RÉALISATION D'UN OBJET CONNECTÉ

Réaliser par :
EL ASRI AYOUB
AIT TALB ALI EL mehdi
El HBOUSSI Souhail

Encadrant :
M. STEPHANE Nicolas

Master 1 IGIS, spécialité GEII
2015/2016

Remerciements :

Avant de commencer la présentation de ce travail, on profite de l'occasion pour remercier toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet ter.

On tient à exprimer nos vifs remerciements pour notre respectueux Professeur, **M. STEPHANE Nicolas**, d'avoir accepté de nous encadrer pour notre projet ter , ainsi que pour son soutien, ses remarques pertinentes et son encouragement.

Nos remerciements vont aussi à tous nos professeurs, enseignants et toutes les personnes qui nous ont soutenus jusqu'au bout, et qui n'ont pas cessé de nous donner des conseils très importants en signe de reconnaissance.

I-Résumé :

L'objectif de ce projet TER est de réaliser un petit objet connecté a internet que l'on pourra localiser et contrôler a distance a l'aide d'un Smartphone ou tablette.

Cet objet sera sous forme d'un porte clef qui est capable de transmettre sa position à son propriétaire et émettre un signal sonore ou faire clignoté à la demande en cas de perte ou éventuellement en cas de vol car dans un petit périmètre(une maison par exemple) le GPS n'a pas la précision requise pour déterminer la position exacte de l'objet.

L'objet connecté fonctionnera avec une application web de type ''responsive''. L'application se chargera de géo-localiser via (Google Map) l'objet recherché , le faire sonner ou encore le faire clignoté, tout ça à l'aide d'un ensemble Arduino Uno + shield GPS + shield wi-fi.

Abstract :

The objective of this project is to realize a small connected object to Internet which we can control from a distance remotely using a laptop, smart phone or a tablet .

This object will take the form of a key Port which is able to send its position to its owner and emit a sound signal or flashing on demand in case of loss or robbery, because in a small perimeter, a house for example the GPS hasn't the required precision to specify the exact position of the object.

The connected object will work with a Web responsive application or mobile application dedicated Android . the application will take care of giving geographic positions on (Google Map) the sought object,ring it or flashing it, all this with a set of Arduino Uno + shield GPS + shield wi-fi.

Table des matières

II. État d'art

Internet des objets (IOT)

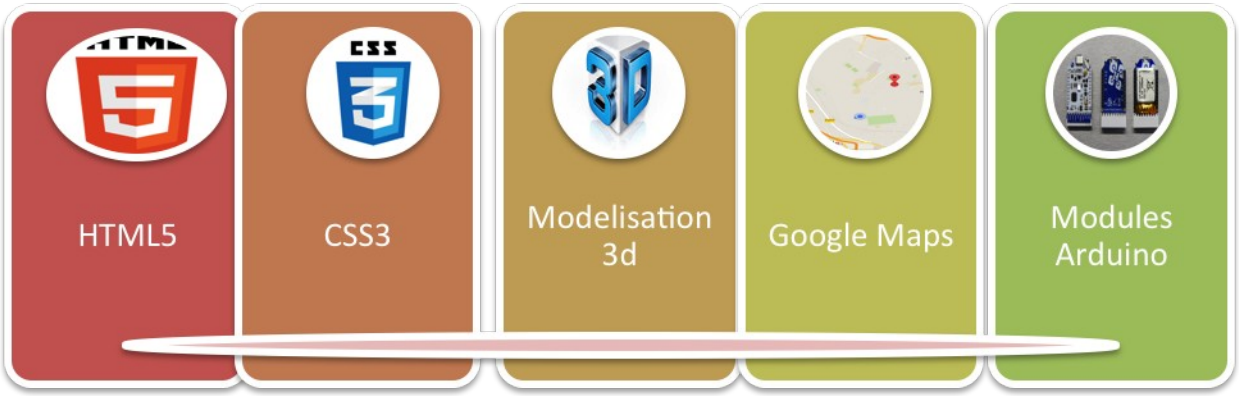
Représente l'expansion d'Internet à des choses/objets et à des lieux dans le monde physique. L'Internet des objets désigne le « mouvement de liaison » d'objets quotidiens ou de nouveaux objets à Internet. Les objets connecté sont reliés a internet on parle d'ailleurs du web 3.0 ses derniers peuvent communiquer avec d'autres systèmes pour obtenir ou fournir de l'information. Cela est rendu possible avec la miniaturisation des composantes électroniques.

l'existant

Wistiki : l'objet connecté pour retrouver ses affaires perdus

Fonctionnalitées :

Outils de développement :



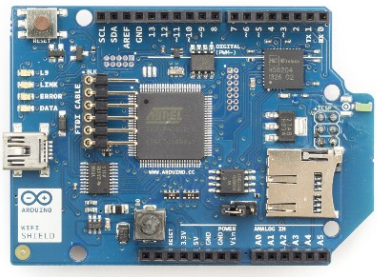
III-Matériel utilisé

Pour réaliser notre objet connecté, on a utilisé :

-Carte Arduino Uno :

La carte Arduino Uno est une carte microcontrôleur basée sur le ATmega328P. Il dispose de 14 broches numériques d'entrée / sortie (dont 6 peuvent être utilisées comme sorties PWM), 6 entrées analogiques, un cristal de quartz 16 MHz, d'une connexion USB, une prise de courant, un tête ICSP et d'un bouton de réinitialisation.

-Shield wi-fi :



Le Shield wi-fi permet à une carte Arduino de se connecter à Internet en utilisant la spécification sans fil 802.11 (Wi-Fi). Il est basé sur le HDG204 système 802.11b / g sans fil in-Package. Un AT32UC3 fournit un réseau (IP) empiler capable à la fois TCP et UDP

Shield GPS :

Ce shield GPS est fourni totalement assemblé et ne nécessite pas d'antenne pour fonctionner. Il est compatible avec les cartes Arduino UNO, Duemilanove, Leonardo et Mega. Ce shield GPS utilise le récepteur SUP500F de Skytraq. Ce récepteur GPS possède 65 canaux à 10hz ainsi qu'une antenne intelligente intégrée. Une interface série 3V LVTTL est disponible.
Référence : A-000000-00141

IV-Travail effectué

1) Phase de conception

Diagramme de Cas d'utilisation :

Le diagramme de cas d'utilisation nous décrit, sous forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur et nous permet de définir les limites du système et ses relations avec l'environnement.

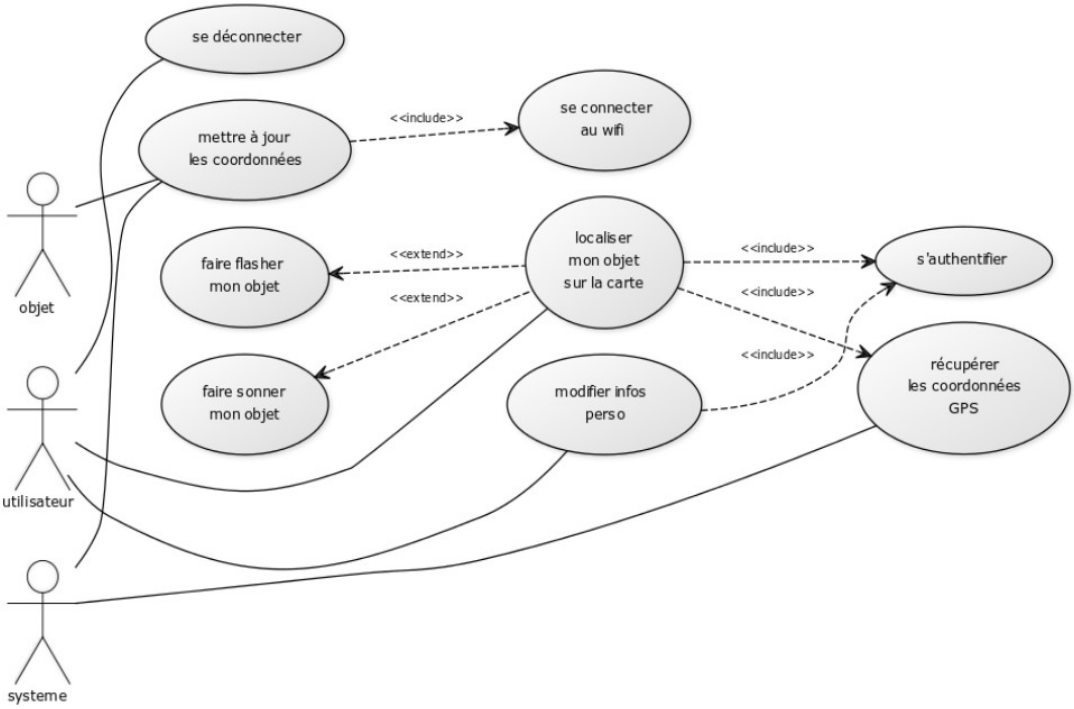


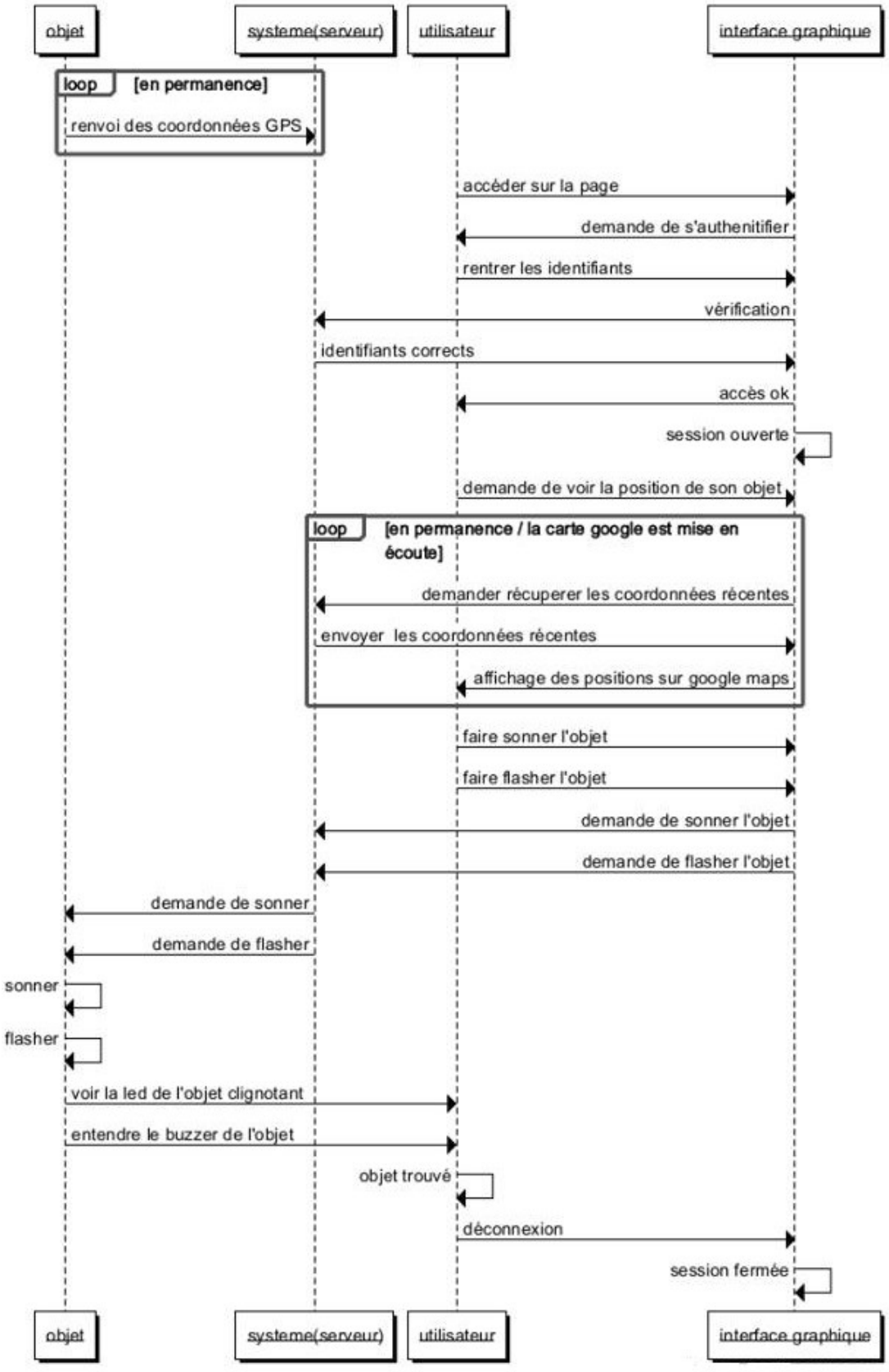
Diagramme de séquence :

Il représente les interactions entre objets en précisant la chronologie des échanges de messages et représente une instance d'un cas d'utilisation (les scénarios possible d'un cas d'utilisation donné).

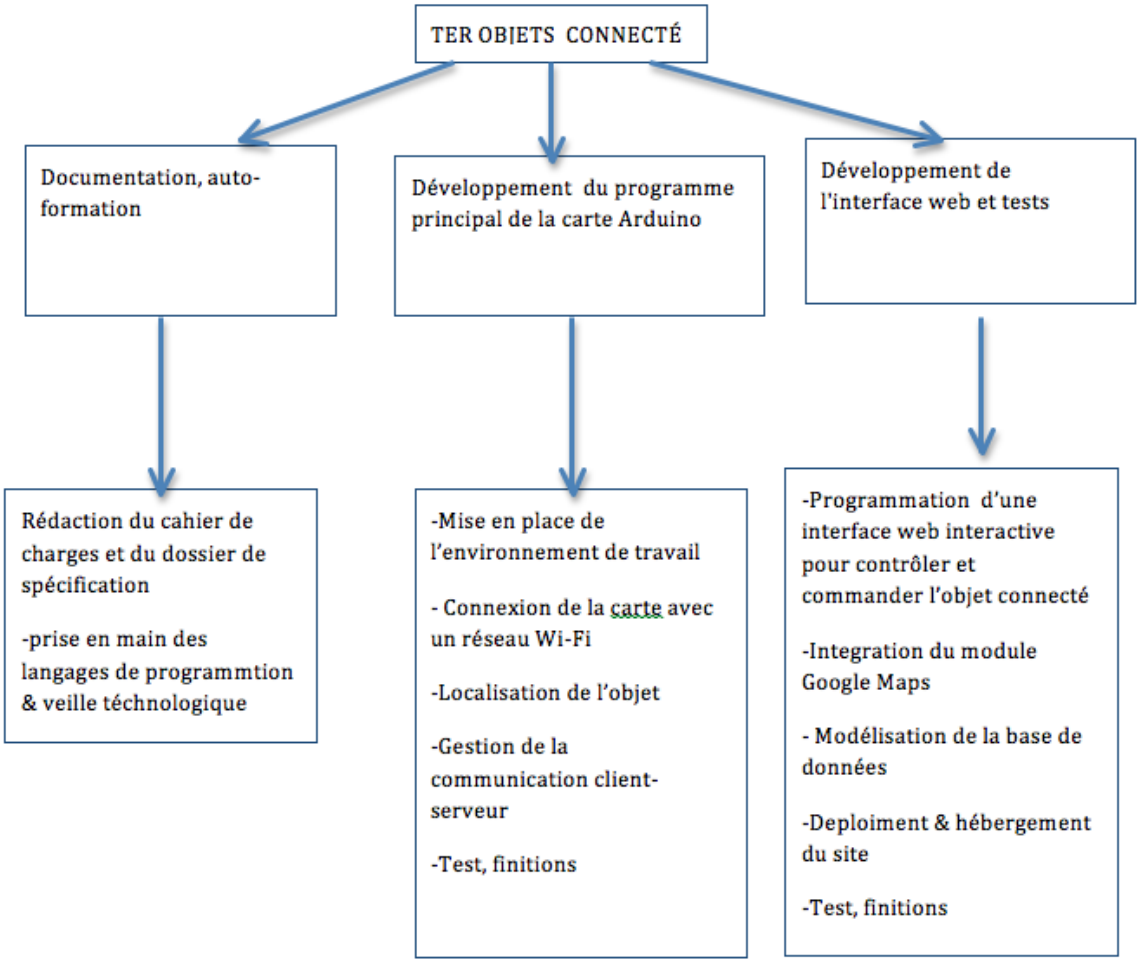
Il nous montre sous forme de scénarios, la chronologie des envois de messages issus d'un cas d'utilisation

Le diagramme de séquence fait ressortir :

- Les acteurs
- Les objets
- Les messages


objet
systeme(serveur)
utilisateur
interface graphique
loop
[en permanence]
renvoi des coordonnées GPS
accéder sur la page
demande de s'authenitifier
rentrer les identifiants
vérification
identifiants corrects
accès ok
session ouverte
demande de voir la position de son objet
loop
[en permanence / la carte google est mise en écoute]
demander récuperer les coordonnées récentes
envoyer les coordonnées récentes
affichage des positions sur google maps
faire sonner l'objet
faire flasher l'objet
demande de sonner l'objet
demande de flasher l'objet
demande de sonner
demande de flasher
sonner
flasher
voir la led de l'objet clignotant
entendre le buzzer de l'objet
objet trouvé
déconnexion
session fermée
objet
systeme(serveur)
utilisateur
interface graphique

Les grandes taches du projet :



2) Mise en place de l'environnement de travail

- Installation du système d'exploitation Ubuntu 15.10
- Installation du logiciel de développement Arduino 2:1.0.5, et installation des librairies nécessaires(GPS+Wi-Fi);
- Installations des outils de modélisation de la base de données

XAMPP v5.6.15, c' est un ensemble de logiciels gratuits multiplate-forme permettant de mettre en place facilement un serveur Web. Il s'agit d'une distribution de logiciels libres (Apache MySQL Perl PHP) offrant une bonne souplesse d'utilisation.

Procédure d'installation :

1)Obtenir la dernière version de XAMPP

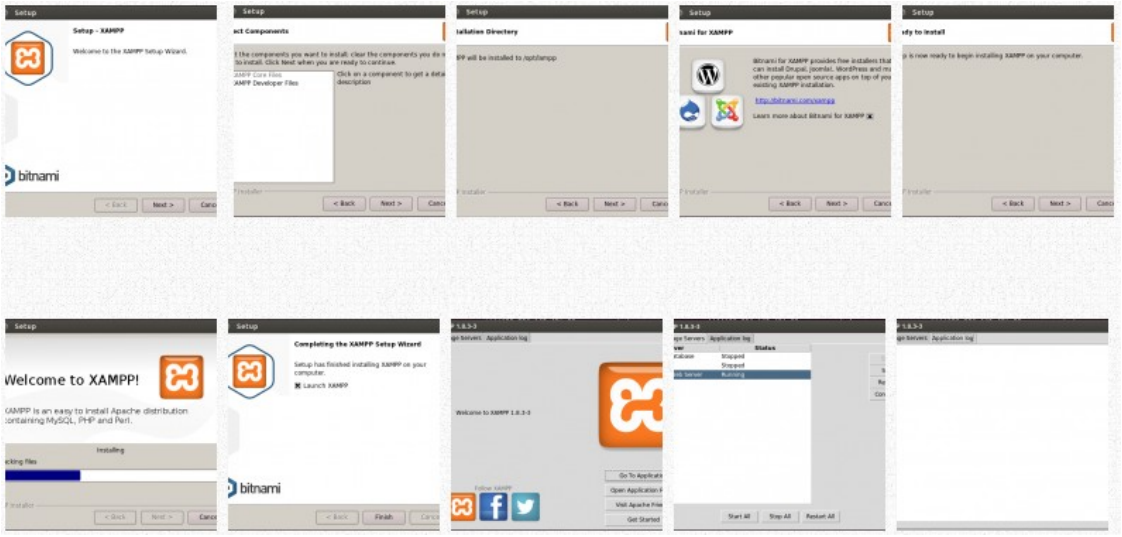
2) Installer XAMPP

d'abord il faut changer les permissions:

```
$ sudo chmod 755 xampp-linux-x64-1.8.2-installer.run
```

Exécuter l'installation:

```
$ sudo ./xampp-linux-x64-1.8.2-installer.run
```



-XAMPP est maintenant installé dans le répertoire **/opt/lampp**.

3) démarrage de XAMPP

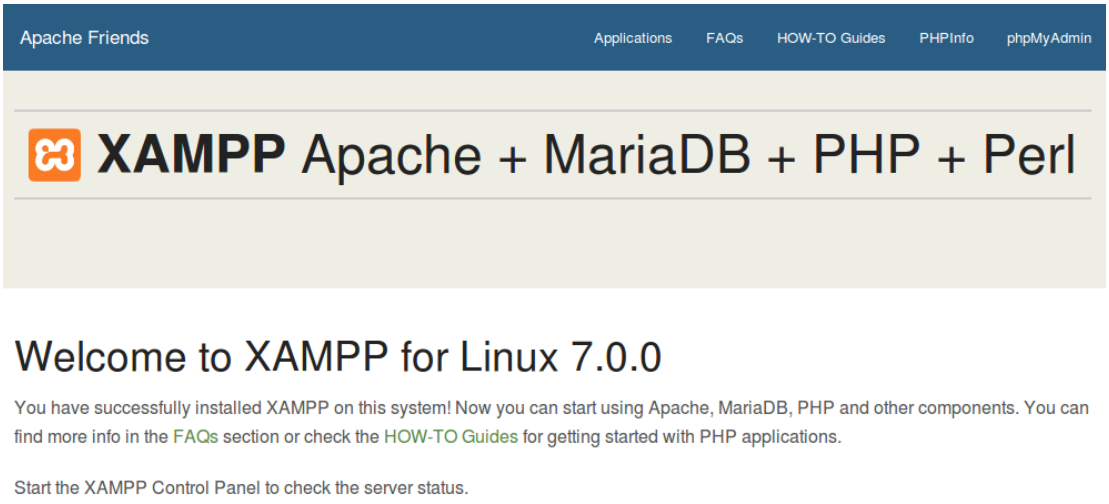
Maintenant que XAMPP est bien installé, on doit démarrer les services avec cette commande :

```
$ sudo /opt/lampp/lampp start
```

```
ayoub@ayoub-N75SF:~$ sudo /opt/lampp/lampp start
Mot de passe [sudo] pour ayoub :
Starting XAMPP for Linux 7.0.0-0...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
```

4) Test

Pour accéder à notre serveur local qu'on vient d'installer il faut juste taper dans la barre d'adresse du navigateur l'adresse suivante: http://localhost



Maintenant XAMPP est prêt a l'utilisation

3) Localisation de l'objet

La première étape dans ce projet c'était de récupérer les coordonnées GPS de l'objet, pour ceci on a utiliser un shield GPS et les bibliothèque fournie avec.

La question qui se pose c'est comment ça fonctionne un shield GPS ?

Principe du Fonctionnement d'un shield GPS

Déjà avant d'expliquer le principe de fonctionnement on expliquera la notion de GPS en général car notre shield Arduino GPS se base sur le même principe qu'un GPS classique.

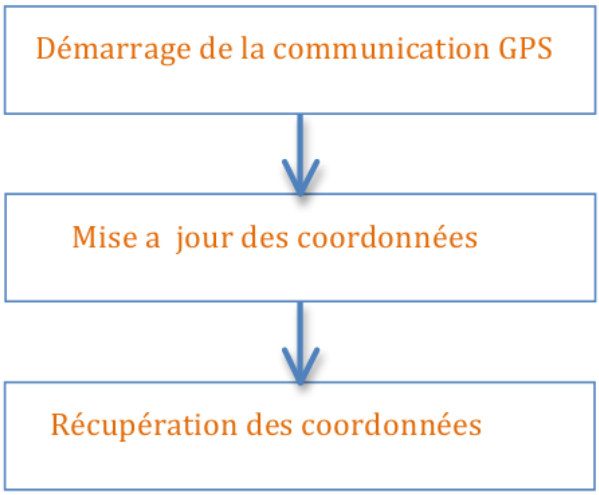
Alors GPS c'est quoi ?

GPS (Global Positioning System) est un système de positionnement par satellites conçu par et pour le département de la défense des Etats-Unis (DoD). Un GPS est constitué de 24 satellites et utilise la triangulation pour se localiser. Le GPS a deux fonctionnalités principales :

- ✓ Localisation de la position avec une précision < 5m
- ✓ Détermination de l'heure exacte avec une précision < 1us

Le shield GPS ajoute des fonctions de géolocalisation par satellite à notre carte ARDUINO Uno, il nous a donné la possibilité de connaître la position (latitude, longitude) , on tient a préciser que le shield GPS fournit pas mal de fonctionnalitées comme la vitesse et l'enregistrement des déplacement et la détermination de l'orientation et bien d'autres fonctionnalitées .

Librairie Arduino GPS :La librairie fournit par le constructeur fournit la plupart des fonctionnalités GPS NMEA donc pour récupérer les coordonnées on a suivi l'acheminement suivant :



-Démarrage de la communication GPS se fait à l'aide de la méthode **init()**

```
void dGPS::init(){
  gpsSerial.begin(9600);
  memset(linea, ' ', sizeof(linea));
}
```

Cette fonction permet de démarrer la communication en série avec le shield GPS, cette étape est primordiale dans la phase de localisation car si on démarre pas la communication avec le shield GPS on peut pas localiser l'objet.

Récupération des coordonnées (latitude & longitude) :

Pour récupérer les coordonnées GPS on a utilisé les 2 fonctions :

Lat() et **Lon()**

```
float dGPS::Lat(){
        return fLat;
}

float dGPS::Lon(){
        return fLon;
}
```

ces 2 fonctions permettent de récupérer respectivement la latitude et la longitude.

Pour vérifier la bonne récupération des coordonnées dans la fonction principale on instancie un nouveau objet de type **dGPS** et on fait appel a nos 2 fonctions qui permettent la récupération des coordonnées de la manière suivante :

```
dGPS dgps = dGPS();
latitude=dgps.Lon();
longitude=dgps.Lat();
```

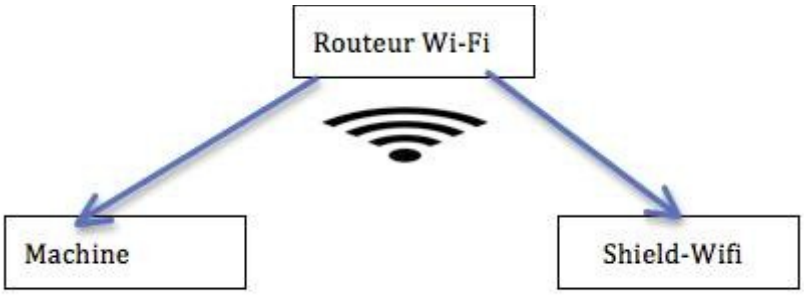
On obtient les coordonnées sur le moniteur série comme montrer dans la figure ci-dessous :

```
longitude=1.076599955558776855468 7&latitude=49.439754486083984375000
```

Une fois que les coordonnées sont récupérée, l'étape suivante c'est d'envoyer les coordonnées a notre base de données a fin que notre interface pourra récupérer ces coordonnées et les afficher dans la carte Google map.

4) Connexion de la carte avec un réseau Wi-Fi

Afin de connecter notre carte Arduino à un réseau Wi-fi local ou à internet, on a utilisé un shield Wi-Fi conçu spécialement pour ça.



Dans le programme principal on a donc besoin d'inclure la classe **Wifi.h**

notre application est de type client-serveur, on a donc besoin d'inclure les classes **WifiClient.h** et **WifiServer.h** qui vont nous fournir toutes les fonctionnalités nécessaires pour réaliser une application client-serveur.

```
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFi.h>
```

Pour établir la connexion, on a programmé une fonction **connectWifi()**

```
void connectWifi() {
  // Attempt to connect to wifi network
  while ( status != WL_CONNECTED)
  {
    Serial.print("Tentative de connexion au réseau: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, password);
    // Wait 10 seconds for connection
    delay(10000);
  }
     Serial.print("Connexion établie");

}
```

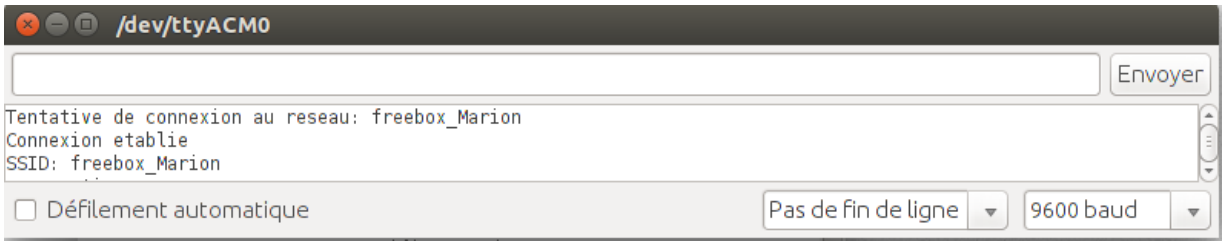
Tentative de connexion avec la fonction begin

On attends 10 secondes le temps que la carte se connecte a notre réseau avant de faire un nouvelle tentative si la 1ere a échoué

Une fois la connexion est établie, on affiche sur le moniteur le nom du réseau auquel on est connecté grâce à la fonction **printwifistatus()**

```
void printWifiStatus() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
}
```

On obtient l'affichage suivant sur le port série



5) Modélisation de la base de données

Le système de gestion de la base de données utilisé :

MySQL : est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL (licence publique générale) et propriétaire. Il fait partie des logiciels de gestion de base de données les 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.

Son nom vient du prénom de la fille du cocréateur Michael Widenius, My. SQL fait référence au Structured Query Language, le langage de requête utilisé.

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread, multi-utilisateur, un logiciel libre et open source.

Source : https://fr.wikipedia.org/wiki/MySQL

Dans la conception de la base de données on a implémenté tout ce qu'on a vu dans le cours de conception (UML) avec Monsieur Youssouf SaidAli.

L'échange et le stockage des données entre l'interface graphique (utilisateur) et l'objet (carte arduino + shield wifi et gps), se fait sur deux tables, utilisateur et objet.

La table **utilisateur** à comme attributs :

- Identifiant comme clé primaire de la table ;
- nom_d_utilisateur et mot_de_passe : pour s'authentifier sur l'application ;
- nom, prenom, telephone et e_mail :informations personnelles ;
- état : attribut booléen pour faire sonner et flasher ;

La table **objet** à comme attributs :

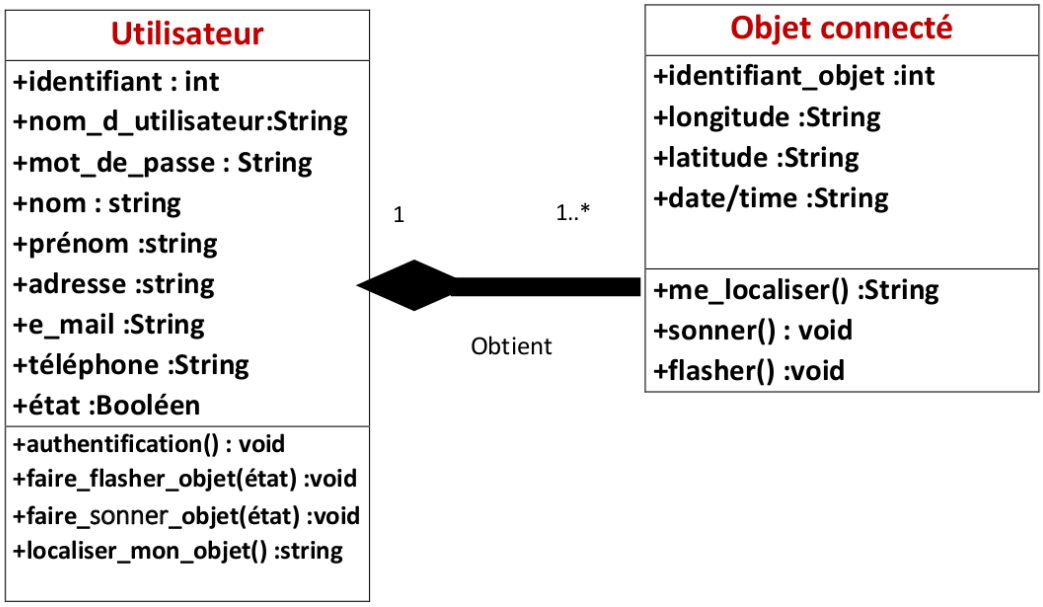
- Identifiant_objet : comme clé primaire de la table ;
- longitude et latitude : coordonnées GPS de l'objet fournis par le shield GPS ;

- date/time : type datetime, sera affecté à chaque mise à jour des coordonnées par la date et l'heure de la localisation ;

Diagramme de classes :

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Comme le montre la figure ci-dessous, le diagramme de classes de notre base de données :



6) Communication client-serveur

Après la première tache qui a été la récupération des coordonnées GPS on a mis en place un mécanisme de communication entre client et serveur :

La première phase c'était la création d'un réseau local entre un routeur x (box, téléphone,..) ,l'Arduino et la machine (client) donc le routeur offre une connexion wi-fi et les 2 équipements se connectent au même réseau wi-fi proposé par le routeur.

une fois la phase connexion est établie, on spécifie l'adresse IP du serveur hébergeant, qui est soit l'adresse IP de notre machine ou bien l'adresse IP du serveur web distant sur lequel notre site sera hébergé.

Après la spécification de l'adresse IP du serveur on vérifie si le client est bien connecté avec le serveur via le port 80 , si oui on procède a l'écriture des coordonnées dans la base de données .

Insertion des coordonnées dans la base de données

A fin d'insérer les coordonnés dans la base de données, on effectue depuis l'Arduino une requête HTTP de type **GET** qui va envoyer des variables vers le script PHP qui va écrire dans la base, ces variables dans Notre cas c'est les coordonnés GPS.
Le format de la requête est le suivant:

"GET /con.php? longitude=value & latitude = value "

Le fichier **con.php** contient un script php qui permet la connexion a la base de données et une requête **MySQL** de type **insert** qui permet l'insertion des coordonnées GPS dans la base de données :

La figure ci-dessous nous montre que la phase d'écriture dans la base a été bien effectué

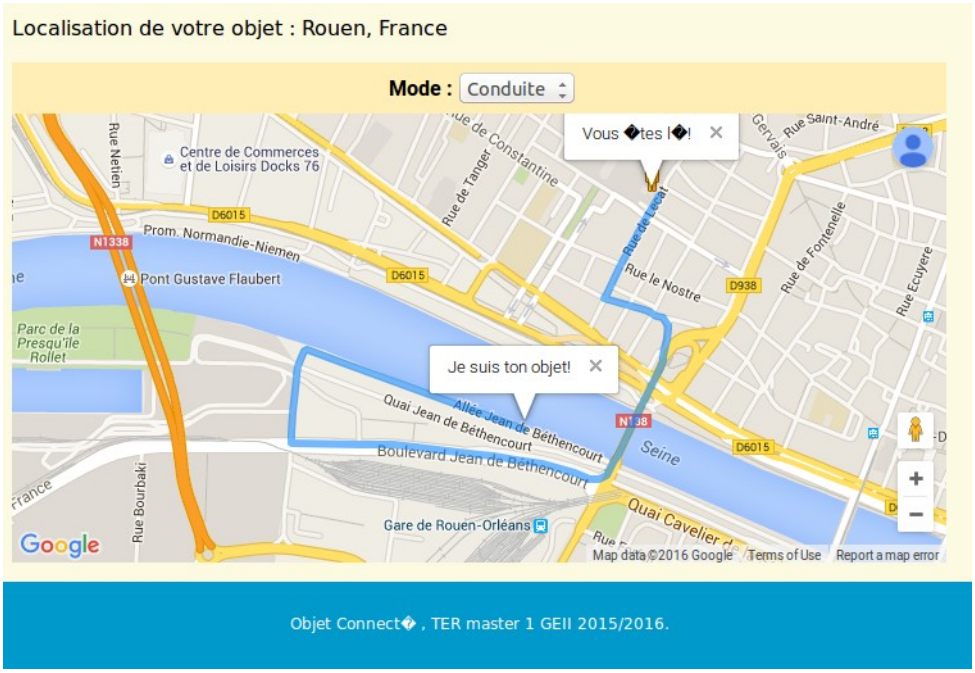
id_objet	lat	lon	id_user
1	49.455646514892578125000	1.062524914741516113281 2	2
2	49.439754486083984375000	1.076599955558776855468 7	2
3	49.440273284912109375000	1.077451586723327636718 7	2
4	49.440940856933593750000	1.078243255615234375000 0	2

Récupération des coordonnées depuis la base de données

A fin de localiser l'objet en question, l'interface web doit récupérer les coordonnées depuis la base de données et les afficher sur la carte.

Pour ceci on a utilisé une requête MySQL de type Select qui permet de sélectionner la latitude et la longitude depuis la base de données

La figure ci-dessous nous montre que la phase lecture depuis la base a bien été effectuée



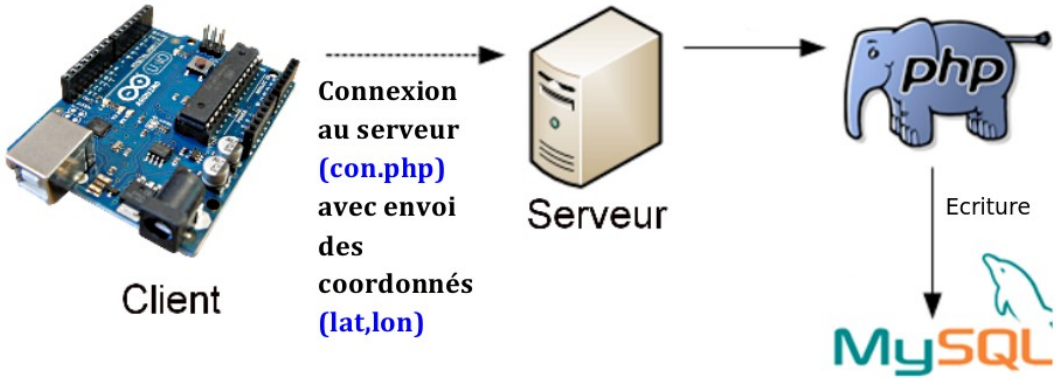
Pour conclure cette partie on peut schématiser le mécanisme de communication client/serveur adopté de la manière suivante :

Phase 1 :

l'Arduino est en mode client , il envoie des demandes au serveur pour écrire dans la base de données, il va en fait envoyer les coordonnées GPS relevées sur notre shield GPS a notre programme qui est présent sur le serveur Apache (con.php).

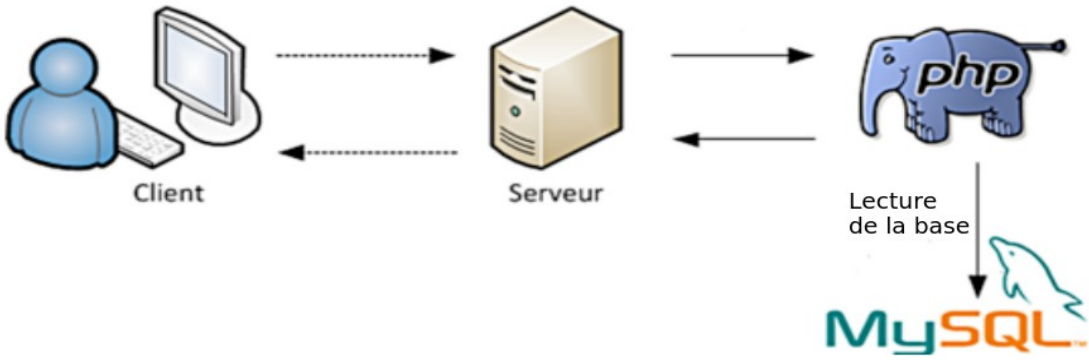
Ce programme va ensuite insérer les coordonnées dans la base de données.

Ci-dessous le schéma de principe :



Phase 2 :

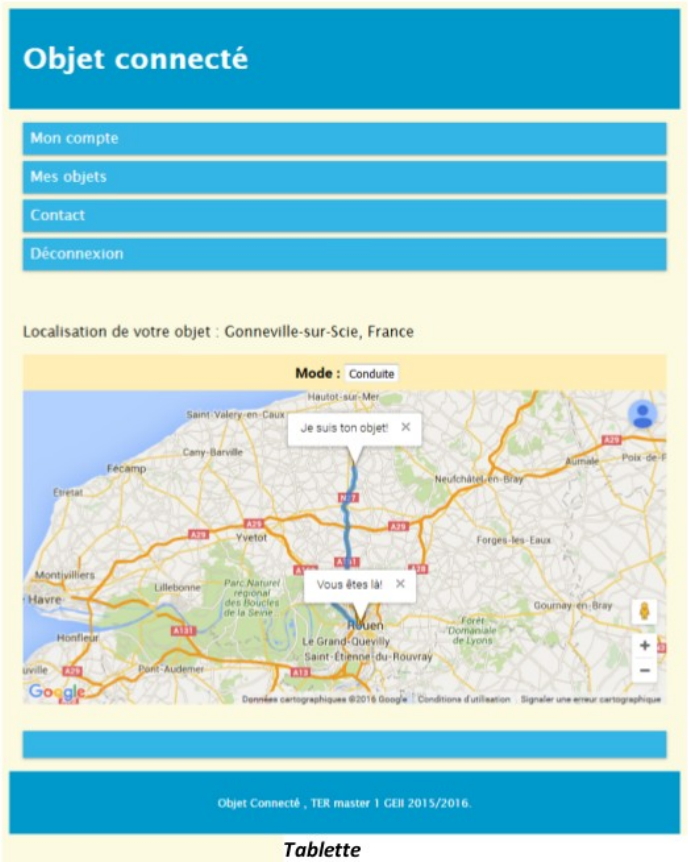
dans cette Phase le client devient la personne qui cherche l'objet donc le client interroge le serveur afin de visualiser la localisation de son objet via une page qu'on génère par le serveur et qui contient les coordonnées récupérées depuis la base de données :



7) Développement de l'interface web

1. Maquette :

Smartphone



Tablette

On a conçu une maquette responsive (responsive template) à l'aide du CSS 3 et HTML5 pour que notre interface web soit correctement visionnée sur les différentes résolutions d'écrans des terminaux (ordinateur, tablette et smartphone) :

2. Authentification :

Cette phase a été réalisée en utilisant une gestion de session par PHP 5.

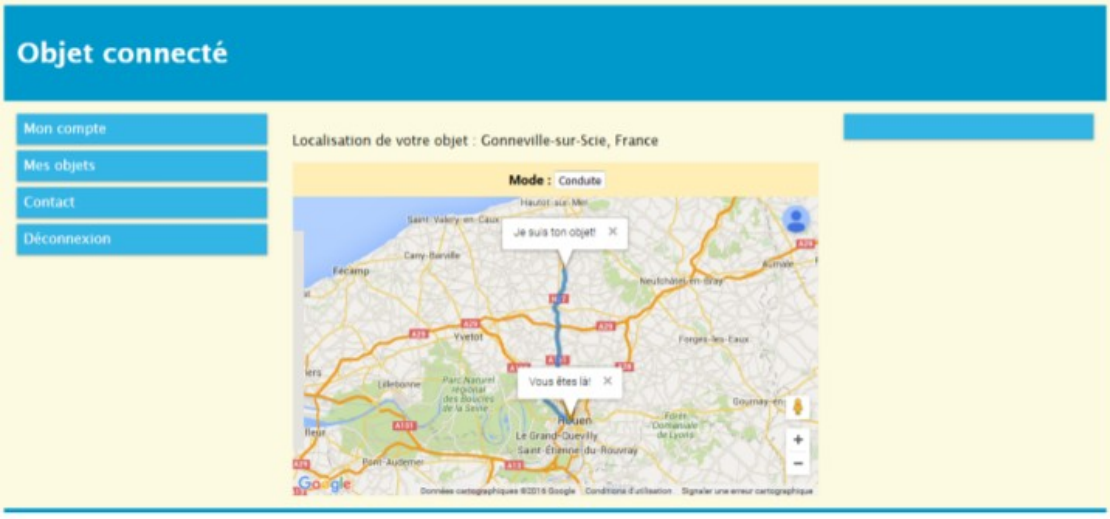
Tout d'abord dans la page de connexion on a mis une balise HTML 5 de type « FORM», avec une méthode « POST », qui contient deux balises « INPUT » de type texte (nom d'utilisateur et mot de passe) et un bouton de type « SUBMIT » (se connecter),

La méthode POST permet d'envoyer les valeurs récupérées dans les champs « INPUT » en cliquant sur le bouton « SUBMIT » vers la page elle-même, et le traitement après le « SUBMIT » est comme ceci :

- On prend les valeurs stockées dans « $_POST['login'] » et « $_POST['password'] » , on les compares avec les champs récupérés de la base de données 'login' et 'password' de la table utilisateur, s'ils sont juste accès ok et session ouverte, sinon accès rejeté.

Après que la session est ouverte l'utilisateur est redirigé vers son espace de localisation.

Page d'authentification



Page de localisation si authentification ok

Page d'authentification si les identifiants ne sont pas corrects :

Page d'authentification si les identifiants ne sont pas corrects

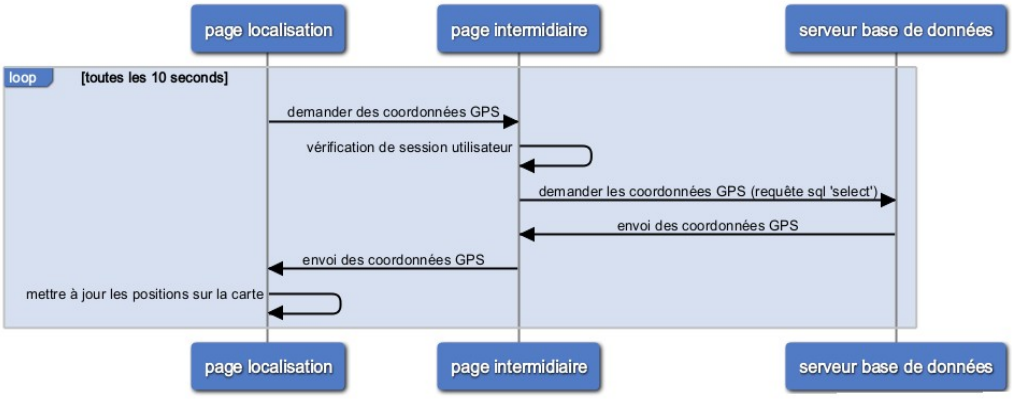
3. Espace de localisation :

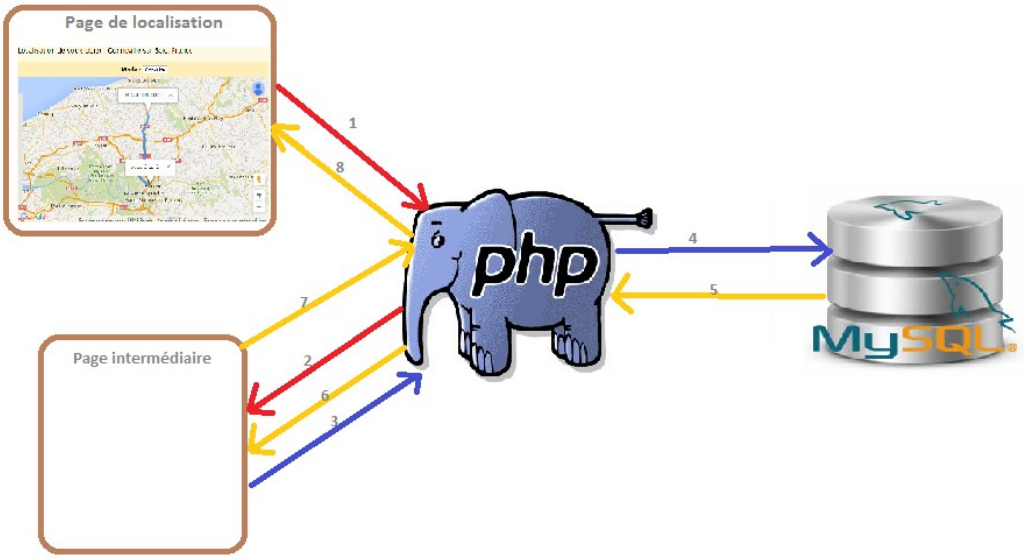
Dans cette page on à la carte géographique de Google dont on a déjà parlé dans la partie API Google Map.

Cet espace est accessible si seulement si une session utilisateur est en cours pour des raisons de confidentialité et de sécurité, et aussi pour que chaque utilisateur ait un espace privé de localisation de ses propres objets.

La récupération des coordonnées stockées de l'objet en permanence dans la base de données pour les afficher dans la carte, se fait par une page PHP intermédiaire, qui tourne derrière, interrogée par un script 'Ajax' dans la page de localisation par une méthode 'POST'.

La page intermédiaire a comme rôle de récupérer les coordonnées 'lat' et 'long' stockées dans la table 'objet', par une connexion au serveur base de données Mysql interrogé par une requête SQL 'select', et les renvoyer par une méthode 'POST' toutes les 10 secondes vers la page de localisation sous forme de balises HTML 5 'INPUT' (leur identifiants 'lat' et 'long') de type 'hidden'. Et ensuite le script de la carte Google récupère les valeurs des champs 'lat' et 'long' en permanence toutes les 10 secondes pour afficher les nouvelles positions ainsi que mettre à jour l'adresse postale.





1.interrogation du serveur php pour l'envoi des données (session d'utilisateur) à la page intermédiaire par un script en permanence toutes les 10 secondes.

2.le serveur php envoie des données concernant la session d'utilisateur postées vers la page intermédiaire.

3.la page intermédiaire traite les données reçu et envoie une demande au serveur php pour interroger le serveur Mysql .

4.le serveur php de son côté envoie la requête sql préparée par la page intermédiaire pour récupérer les coordonnées de l'objet de l'utilisateur.

5.le serveur Mysql traite la requête reçue , et envoie les coordonnées au serveur php.

6.le serveur php renvoie ces coordonnées à la page intermédiaire

7.la page intermédiaire encapsule les coordonnées dans des balises HTML5 de type 'INPUT' et les envois via le serveur php vers la page de localisation.

8.le serveur php renvoie ses balises à la page de localisation ,et dans cette dernière il y a un script qui tourne en permanence toutes les 10 secondes pour mettre à jour la carte géographique.

8) Intégration de l'API Google Map

API (application programming interfaces) Google est un ensemble des librairies de codes compilés développées par Google prêt à l'emploi que l'on peut intégrer dans les programme, qui permettent la communication avec les services Google.

Google a mis à la disposition des développeurs un service de cartographie que l'on peut utiliser dans les interfaces web, qui propose la géolocalisation par navigateur, et traduit les coordonnées GPS en position (identifiée en marqueur comme un bonhomme pour l'utilisateur) sur la carte Google Map ainsi qu'en adresse postale en proposant plusieurs mode d'itinéraire liant la position de l'utilisateur avec son objet connecté ,éventuellement plus d'autres options.

Sur notre interface on initialise la carte par la position courante de l'utilisateur et de l'objet, et puis on l'a mis en écoute avec un délai de 10 seconds, si les positions changent les marqueurs pointés changent de position aussi sur la carte.

L'utilisateur a la main pour choisir le mode d'itinéraire pour tracer le chemin vers son objet.

Sources d'infos :

https://en.wikipedia.org/wiki/Google_APIs

https://developers.google.com/maps/documentation/javascript/

Les fonctions de google map API utilisées :

La fonction « initmap » : permet d'initialiser une carte géographique de Google personnalisée en mettant les options qui convient :

- Initialisation de la variable myOptions :

```
var myOptions = {
zoom: 13,
center:p2,
mapTypeControl: false,
navigationControlOptions: {style: google.maps.NavigationControlStyle.SMALL},
mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

- Instanciation de la carte géographique de Google:

On instancie un objet de type google.maps.map qui prend les paramètres suivant :

- L'identifiant « map » de la division HTML là où on veut créer notre carte.
- myOptions

```
var map = new google.maps.Map(document.getElementById('map'), myOptions);
```

La fonction « navigator.geolocation.getCurrentPosition » :

permet de retourner les coordonnées du terminal (ordinateur ou smartphone) à l'aide de l'objet :

```
p1 = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
```

On donne un marqueur sur la position de l'utilisateur comme suivant :

```
var marker = new google.maps.Marker({
    position: p1,
    map: map,
    title:"Vous êtes là!",
    icon:"me.gif"
});
```

La fonction « geocoder.geocode » :

qui permet de traduire les coordonnées GPS en adresse postale qu'on affiche au dessus de la carte :

```
geocoder.geocode({'location': latlng}, function(results, status) {

  if (status === google.maps.GeocoderStatus.OK) {
    if (results[1]) {
        //s.innerHTML =status;
       s.innerHTML = results[1].formatted_address;

    }
    }
  }
```

Résultat sur l'interface Web :

9) Déploiement et hébergement du site

A fin d'avoir un produit fini et qu'on peut tester en ligne on a décider d'héberger notre site sur les pages perso free.

Pourquoi les pages perso chez free ?

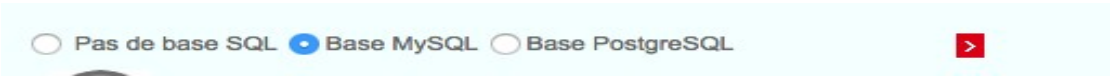
Le service pages perso permet de disposer d'un espace disque sur les serveurs de free dédié exclusivement à héberger des pages web personnelles.

L'ensemble de nos pages web sera accessible par un navigateur **http** et elles comportent une page d'accueil (index de répertoire) qui permettra d'accéder directement a l'ensemble des fichiers contenus sur l'espace réservé.

Étape 1 : Activation de la page perso

pour ceci on a créé une adresse mail free ensuite on se connecte sur l'espace perso pour activer la page perso comme le montre la figure ci-dessous :

Ensuite on choisi le type de la base de données :



Et enfin on active la base de données

Le délai d'activation de notre page perso a pris 20 jours l'ensemble des pages a été activer le 18/01/ 2016

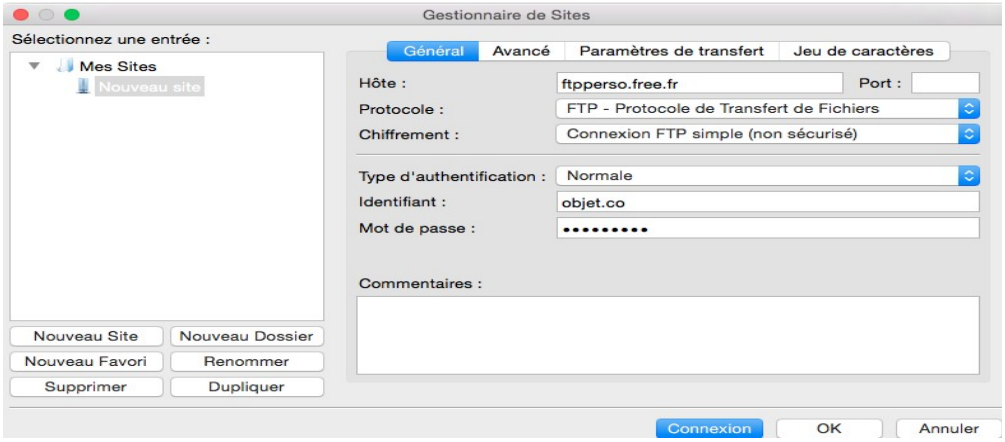
Etapes 1 : hébergement du site

A fin de héberger notre site on a utilisé l'hébergeur **FilleZilla** qui nous a permis de gérer l'ensemble de nos pages perso et l'hébergement de ces derniers dans l'espace perso free.

Hébergeur de site

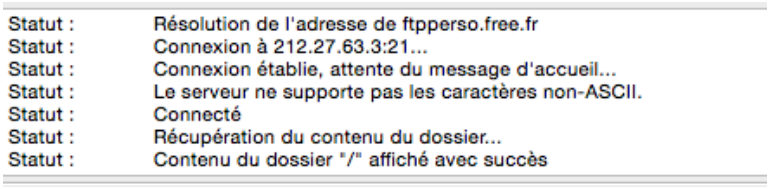
Configuration de Fillezilla :

La première étape dans la configuration de fillezilla et le paramétrage de l'accès FTP dans FileZilla



comme le montre la figure en –dessous on spécifier le hôte ftp.perso.free.fr et ensuite le type de protocole qui est **ftp** est enfin on a renseigné l'identifiant et le mot de passe de notre espace perso free .

Et comme le montre la figure ci-dessous on voit bien que la connexion avec notre serveur a bien été effectuée :



Notre serveur a pour adresse IP **212.27.63.3** et le port **21** car le protocole utilisé est **FTP** afin que l'Arduino communiquer avec notre serveur distants on doit lui spécifier ces deux informations **(@IP,PORT)**

Configuration du fichier .htaccess

Les fichiers .htaccess fournissent une méthode pour modifier la configuration du serveur au niveau de chaque répertoire . Un fichier contenant une ou plusieurs directives de configuration , est placé dans un répertoire de documents particulier ,et ses directives s'appliquent a ce répertoire et a tous ses sous répertoires .
Configuration du fichier .**htaccess** pour notre page perso **free**

```
<IfDefine Free>
php 1
</IfDefine>
Options -Indexes
```

On tient a préciser que la phase hébergement est toujours en cours de réalisation.

10) Contraintes techniques et solutions

-Mise à jour du firmware

La première contrainte rencontrer dans ce projet c'etait la mise du shield Wi-Fi alors pour palier a ce problème on a procédé comme ceci :

Étape 1 :

Téléchargement & installation du DFU PROGRAMMER qui permet la mise à jour du shield Wi-Fi via la commande :

```
sudo apt-get install dfu-programme
```

Étape 2 :

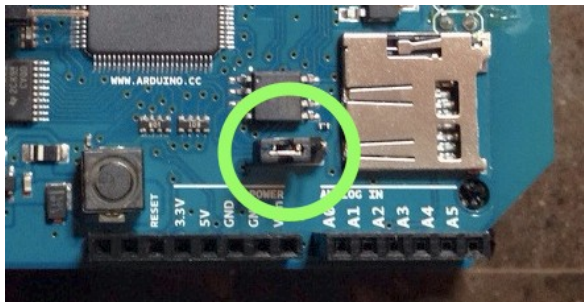
Mise a jour de Mac- port le Mac-port est un gestionnaire de paquet qui est similaire apt-get sur l'OS linux :

```
sudo port selfupdate
```

```
sudo port upgrade outdated
```

Étape 3 :

Une fois l'environnement du travail a été mis en place on télécharge le firmware et ensuite on met l'arduino shield en mode programmation en connectant le cavalier comme le montre la photo ci-dessous

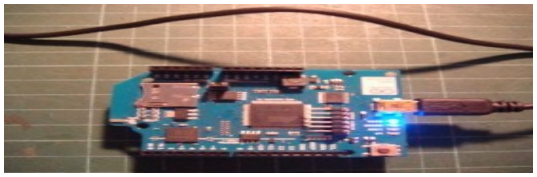


Ensuite on exécute le firmware dans le chemin suivant :

```
~/arduino1.0.6/hardware/avr/arduino/firmwares/wifishield/scripts
```

Via la Commande : `sudo ./ArduinoWifiShield_upgrade.sh –h`

La figure ci-dessous montre bien que l'arduino Wi-Fi a été mis a jour :



l'arduino Wi-Fi shield allume la led on Blue ce qui signifie que l'arduino a bien été mis à jour.

-Incompatibilité de Windows et MAC OS :

On avait rencontré des problèmes de fonctionnement du serveur local et de la base de données sur les systèmes d'exploitation Windows et sur MAC OS.

L'accès au serveur local et à la base de données depuis un autre appareil appartenant au même réseau que la machine HOST était impossible, ce qui nous posait un problème car notre carte a besoin d'accéder à la base de données pour pouvoir insérer les coordonnées dans la table.

On avait essayé plusieurs logiciels(EasyPHP, WAMP, MAMP) pour mettre en place le serveur et la base mais le problème est toujours le même.

Solution :

à la base on possédait que Windows et MAC OS donc on s'est dit que peut être le problème vient du système d'exploitation, donc on avait installé Ubuntu qui est open source, et effectivement le problème venait du système d'exploitation.

Ensuite on avait installé le logiciel XAMPP (équivalent de EasyPHP sous Windows et MAMP sous MAC OS) et le problème est résolu.

-Écriture dans la base de données :

Afin d'insérer les coordonnés dans la base de données, on effectue depuis l'Arduino une requête HTTP de type GET qui va envoyer des variables vers le script PHP qui va écrire dans la base, ces variables dans notre cas c'est les coordonnés GPS.

Le format de la requête est le suivant :

```
Client.print(''GET/ma_page.php?var_1=val_1&var_2=val_2….var_n=val_n
HTTP/1.1'') ;
```

C'est pratique et fonctionnel pour envoyer des données statiques, mais nous on a besoin d'envoyer des données qui sont variables(position) et qu'on met a jour avant chaque envoi.

Il existe une méthode pour envoyer des données dynamiques, c'est de fragmenter la requête en plusieurs parties comme ceci :

```
Client.print(''GET /ma_page.php?var_1='') ;

Client.print(val_1) ;

Client.print(''&var_2='') ;

Client.print(val_2) ;

Client.print('' HTTP/1.1'') ;
```

Malheureusement cette technique de fragmentation n'a pas fonctionné , on a supposé que le problème vient du faite qu'on insère des données de type réel au milieu d'une chaîne de caractères, pour cela on a développé une fonction de conversion "**float_to_string**" qui prend en argument le réel a convertir et la précision(nombre de chiffres après la virgule car une haute précision est requise >20) et qui renvoie un "string".

Malheureusement ça n'a pas marché, donc on a cherché une autre solution.

Après la réalisation de multiples essais, on avait finalement réussi à insérer des données saisies en dur sans fragmentation, d'où est venu l'idée de construire d'abord la chaîne de caractères qui sera envoyée puis effectuer la requête.

Voici un extrait du code qui montre comment la chaîne est construite

```
String chaine = "GET /con.php?longitude=";
dgps.update(desLat, desLon);

floatVal_lon=dgps.Lon();
floatVal_lat=dgps.Lat();

float_to_string (floatVal_lon,charVal_lon,22);
float_to_string (floatVal_lat,charVal_lat,22);
Serial.print("lon : ");
for(int i=0;i<24;i++)
{
      chaine+=charVal_lon[i];
      Serial.print(charVal_lon[i]);
}

Serial.print("\n");
char s[11]="&latitude=";
Serial.print("lat : ");
for(int i=0;i<=9;i++)
{
     chaine+=s[i];
}

for(int i=0;i<24;i++)
{
  chaine+=charVal_lat[i];
  Serial.print(charVal_lat[i]);
}
    Serial.print("\n");
    Serial.print("chaine : ");

 for(int i=0;i<chaine.length();i++)
{
  Serial.print(chaine[i]);
}
client.println(chaine);
client.println( " HTTP/1.1");
```

Création de la chaine de caracteres

Mise a jour des coordonnées GPS

Récupération des coordonnées dans des variables

Conversion des coordonées en type string

Concaténation de la chaine

Concaténation de la chaine

Concaténation de la chaine

executuion de la requete

V-Améliorations

On avait décidé en accord avec notre encadrant M. NICOLAS et Mme. PETITJEAN de poursuivre le projet en 2eme semestre, car il nécessite encore quelque amélioration pour un fonctionnement complet et fiable.

On avait réfléchi a plusieurs amélioration on avait cité :

- **Ajout d'une LED et un BUZZER:**

  La précision de notre Shield GPS n'est pas très grande, quand il s'agit des distances inférieurs à 10m, il est difficile de connaître l'endroit exacte de l'objet.

  Pour résoudre ce problème, on a pensé à mettre une LED que le client peut commander à l'aide d'un bouton dans l'interface web, la LED peut être allumée pour repérer l'objet dans un endroit a faible luminosité.

  La deuxième solution c'est mettre un buzzer qui sera également commandé par un bouton sur l'interface, le client pourra l'activer pour émettre un son afin de repérer son objet dans une une maison par exemple.

- **Modélisation et fabrication d'un boiter :**

  La modélisation en 3D puis l'impression du boîtier qui va contenir notre objet à l'aide de l'imprimante 3D dont on dispose dans notre faculté.

- **Remplacement du shield wi-fi par un shield GPRS :**

  La solution proposée qui consiste à utiliser un shield wi-fi pour connecter la carte Arduino à internet reste une solution prototype, car en cas de perte, l'objet doit être à proximité d'un réseau wi-fi pour pouvoir se connecter et envoyer sa position, et s'il n'y a pas de wi-fi a proximité il sera inutile, donc on propose d'utiliser un shield GPRS qui fonctionne avec une carte SIM.

  Celle solution est plus pratique car c'est beaucoup plus facile de trouver du réseau téléphonique que trouver un point d'accès wi-fi.

- **Utiliser un Arduino Airbord**

- **Développement d'une application Android**

VI-Bibliographie

http://www.generationrobots.com/fr/401115-shield-gps-pour-arduino.html

https://www.arduino.cc/en/Main/ArduinoBoardUno

https://www.arduino.cc/en/Main/ArduinoWiFiShield

http://trentejours.com/test-du-wistiki-lobjet-connecte-made-in-france-pour-retrouver-vos-cles/

http://www.wistiki.com/fr/

http://www.digitaltrends.com/home/heck-internet-things-dont-yet/

www.google.com/images

https://en.wikipedia.org/wiki/Internet_of_Things

http://www.les-objets-connectes.fr/

https://fr.wikipedia.org/wiki/XAMP