Conception

Ce projet est organisé autour d'entités hétérogènes qui communiquent entre elles en réseau Wi-Fi grâce à un protocole que nous avons mis en place.
Dans ce chapitre seront détaillés le protocole en question ainsi que l'architecture réseau adoptée.

1 - Protocole

Il existe trois différent rôles, le serveur, les ancres, et les mobiles.
Chaque noeud du réseau peut tenir simultanément un ou plusieurs d'entre eux,
la seule contrainte étant que le serveur doit être unique.
La communication s'effectue de manière centralisé en passant par le noeud qui tient le rôle de serveur et en utilisant un format de message spécifique détaillé dans la documentation technique.

Nous allons maintenant présenter en détail se que font chacun des acteurs du réseau.

1.1 - Le Serveur

Le serveur est le noeud central du réseau, il a pour objectif de permettre la communication entre les autres noeuds. Pour cela, il doit donc attribuer à chaque noeud un identifiant unique, permettre la retransmission des messages et effectuer un recensement dynamique des noeud connectés au réseau ainsi que de leurs rôles respectifs. De plus, le serveur garde en mémoire les logs envoyé par chaque mobile et permet leur sauvegarde.

À son lancement, le serveur lance une boucle d'acceptation des client et attends une nouvelle connexion. Chaque nouveau client est gérer dans un fils d'exécution différent de la manière suivante.

En premier lieu, le client est informé de son identifiant propre, le serveur envoi donc un message au client et attends une confirmation de réception de celui-ci.
Cette étape, ainsi que la suivante, sont répétés un nombre arbitraire de fois. Si tout les essais se solde par un échec, la communication est interrompue.

Une fois que le client connais son identifiant, le serveur lui demande son type, c'est à dire les rôles qu'il rempli, puis, après réception de la réponse, il met à jour ses annuaires de mobiles et ancres.

Une fois ces deux étapes terminées, la boucle de communication est lancée. Le serveur vas attendre que le client lui envoi un message, et selon s'il en est le destinataire, il le traitera, ou le retransmettra à son destinataire. Les client peuvent donc de cette manière communiquer directement avec le serveur. Cela leur donne la possibilité de demander au serveur la liste des ancres connectées, de lui envoyer un log ou bien d'annoncer leur sortie du réseau.

1.2 - L'Ancre

Une ancre est un noeud qui connait sa position et qui est capable de fournir à un mobile une évaluation de la distance qui les séparent.

À son lancement, l'ancre va se connectée au serveur et attendre de recevoir son identifiant. Ensuite, elle attendra que le serveur lui demande son rôle et répondra.
Une fois la boucle de communication lancée avec le serveur, l'ancre attendra qu'un message lui parvienne d'un mobile. Les ancres ne traite qu'un seul type de message, les demandes d'état. Elle y répondent en envoyant un message qui contient leur position ainsi qu'une évaluation de distance obtenue grâce à leurs capteur ultrason.

1.3 - Le Mobile

Le mobile est le noeud qui cherche à estimer position. Il vas pour cela utiliser le serveur afin d'obtenir les identifiants des ancres alentour, puis leurs demandera d'évaluer la distance qui les séparent en continu, afin d'estimer au mieux sa position.

Le mobile se comporte à son lancement comme une ancre, il se connecte au serveur, récupère son identifiant et informe le serveur de son type.
Il demande ensuite au serveur de lui transmettre la liste des ancres jusqu'a en obtenir un nombre qu'il juge suffisant (3 dans nos tests). Une fois la liste obtenu, le mobile vas demander à chaque ancre de lui transmettre sa position.
Le mobile initialise alors un AVT pour chacune des ancres ainsi que deux autres, pour chacune des composante de sa position (on rappel que le mobile évolue en deux dimensions). Le mobile commence alors à chercher sa position en itérant la démarche suivante.

Il demande d'abord successivement à chacune des ancres de lui transmettre une évaluation de la distance qui les séparent et met à jour à chaque fois les valeurs de

leurs AVTs respectifs. Ensuite, il effectue une opération de trilatération pour obtenir une nouvelle approximation de sa position, et enfin il utilise se résultat pour mettre à jour l'AVT correspondant à sa position.
Dans le cas ou une ancre quitte le réseau ou ne répond plus, le mobile redemande la liste des ancres au serveur jusqu'a en avoir à nouveau un nombre satisfaisant.
Il arrête à ce moment là de calculer sa position mais continue à mettre à jour les AVTs des ancres encore disponible pour pouvoir se repositionner ensuite au plus vite.

Pour finir, à chaque itération du processus, le mobile envoie un message « log » au serveur qui contient les valeurs courante des AVTs de sa position et des distance avec les ancres qu'il interroge.

2 - Réseau

2.1 - Première solution envisagée

Pour répondre aux exigences du projet, une première solution en Ad-Hoc a été étudiée. Celle-ci devait permettre la création d'un réseau de communication entre les différents noeuds qui serait robuste face aux entrée et sorties de ces derniers dans le réseau en question. Cette solution aurait eût de plus la particularité de s'affranchir d'un serveur central, lui offrant d'autant plus de robustesse et lui permettant de s'établir sur un périmètre plus grand.

La réalisation d'un tel réseau avec le matériel proposé implique une conception divisée en deux parties, une première sur Raspberry Pi, et une seconde destinée à l'Arduino et au module ESP.

2.1.1 - Raspberry Pi

Le déploiement d'un réseau Ad-Hoc sur Raspberry est relativement simple.
Il faut tout d'abord connecter un dongle Wi-Fi compatible avec le mode Ad-Hoc à la Raspberry Pi. Ensuite, il faut de modifier le fichier '***/etc/network/interfaces***' passer en mode Ad-Hoc ainsi que pour donner à la Raspberry une adresse IP unique et lui indiquer le nom du réseau qui sera utilisé.

2.1.2 - Arduino et ESP8266

La mise en place du réseau ad-hoc sur Arduino et le module ESP8266 est par contre plus complexe. Pour pouvoir y arriver, il faut d'abord réussir à faire communiquer correctement l'Arduino avec l'ESP et ensuite faire en sorte que l'ESP se connecte au réseau.

Toute communication entre l'Arduino et le module ESP passe par l'envoi et la réception de commandes textuelles. Ce mode de communication est très contraignant car il oblige à gérer de multiples erreurs et délais. Pour le simplifier, et pour le rendre plus rapide, nous avons mis en place une interface entre les deux appareils. Celle-ci est basé sur la librairie WeeESP8266 trouvé en ligne. Cette librairie étant implémentée pour fonctionner avec un autre modèle d'Arduino, nous l'avons simplement adaptée au nôtre.

Une fois l'intégration de l'ESP correctement effectuée, il faut faire en sorte que celui-ci se connecte au réseau Ad-Hoc.
Le module ESP propose trois modes de fonctionnement, le mode « station », le mode « access point » et enfin le mode « both » qui permet d'activer les deux premiers modes simultanément.
Le protocole Ad-Hoc n'est donc pas pris en charge nativement.
Deux solution sont alors possible.
Modifier l'ESP pour lui permettre de rejoindre un réseau Ad-hoc déjà existant, et en créer un dans le cas contraire.
Ou bien, simuler un réseau ad-hoc en utilisant les mode de communication déjà existant de l'ESP.

[MESHISH - Ad-hoc pas possible]

2.1.3 - Solution Retenue

À cause de la difficulté et du temps nécéssaire à la réalisation d'un réseau Ad-Hoc couvrant à la fois les Raspberry et les Arduino, et dans l'optique d'assurer la réalisation des tests en respectant les délais imposés par le projet, le réseau Ad-Hoc n'as finalement pas été retenu.
À la place, une architecture Client - Serveur classique à été adopté. Bien qu'étant, comme exprimée plus haut, moins robuste et plus contraignante à utiliser, elle possède l'avantage d'être simple et facile à mettre en place. Une telle architecture nous autorise donc à consacrer plus de temps à la réalisation des tests.