## Wifi

Le wifi est implémenté, sur l’ESP avec le module Gps celui-ci est mis en mode AP (Acces Point) il crée donc le réseau wifi. Les autres ESP sont en mode station ils se connectent juste sur le réseau wifi créé par l’AP (à savoir que l’esp est de base en mode station).

Le SSID étant testRes et le mot de passe étant test1234 ceux-ci peuvent être changé dans le code (il faut penser à mettre ces données dans un fichier externe).

### Lien Wifi

Très bonne vidéo expliquant la mise en place du wifi :

https://www.youtube.com/watch?v=U4GaKrsXlPc&ab\_channel=TommyDesrochers

## GPS

Le Gps est toujours bug je soupçonne un souci de pôle Nord / Sud

## AsyncWebServer

Le server web fonctionne sur l’ESP avec le module GPS, celui-ci permet une communication entre les ESP grâce à des points d’entrer que nous pouvons créer. Il est donc possible de demander des infirmations au server ou encore de lui envoyer des informations en attendant un retour.

Exemples de points d’entrer :

Une image contenant texte

Description générée automatiquement

### Lien AsyncWebServer

Lien donnant beaucoup d’info sur l’AsyncWebServer et son implémentation :

https://github.com/me-no-dev/ESPAsyncWebServer#using-platformio

## OTA

L’Implémentation d’OTA est partiellement réussie étant donné que le téléchargement fonctionne les ¾ du temps.

L’OTA fonctionne en envoyant des .bin vers l’ESP. Pour avoir le .bin il faut aller dans l’IDE d’Arduino -> Sketch -> Export compiled binary celui-ci sera probablement créer dans un dossier build qui va se situer dans le dossier ou tu as sauvegardé le. ino

Le code qui permet l’OTA devrait à l’avenir se situer dans un .h ou dans une librairie afin de ne pas surcharger le code principal. A noter que l’on pourrait grâce à l’OTA importer une bibliothèque permettant le remote debug -> voir le lien en bas de la page

Piste pour le faire fonctionner :

* Enlever les différentes applications qui pourraient envoyer des querry dans le réseau et donc surcharger celui-ci
* Ne pas passer par l’ide d’Arduino plutôt préféré la ligne de code - >

espota.py -r -i esp\_ip lien vers le build -d -t 1000 -r -I host\_ip -p port\_esp

Exemple :



espota.py ce situe dans

C:\Users\maxim\OneDrive\Documents\ArduinoData\packages\esp32\hardware\esp32\2.0.3\tools en clair ça doit se situer dans ton dossier d’installation d’Arduino

* Utiliser le port 3232 pour l’esp, par défaut c’est le 8266 mais il ne fonctionne pas pour moi
* Éventuellement faire des pings avant voir si tu sais ping les différents ESP
* Ne pas oublier de se connecter dans le réseau des ESP

### Lien utile sur l’OTA

Documentation basic avec du code :

<https://www.aranacorp.com/fr/programmer-un-esp32-via-wifi-avec-lide-arduino-ota/>

### Alternative OTA

Il existe ElegantOTA qui utilise le principe de server pour mettre à jour l’ESP via un lien style 192.168.4.1/update sachant que le /update est le même peut importe le server il n’y a que l’ip qui change.

Le souci principal étant l’installation d’un server sur chaque ESP qui semble un peu overkill.

### Lien OTAElegant

Monsieur sympathique qui explique comment faire :

https://www.youtube.com/watch?v=OaHzlBTUrRQ&t=662s&ab\_channel=RalphSBacon

## Librairies

Une optimisation et un clear des librairies doit éventuellement être fait dans le code. Eventuellement revérifier les différentes licences des librairies.