nécessaire de créer un système d'authentification pour protéger la base de données. C'est le rôle de la fonction 2.5. Pour cela, chaque membre est authentifié par son adresse e-mail et un mot de passe et les trackers sont identifiés par des tokens. Afin d'éviter les conflits, il est impossible de créer deux membres avec la même adresse e-mail. Pour chaque action sur l'application web, le contrôleur vérifie si l'utilisateur a le droit d'exécuter ces actions.

Par exemple, dans le contrôleur de la page member, nous avons ajouté la ligne :

```
before_filter :authenticate_admin, :except => [:index, :show]
```

Cette ligne signifie qu'il faut être authentifié en tant qu'administrateur pour effectuer une action autre que l'affichage de l'index ou les informations d'un membre. La fonction `authenticate_admin` est disponible dans application_controler.rb.

```
1    class ApplicationController < ActionController::Base
2      # Prevent CSRF attacks by raising an exception.
3      # For APIs, you may want to use :null_session instead.
4      protect_from_forgery with: :exception
5      protected
6      def authenticate_admin
7        authenticate_or_request_with_http_basic do |email, password|
8          if m=Member.find_by_email(email)
9            true if (m.authenticate(password) && m.role == "administrator")
10         else
11           false
12         end
13       end
14     end
15   end
```

Figure 20. Classe ApplicationController

La ligne 9 est intéressante. C'est un test qui n'est valide que si le mot de passe est valide et si le rôle de l'utilisateur est administrateur. L'utilisation de la fonction `authenticate` n'est pas anodine. Elle est disponible grâce à l'installation du gem bcrypt. En effet, sans l'utilisation de cette gem, le mot de passe enregistré par l'utilisateur serait disponible en clair dans la base de données. La lecture de la classe Member du modèle de la base de données permet de voir la puissance du gem bcrypt.

```
1    class Member < ActiveRecord::Base
2      belongs_to :team
3      validates :name, presence: true, length: { in: 3..50,
4        too_long: "%{count} characters is the maximum allowed",
5        too_short:"%{count} characters is the minimum allowed"  }
6        has_secure_password
7      validates :password, presence: true, length: { in: 6..50,
8        too_long: "%{count} characters is the maximum allowed",
9        too_short:"%{count} characters is the minimum allowed"  }
10     validates :role, inclusion: { in: %w(administrator visitor player),
11       message: "%{value} is not a valid size" }, allow_nil: true
12     validates :email, presence: true, uniqueness: true, on: :create
13     validates :team,  presence: true
14
15   end
```

Figure 21. Classe Member

C'est la ligne 6 qui nous intéresse. Le simple ajout de `has_secure_password` permet à l'application, lors de la création du mot de passe, de crypter celui-ci, de l'enregistrer en tant que password_digest et de rendre disponible la fonction `authenticate` de la classe ApplicationControleur. Dorénavant, il n'est plus possible de récupérer les mots de passe dans la base de données, seulement le password_digest qui ressemble à cela :

```
$2a$10$QIFk4ytMIzE03/njtSMFmedzhTyv8DVMMtWjqnFeW9FcQpBEf.u0
```

6. Estimation du coût de développement du projet SWARMON :

Voici une estimation des coûts en matériels et main d'œuvre de notre projet. La fonction FP4.1 - Maitriser les coûts fixe un prix de 300€ pour le prototype du tracker et de 200€ pour la version finale. Cette fonction ne concerne que la conception matérielle du boitier. Le coût de développement de la partie logicielle et la prise en main du matériel n'est pas prise en compte. Elle apparaît ci-dessous à titre indicatif.

- **Formation RoR** : 2000 €HT/personne pour 4 jours
  - *Source* : Annonce Humancoders.com

- **Temps de travailpar étudiant**: 5 étudiants
  - Projet : 140 H
  - Ateliers :
    - 9 H RoR
    - 6 H Android
    - 9 H ARMadeus
    - 6 H Carte électronique

- **Achat de matériel** :
  - Module 3G/GPS :
    - Prix unitaire : 52€ HT
    - Prix total : 248.77€ TTC
    - *Source* : Devis ENSTA Bretagne MODEM 3G
  - Carte de développement :
    - Prix unitaire 95€ HT
    - *Source* : Fiche Farnell
  - Batterie :
    - Prix unitaire : 13$ HT
    - *Source* : Fiche Sparkfun

- **Développement d'une carte électronique :**
  - Carte électronique :
    - Prix : 99€ HT
    - Prix total : 3 850€ TTC
    - *Source* : Devis PROTECNO
  - Composants électronique :
    - Prix : 10 € (estimation)
    - *Source* : /

Finalement, le coût de la partie matérielle d'un prototype nous revient à 157€ HT avec la carte de développement et 171€ HT avec la carte provenant de chez ProTecno. La CAO étant déjà réalisé, ces frais n'ont pas été inclus dans le prix, et des économies d'échelle peuvent être réalisées avec des commandes en grande quantité de la carte personnalisée.

7. Conception de la carte électronique

La conception du module 3G / GPS a été réalisée à partir de la carte de développement fournie par le fabricant. Afin d'abaisser le coût d'un tracker et d'optimiser l'espace occupé par cette carte, nous avons décidé de concevoir une nouvelle carte électronique par nos propre moyens.

Pour élaborer cette carte, nous avons besoin de disposer des interfaces suivantes :

- Connecteur pour le SIM5218 ;
- interface µUSB ;
- slot pour carte µSD ;
- slot pour carte SIM.

De plus, la carte doit disposer de LEDs, de switchs et boutons pour la mise sous tension et l'activation des transmissions. La prise USB doit également permettre de communiquer avec le robot (lien série) et de recharger la batterie via un module complémentaire.

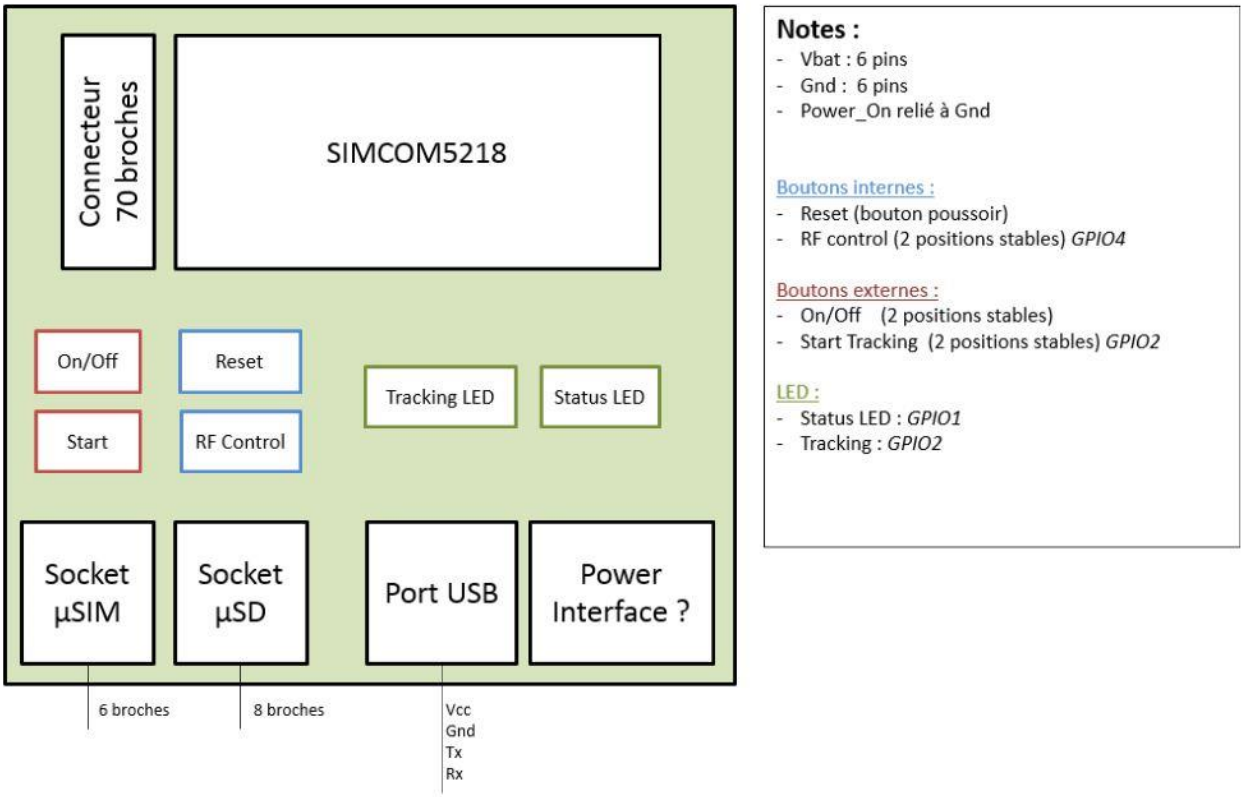


Figure 22. Schéma de la carte électronique

Voici un schéma de principe représentant l'allure générale de la carte électronique. La figure ci-dessous représente le schéma électrique associée. Le schéma a été réalisée avec l'aide de M. Le Maillot.

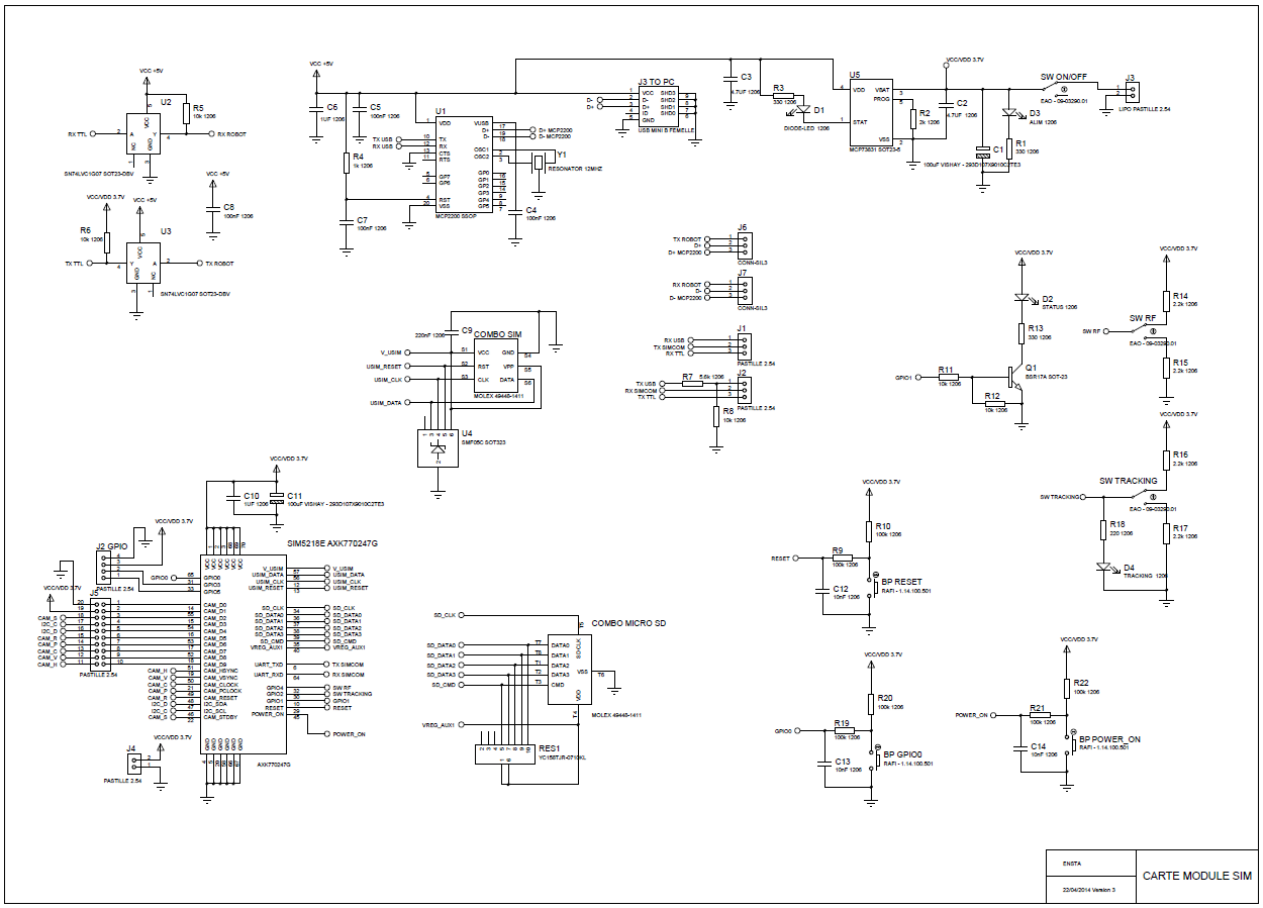


Figure 22. Schéma de la carte électronique par ProTecno

Une fois la conception achevée, nous avons contacté l'industriel ProTecno, spécialisé dans la fabrication de cartes électroniques. L'industriel nous fournira ainsi les PCBs quatre couches avec les composants additionnels installés et des dimensions proches de celles des batteries utilisées.

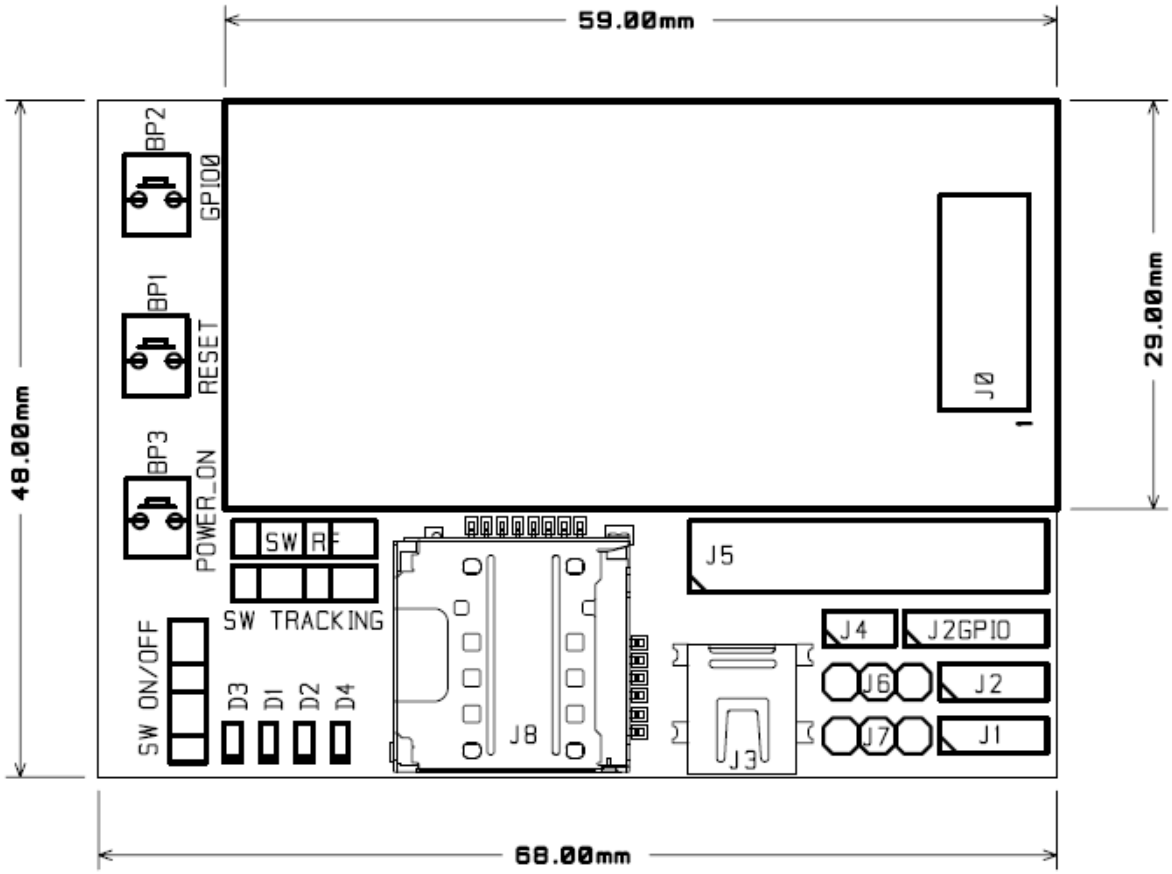


Figure 23. Carte en cours de production par ProTecno

Conclusion

Après 8 mois de travail sur ce projet, une conclusion s'impose : le développement d'un produit « fini » est d'une complexité extraordinaire. Entre la prise en main de technologies complètement inconnues (électronique générale, Ruby on Rails, …), la gestion du temps de développement et des délais d'approvisionnement et la validation des solutions, de nombreux imprévus ont surgi tout au long du projet.

Aujourd'hui, toutes les bases semblent être posées pour obtenir un produit pleinement fonctionnel, mais il reste une quantité de travail non-négligeable à fournir pour présenter un produit utilisable lors de la WRSC en septembre 2014.

Ce projet aura été pour nous une occasion de comprendre les contraintes et la complexité de la conception d'un produit dans l'industrie, et aura donc été une expérience extrêmement formatrice, tant sur le plan de la technique que de la gestion de projet et d'équipe ou de l'ingénierie système.

Références

[1] HelicoMicro, Test du tracker GPS TK102, 2013
http://www.helicomicro.com/tracker-gps-tk102-le-test/

[2] Wikipédia, List of single-board computers, 2014
http://en.wikipedia.org/wiki/List_of_single-board_computers

[3] Fait Main Magazine, XBee&Arduino, 2013
http://faitmain.org/volume-2/xbee-arduino.html

[4] Digi International, XBee Range Extenders, 2013
http://www.digi.com/products/wireless-modems-peripherals/xbee-range-extenders/

[5] BeagleBone Black, 2013
http://beagleboard.org/products/beaglebone%20black

[6] Arduino, 2013
http://arduino.cc/

[7] Olivier Reynet
https://eportfolio.ensta-bretagne.fr/user/view.php?id=17

[8] Michael Hartl, Tutoriel Ruby on Rails, 2010
http://french.railstutorial.org/chapters/beginning

[9] Bruno Aizier, Ingénierie Système, 2013
https://moodle.ensta-bretagne.fr/mod/folder/view.php?id=24368

[10] Wikipedia, Application Programming Interface, 2014
http://en.wikipedia.org/wiki/Application_programming_interface

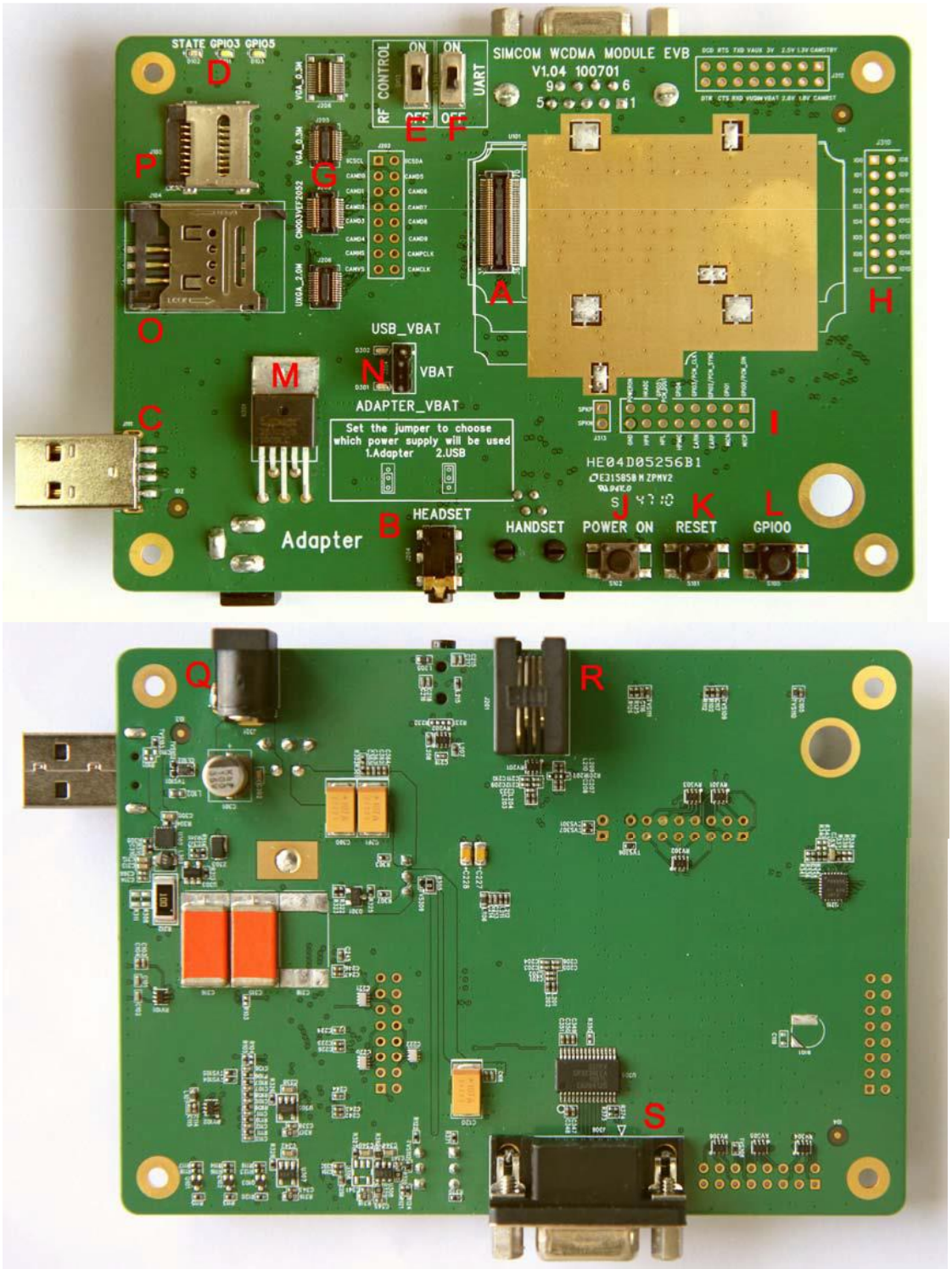
[11] Nicolas Brocheton, API SWARMON, 2013
haggis.ensta-bretagne.fr.3000

Annexe 1 : Lexique

- **Tracker :** Dispositif permettant de suivre un objet ou une personne. Selon le contexte nous utiliserons indifféremment les termes «tracker», «système embarqué» et «boitier étanche» pour désigner la même entité physique.
- **Android :** Système d'exploitation OpenSource développé par Google basé sur le noyau Unix pour smartphones, tablettes et PDA exécutant des programmes prévus pour la plateforme Java grâce à la machine virtuelle Dalvikintégrée au système (et non via la machine virtuelle Java fournie par Oracle).
- **PHP** *HypertextPreprocessor* :Langage de programmation libre, compilé à la volée principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP3, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale.
- **MySQL :** Système de gestion de base de données (SGBD) relationnelles, utilisant le langage SQL. Il est l'un des plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server.
- **Daemon :** désigne un type de programme informatique, un processus ou un ensemble de processus qui s'exécute en arrière-plan plutôt que sous le contrôle direct d'un utilisateur.
- **Xbee :** Module radio commercialisé par Digi International.
- **Framework :** Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel.
- **3G :** désigne une génération de normes de communication pour la téléphonie mobile. Elle est représentée principalement par les normes Universal Mobile Telecommunications System (UMTS) et CDMA2000, permettant des débits (de 2 à 42 Mb/s définis par la dernière génération des réseaux UMTS : l'HSPA+ DC) qui sont bien plus rapides qu'avec la génération précédente, par exemple le GSM.
- **BeagleBone Black :** Single Board Computer utilisant un processeur ARM. De la taille d'une carte de crédit, il peut être utilisé pour la robotique et l'embarqué grâce à ses nombreuses interfaces, sa puissance de calcul, et la possibilitéd'utiliser un Linux standard.
- **Arduino :** Famille de circuits imprimés embarquant un microcontrôleur, partiellement libres (les plans de la carte elle-même sont publiés en licence libre mais dont certains composants sur la carte, comme le microcontrôleur par exemple, ne sont pas en licence libre) qui peuvent être programmés pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme la domotique (le contrôle des appareils domestiques - éclairage, chauffage…), le pilotage d'un robot, etc... exploitant des entrées/sorties simples
- **Ruby on Rails :**abrégéRoR ou Rails,framework web libre écrit en Ruby. Il suit le motif de conception Modèle-Vue-Contrôleur (MVC). En tant que framework, il propose une structure au programmeur qui lui permet de développer plus vite et plus intuitivement. Il ajoute aussi un grand niveau d'abstraction dans la programmation de l'application par un ensemble de fonctions de haut niveau qui lui offre ainsi l'économie d'écrire lui-même la plupart des routines obligatoires d'une application web.

- **Gem :** Un format de paquet du gestionnaire de paquets RubyGems pour la bibliothèque de langage de programmation Ruby.
- **Linux pour l'embarqué :** désigne un système d'exploitation basé sur Linux et adapté à un système embarqué.Contrairement aux versions de Linux destinées aux ordinateurs personnels et aux serveurs, les différents systèmes Linux embarqués sont adaptés spécifiquement pour des systèmes aux ressources limitées.
- **Sprint :** Petite tâche précise que l'on essaye de réaliser en un temps réduit.
- **Protocole NMEA :** Une spécification pour la communication entre équipements marins dont les équipements GPS.

Annexe 2 : Extrait de documentation de la carte de développement



Légende :

A: SIM521x module interface

B: Headset interface

C: USB interface

D: GPIO led

E: RF control switch (Before the SIM521x is powered on, please make sure that RF control
switch is ON)

F: Serial port shutdown switch (If one wants to use UART, please switch it to ON at first.)

G: Camera interface

H: I_2C expand IO interface

I: Audio & GPIO test interface

J: Power on/off of SIM521x

K: RESET key

L: GPIO0 key

M: LDO MIC29302

N: Power select jumper

O: SIM card interface

P: SD card interface

Q: DC power in

R: Handset interface

S: UART interface for AT command transmitting, data exchanging

Annexe 3 : Configuration et utilisation d'un module SIMCOM 5218

Démarrage pour la configuration du SIMCOM avec la carte de développement

- Insérer une carte SIM, **impossible de démarrer sans SIM**
- Switchs UART (port série) et RF Control (mode avion) en position **ON**
- Relier le port série à un PC (via l'adaptateur USB par exemple)
- Alimenter la carte (USB ou secteur), **positionner le cavalier d'alimentation** selon la source choisie
- Démarrer le module grâce au bouton POWER ON

Accès au terminal série via PuTTY

- `dmesg | grep tty` pour identifier le port série à utiliser
- démarrer PuTTY, *Hostname : /dev/ttyUSB0* (adapter selon dmesg), *Speed : 115200*, *Connection type : serial*, *Open*
- `ati` puis "entrée" pour tester la communication avec le module

Accès au terminal série via Python Serial (pas de lecture du port)

```
> python

import serial

ser=serial.Serial('/dev/ttyUSB0',115200)

ser.write('ati \r\n')
```

Configuration de l'accès internet

Liste de configurations PDP Pour Orange (testé sur Mobicarte):

```
at+cgsockcont=1,"IP","orange.fr"        #définition du point d'accès

at+csockauth=1,1,"orange","orange"        #configuration de l'authentification

at+csocksetpn=1                            #profil à utiliser (16 profils peuvent être
stockés sur le SIMCOM)
```

Pour Bouygues (testé sur B&You):

```
at+cgsockcont=2,"IP","mmsbouygtel.com"

at+csockauth=2,0

at+csocksetpn=2
```

And it works !

Commandes de test pour poster un visiteur sur le prototype CGI sur Haggis

```
ser.write('at+netopen=,,1 \r\n')            #ouverture de la connexion

ser.write('at+chttpact="haggis.ensta-bretagne.fr",80 \r')

ser.write('GET http://haggis.ensta-bretagne.fr/cgi-bin/getapp.py?visiteur=blablabla
HTTP/1.1\r\nHost: http://haggis.ensta-bretagne.fr\r\nContent-length:0\r\n\r\n\r\n \x1A')

ser.write('at+netclose \r\n')                #fermeture de la connexion
```

Utilisation des scripts LUA

Lancement d'un script depuis la carte SD

```
at+cscriptstart="D:\scritp.lua"
```

Arrêt du script

```
at+scriptstop
```

Copier un script vers la mémoire du SIMCOM (nommer le script *autorun.lua* pour qu'il s'exécute au démarrage)

```
at+fscopy="D:\script.lua","C:\script.lua"
```

Annexe 4 : Script Lua - Envoi des coordonnées GPS du tracker au serveur.

```
printdir(1)   # Permet de visionner les print en console

sio.send('at+netopen=,,1\r\n')  # Connexion à Internet
waitevt(100)   #Les waitevt() permettent l'attente de la fin du
traitement de la commande précédente.
waitevt(5000)
sio.send('at+cgps=1\r\n')# Démarrage du GPS
waitevt(1000)

while true do
rst= gps.gpsinfo();    # Récuperation des coordonées GPS
print(rst,"\r\n");

cmd1='at+chttpact="haggis.ensta-bretagne.fr",80 \r'
sio.send(cmd1);# Envoi de la commande pour connexion à Haggis
rtc1=sio.recv(1000)
waitevt(700)
print(rtc1)
str2='GET http://haggis.ensta-bretagne.fr/cgi-
bin/getapp.py?visiteur='
str10='jjenbalade'
str11='&gps='
str3=' HTTP/1.1\r\nHost: http://haggis.ensta-bretagne.fr\r\nContent-
length:0\r\n\r\n\r\n'
str4=string.concat(str2,str10)
str12=string.concat(str4,str11)
str13=string.concat(str12,rst)
str5=string.concat(str13,str3)
cmd2=string.concat(str5,string.char(0x1A))
print(cmd2)
sio.send(cmd2);# Envoi de la commande pour envoi des coordonnées
rtc2=sio.recv(10000)
waitevt(100)
print(rtc2)
end
```

Annexe 5 : Plan de validation fonctionnelle

Test unitaire

	Tests unitaires	Description	Avancement	Test	Conclusion
Module SIM	Formater les données GPS	Script lua pour récupérer les coordonnées GPS.	OK	Test autour du stade, affichage sur Google Maps pour vérifier le fonctionnement	Fonctionne
Module SIM	Stocker les données	Script lua pour stocker les données sur la carte SD.	/		
Module SIM	Communiquer en 3G	Requête http vers le serveur haggis pour transmettre les données.	OK	Réception de la requête avec un CGI Python	Fonctionne
Module SIM	Etudier la précision du GPS	Comparer les données GPS avec une centrale GPS Différentiel.	OK	Test autour du stade	~ 8m Supérieur à 5m, non valide
Batterie	S'adapter à l'énergie	Tester le Module de développement avec une batterie.	OK	Alimentation directe avec batterie 3.7V/2000mAh de la carte de dev	Fonctionne
Batterie	Estimer la consommation en charge	Mesurer la consommation du Module en pleine charge.	OK	Mesure ampèremètre+voltmètre de l'alimentation de la carte de dev	0.6W
Batterie	Estimer l'autonomie en charge	Mesurer l'autonomie du Module en pleine charge.	/		

Lien série	Alerter le tracker	Envoyer un message du serveur vers le tracker.	/		
Lien série	Alerter le drône	Envoyer un message du tracker vers le drône.	/		
HTML/CSS	Créer un IHM graphique	Vérifier l'ergonomir du site.	OK	http://haggis.ensta-bretagne.fr/site/home.html	Valide
Javascript	Intégrer Google Maps	Intégration de l'API Google Maps v3 dans une page web.	OK	http://haggis.ensta-bretagne.fr/cgi-bin/mapappview.py	Valide
Javascript	Rafraichir la page web	Afficher les nouvelles positions sur la page web sans recharger la carte Google Maps	OK	http://haggis.ensta-bretagne.fr/site/view.html	Valide
Boitier	Vérifier l'étanchéité	Vérifier le bon fonctionnement dans différentes conditions météo.	/		
RoR	remplir la BDD	vérifier fonctionnement de la BDD / création de membres, de robots, d'essais)	ok	utilisation d'un script	Valide
RoR	Validation	contrôle de présence et de conformité de certains parametre dans les tables de la BDD	OK	Les validations ajoutées jusqu'à présent fonctionnent, reste à optimiser le format de l'adresse mail par exemple	En cours d'amélioration
RoR	Création de sessoin	Permettre aux administrateurs et aux leaders d'équipes de valider les inscriptions et la création des équipes		Les views sont mises en places reste à définir correctement les controlleurs et optimiser les modèles	En cours d'amélioration
RoR	Réalisation des Callbacks	Effacer le contenu de tables associés à d'autres lors de leurs suppression	En cours		

Test d'intégration

	Tests d'intégration	Description	Avancement	Test	Conclusion
Module SIM	Vérifier la chaine de transmission	Récupérer la position GPS et l'envoyer au serveur via Internet(3G)	OK	Tour de stade envoyé en autonomie par un script LUA vers un CGI Python	Fonctionne
Module SIM	Remplacer la carte de développement	Vérifier le bon fonctionnement du tracker avec notre carte électronique perso.	/		
Module SIM	Valider le module complet	Tester la transmission + stockage des données + alerte	/		
Batterie	Tester l'autonomie du tracker	Vérifier l'autonomie énergétique du tracker	OK		
Lien série	Communiquer	Vérifier la chaine d'alerte complète	/		
Page web	Intégrer la page web dans RoR	Générer les pages web àl'aide de RoR	/		
Javacript	Récupérer les données dans la BDD	Récupérer la position GPS dans la base de données	/		

Chaine fonctionnelle	
Chaine 1 (Tracker)	Localiser la position du drone (< 5m)
	Formater les données GPS
	Stocker les données dans la mémoire interne
	Sécuriser l'accès aux données
	Communiquer les positions au serveur par le réseau 3G.
Chaine 2 (Serveur)	Stocker les données dans la BDD
	Afficher les positions à travers un IHM graphique
	Suivre l'évolution en temps réel (< 10s)
Chaine 3 (Alerte)	Évaluer le risque de collision
	Alerter le tracker
	Communiquer une alerte au drone
Chaine 4	S'adapter à l'énergie disponible
	S'adapter au drone
	Résister au milieu marin