

E-Shepherd



Surveillance du bétail

Projet d'Informatique et de Sciences du Numérique

Lycée Janson de Sailly

Table des Matières

Présentation du projet	2
Présentation générale	2
Analyse fonctionnelle et Architecture	2
Partie Terrain	6
Partie Serveur	8
La base de données	8
Le listener	9
La page de connexion	10
La page d'accueil	12
La page de configuration du champ	14
La page de visualisation des positions	16
La page d'historique des déplacements	18
Conclusion	19

Présentation du projet

Présentation générale

Dans le cadre des cours d'Informatique et de Sciences du Numérique, je me suis intéressé au moyen de surveiller des animaux qui évoluent en semi-liberté dans de grands espaces. Ce sujet est très important dans notre société où les nouvelles technologies envahissent les champs pour aider les éleveurs dans leurs tâches les plus fatigantes et répétitives.

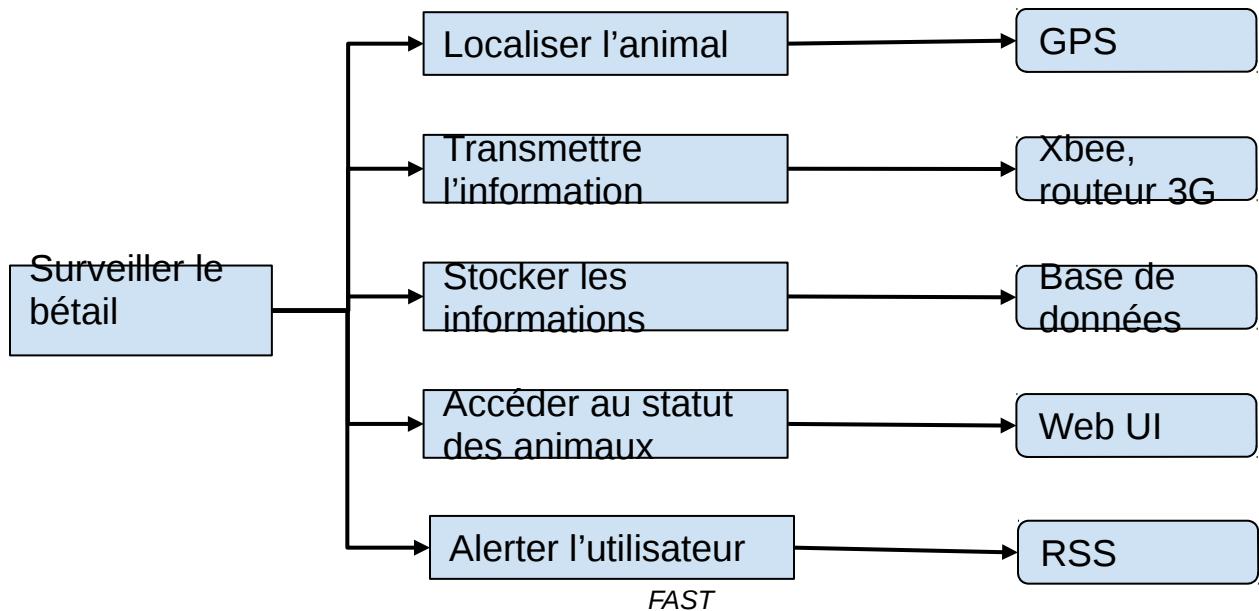
Pour répondre à ce besoin, j'ai décidé de créer une application qui permet à l'éleveur de surveiller son bétail à distance, en temps réel et d'être prévenu au cas où un animal sort du périmètre autorisé.

Ce système a pour avantages :

1. de pouvoir fonctionner dans des endroits où l'accès internet n'est pas possible
2. d'être moins cher sur la durée que les solutions qui utilisent un abonnement GSM par animal. Son coût de déploiement est de moins de 60€ par animal et par station relais.

Analyse fonctionnelle et Architecture

Pour commencer, j'ai réalisé un diagramme FAST qui permet de décomposer les fonctions de service en fonctions techniques. J'ai ensuite commencé à rechercher les solutions techniques et les composants dont j'avais besoin pour mener le projet à bien.

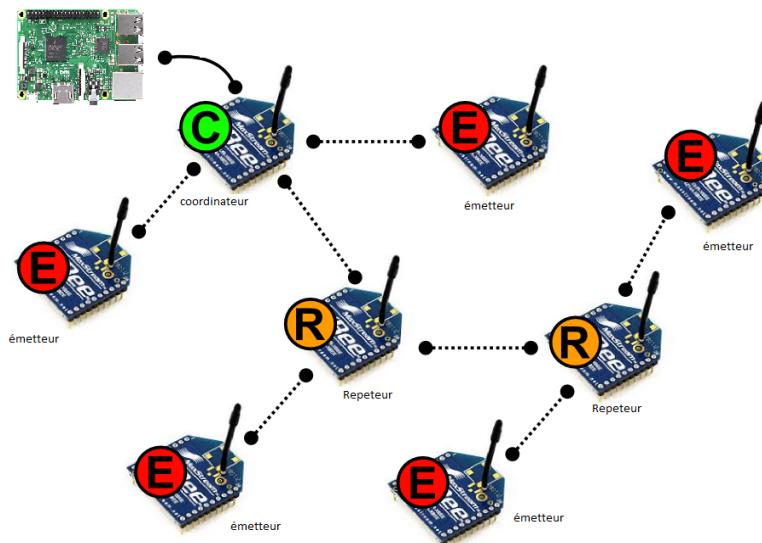


J'ai donc cherché le matériel informatique le plus adapté pour mon projet.

J'ai opté pour des récepteurs-émetteurs XBee qui permettent de déployer facilement un réseau maillé à un coût très réduit tout en ayant une consommation électrique faible et une portée de 1,5 km dans les conditions les plus favorables.

Pour contrôler ceux-ci, j'utilise des Raspberry-Pi qui consomment peu et un port USB pour le GPS.

Pour alimenter le tout, j'ai choisi une batterie bank de téléphone.



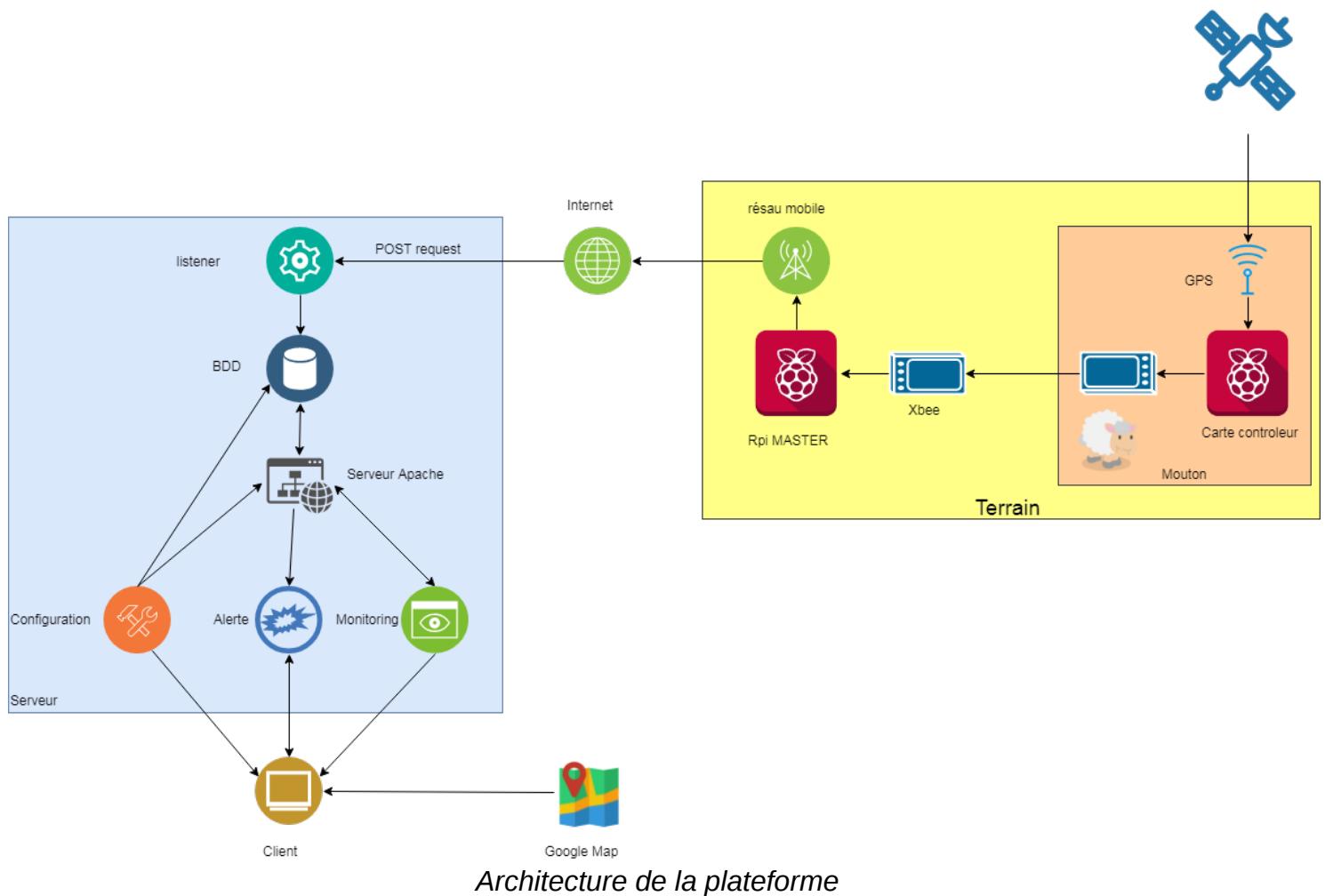
Réseau maillé Xbee

Pour des questions de coûts et d'indisponibilité du matériel au laboratoire de
Maxime Favier Terminale S5

Sciences de l'Ingénieur et d'ISN, j'utiliseraï, à la place des coordonnées GPS, des nombres aléatoires, des Xbee qui ont une portée limitée à 100 m et j'alimenterai les cartes avec un adaptateur secteur pour démontrer la faisabilité de mon projet.

Le principe est donc le suivant :

- Un GPS sur l'animal envoie les coordonnées GPS et son identifiant sur un réseau maillé Xbee
- Cette information, en passant par des stations relais, arrive à un récepteur "maître" qui envoie l'information à un serveur par le réseau mobile.
- Les coordonnées sont mise sur une base de données
- Le berger dispose d'une interface web avec la position de son troupeau



Sur cette base, j'ai construit l'architecture du projet, puis j'ai choisi les solutions logicielles et j'ai développé le code.

J'ai divisé en le projet deux parties : la partie "Terrain", et la partie "Serveur"

Le code utilisé est accessible sur un git <https://github.com/Maxime-le-Goupi/e-shepherd-xbee> sous licence libre Apache.

 Maxime-le-Goupi/e-shepherd is licensed under the Apache License 2.0	Permissions ✓ Commercial use ✓ Modification ✓ Distribution ✓ Patent use ✓ Private use	Limitations ✗ Trademark use ✗ Liability ✗ Warranty	Conditions ⓘ License and copyright notice ⓘ State changes
A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.			

Partie Terrain

Cette partie, très courte, regroupe le code qui va envoyer les informations sur le réseau Xbee et le traitement des trames reçues en vue de leur stockage dans une base de données. J'utilise le langage de programmation Python, qui fonctionne bien sous Linux et est très rapide à développer grâce au nombre important de librairies.

Pour commencer, j'ai écrit le code qui commande l'envoi des coordonnées GPS. Pour diminuer le temps nécessaire au développement, j'utilise la library "Xbee Device" qui permet de récupérer et d'envoyer facilement les trames au module Xbee via le port série de la carte Raspberry pi.

```
import serial, time, random
from digi.xbee.devices import XBeeDevice

device = XBeeDevice("/dev/ttyAMA0", 9600)
device.open()
print("connecté")

while True:
    try:
        req = str([random.random(), random.random(), random.random()])
        device.send_data_broadcast(req)
        print("envoyé !")
        time.sleep(60)

    except KeyboardInterrupt:
        break

device.close()
```

Dans ce code, après s'être connecté au Xbee qui est connecté au port série /dev/ttyAMA0, j'entre dans une boucle infinie. Dans celle ci, je génère une chaîne de caractères constituée de trois nombres aléatoires qui correspondent à la latitude, la longitude et le numéro identifiant de l'animal. Les caractères [et] permettent de bien repérer la trame quand on la recevra . Un délai de 60 secondes est présent dans la boucle pour éviter de saturer la base de données et économiser l'énergie en limitant la charge CPU. Un bloc try - except permet de savoir quand l'utilisateur appuie sur Ctrl+C ce qui a pour effet d'arrêter le programme correctement en terminant la connection série au Xbee.

```
# chargement des libs
```

Maxime Favier Terminale S5

```

import serial, re, requests
from digi.xbee.devices import XBeeDevice

# connection en serial au Xbee
device = XBeeDevice("/dev/ttyAMA0", 9600)
device.open()
print("connecté")

while True:
    try:
        # ---- RÉCUPÉRATION ET TRAITEMENT DE LA TRAME ---- #
        xbee_message = device.read_data() # récupération de la trame
        if(xbee_message != None):
            msg = str(xbee_message.data) # conversion en str
            msg = re.findall(r'\[(.*?)\]',msg) # suppression des données non
voulues
            msg = msg[0] # reconversion en str
            msg = msg.replace(" ", "") # suppression des espaces
            msg = msg.split(",") # conversion en liste
            print(msg)

        # ---- ENVOI DE LA REQUÊTE AU SERVEUR ---- #
        sendcontent = {'idmbutton': msg[2], 'lat': msg[0], 'lng': msg[1], 'mdp':
'2788223e73728d8339cbc5366945c90b0a394bfa6bafe1426cc5eb221fd36bba'}
        r = requests.post("http://localhost/lisnter.php", data=sendcontent)
        print("envoyé")

    except KeyboardInterrupt:
        break

# déconnection
device.close()

```

Dans le programme de réception, on a de nouveau une boucle infinie qui regarde en permanence la sortie du Xbee. Si une trame est reçue, elle est traitée en utilisant une regex qui permet de sélectionner le message envoyé parmi les autres informations données par le Xbee, comme la date d'envoi, le numéro de série des nœuds du réseaux...

En plus, j'envoie une requête sur le “listener” de mon application. Je génère donc un dictionnaire avec les trois valeurs récupérées et un mot de passe (pour éviter que quelqu'un de mal intentionné remplisse la bdd).

Partie Serveur

La base de données

Une base de données se comporte comme un tableau excel à la différence que pour accéder aux données on utilise un langage de programmation spécifique : le SQL. On pourrait faire l'analogie entre une base de données et un document .xls , les tables et les feuilles de calcul...

The screenshot shows the phpMyAdmin interface with the 'positions' table selected. The left sidebar lists databases (geomutton, information_schema, maxime_favier, mysql, performance_schema, phpmyadmin) and tables (area, auth, positions). The 'positions' table is highlighted with a yellow border. The main area displays the table structure with columns: id, idmouton, date, lat, lng, and pos. Below the table, a list of 65 rows of data is shown, each with edit and delete options. The bottom right corner of the data grid has a tooltip 'Données'.

	A	B	C	D	E	Collones
1		idmouton	date	lat	lng	
2		2	18	22/05/18 0.341524525	0.389690845	
3		3	18	22/05/18 0.354558235	0.388244695	
4		4	18	22/05/18 0.529442861	0.137983985	
5		5	13	22/05/18 0.476303132	0.679037434	
6		6	9	22/05/18 0.456540903	0.317984495	
7		7	18	22/05/18 0.751030138	0.258434426	
8		8	3	22/05/18 0.776013494	0.56044265	
9		9	2	22/05/18 0.759890434	0.595116136	
10		10	1	22/05/18 0.405853792	0.048410412	
11		11	6	22/05/18 0.13856311	0.795320845	
12		12	14	22/05/18 0.914399895	0.134121415	
13		13	12	22/05/18 0.768982878	0.996271426	
14		14	9	22/05/18 0.01059826	0.740815731	
15		15	12	22/05/18 0.065236711	0.412235694	
16		16	16	22/05/18 0.55349018	0.292629821	
17		17	14	22/05/18 0.39245720	0.585814216	
18		18	11	22/05/18 0.406683673	0.829308706	
19		19	2	22/05/18 0.761377889	0.252865299	
20		20	8	22/05/18 0.62654673	0.282194579	
21		21	17	22/05/18 0.619471801	0.68388142	
22		22	13	22/05/18 0.973118901	0.9897522	
23		23	20	22/05/18 0.589191203	0.7082122	
24		24	20	22/05/18 0.383400701	0.386931554	
25		25	14	22/05/18 0.080059895	0.48627338	
26		26	5	22/05/18 0.777490273	0.663569013	
27		27	3	22/05/18 0.609359837	0.316122912	
28		28	6	22/05/18 0.070737771	0.834339735	
29		29	1	22/05/18 0.503471980	0.67640732	
30		30				

Pour faire fonctionner la base de données, j'utilise le logiciel Mysql qui est libre et gratuit et phpmyadmin qui permet d'avoir une interface de gestion de celle-ci.

J'ai déterminé la meilleure façon de partitionner les données:

- Une table "auth" qui va enregistrer toutes les informations de l'utilisateur pour l'authentification,
- Une table "position" qui enregistre toutes les données envoyées par le bétail et
- Une table "area" qui stocke la position des sommets du champ.

Le listener

Le listener, après s'être connecté à la base de données Mysql, va récupérer le contenu de la post-request. Je vérifie que les données reçues et que le mot de passe sont corrects.

Puis je prépare une requête SQL qui va insérer les variables dans la base de données, et je l'exécute en mettant en argument la liste de variables à sauvegarder.

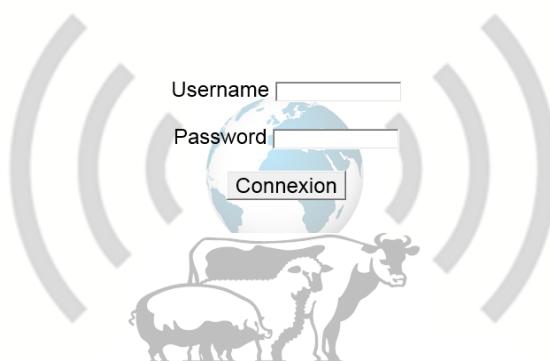
Nom	ID	IDmoutton	datation	lat	lng	pos
Type de données	Clef primaire	Integer	Date-Time	Float	Float	Point (Non utilisé)

```
<?php

try
{
    $bdd = new PDO('mysql:host=localhost;dbname=maxime1_favier;charset=utf8', 'root', 'root');
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}

if(!isset($_POST['idmoutton']) OR !isset($_POST['lat']) OR !isset($_POST['lng']) OR !
isset($_POST['mdp']))
{
    echo 'remplissez tous les champs';
}
elseif($_POST['mdp'] != "2788223e73728d8339cbc5366945c90b0a394bfa6bafe1426cc5eb221fd36bba")
{
    echo 'mdp incorrect';
}
else
{
    $lat = htmlspecialchars($_POST['lat']);
    $lng = htmlspecialchars($_POST['lng']);
    $req = $bdd->prepare("INSERT INTO positions (lat, lng, idmoutton, datation, pos) VALUES (:lat, :lng, :idmoutton, NOW(), GeomFromText(:location))");
    $req->execute(array(
        'lat' => htmlspecialchars($_POST['lat']),
        'lng' => htmlspecialchars($_POST['lng']),
        'idmoutton' => htmlspecialchars($_POST['idmoutton']),
        'location' => 'POINT('. $lat. " ". $lng. ')'
    ));
    echo "processed";
    echo 'POINT('. $lat. " ". $lng. ')';
}
echo "done";
?>
```

La page de connexion



La page de connexion est constituée de deux fichiers :

- un fichier .html qui va gérer tout l'affichage de la page
- un fichier php qui va vérifier l'identifiant et le mot de passe et éventuellement rediriger vers l'espace client.

```
<form id="formulaire" action="check.php" method="post">
    <p>Username <input type="text" class="inputtext" name="user" id="user" > </p>
    <p>Password <input type="password" class="inputtext" name="pass" id="pass"> </p>
    <input type="submit" value="Connexion" id="login_button"/>
</form>
```

Ce formulaire envoie une requête à la page check.php qui va la traiter : il se connecte à la BDD, récupère les variables, encrypte le mot de passe et prépare la requête **SELECT email,userid FROM auth WHERE login = :id AND mdp = :mdp**. Celle-ci vérifie qu'il existe bien un identifiant et un mot de passe correspondants dans la BDD. Si le mot de passe et l'identifiant sont corrects, le programme démarre la session et redirige l'utilisateur vers la page d'accueil.

```

<?php
/*
* check.php
*/
$bdd = new PDO('mysql:host=localhost;dbname=maxime1_favier;charset=utf8', 'root',
'root');

$id = htmlspecialchars($_POST['user']);
$mdp = htmlspecialchars($_POST['pass']);

$pass_hache = hash('sha256', $_POST['pass']);

$req = $bdd->prepare('SELECT email,userid FROM auth WHERE login = :id AND mdp =
:mdp');
$req->execute(array(
    'id' => $id,
    'mdp' => $pass_hache));
$resultat = $req->fetch();

if ($resultat)
{
    session_start();
    $_SESSION['iduser'] = $resultat['userid'];
    $_SESSION['email'] = $resultat['email'];
    header('Location: user/home.php');

}
else{
    //echo 'Mauvais identifiant ou mot de passe !';
    header('Location: ESheepLogon.html');
}
?>

```

La page d'accueil

The screenshot shows a web browser window titled "E-Sheep" with the URL "172.16.252.213/user/home.php". The page is titled "e-Shepherd" and has a sub-header "Stay in touch with your livestock". It features a navigation bar with icons for home, help, location, and power. Below the header is a section titled "Welcome to e-sheep application" with a subtitle "Control pannel". A table lists sheep ID, last refresh, status, and history. The status column includes links labeled "click here". The background features a globe icon and a silhouette of a cow.

Sheep ID	Last refresh	Status	History
1	2018-02-19 18:09:39	Sheep outside the field	click here
10	2018-05-09 14:45:10	Sheep inside the field	click here
20	2018-03-21 15:13:45	Sheep outside the field	click here
30	2018-03-21 15:14:31	Sheep outside the field	click here
45	2018-01-31 16:17:54	Sheep outside the field	click here
55	2018-01-31 16:13:14	Sheep outside the field	click here

Cette page affiche les informations importantes du troupeau :

- Les animaux en dehors du champ,
- La date de dernière actualisation,

Elle propose des liens vers toutes les autres pages, qui permettent de configurer et de voir la position des bêtes sur une carte.

Je commence par initialiser ma bibliothèque qui permet de calculer si un point est à l'intérieur d'un polygone, puis je me connecte à la base de données. Je fais deux requêtes dans cette page : une qui va récupérer les coordonnées des sommets du champ dans la table "area" et les mettre dans une liste, l'autre qui va récupérer toutes les autres informations les plus récentes pour chaque animal. J'utilise une boucle qui parcourt la liste obtenue pour générer le code HTML pour chaque ligne du tableau

```
<?php
// récupération des sommets du champ
$champ = $bdd->query('select `lat`, `longi` from area');
$limite = array();
while($arra = $champ->fetch())
{
    // convertir en une liste
    array_push($limite, ''.$floatval($arra['lat']).' '.$floatval($arra['longi']).'');
}

$champ->closeCursor();

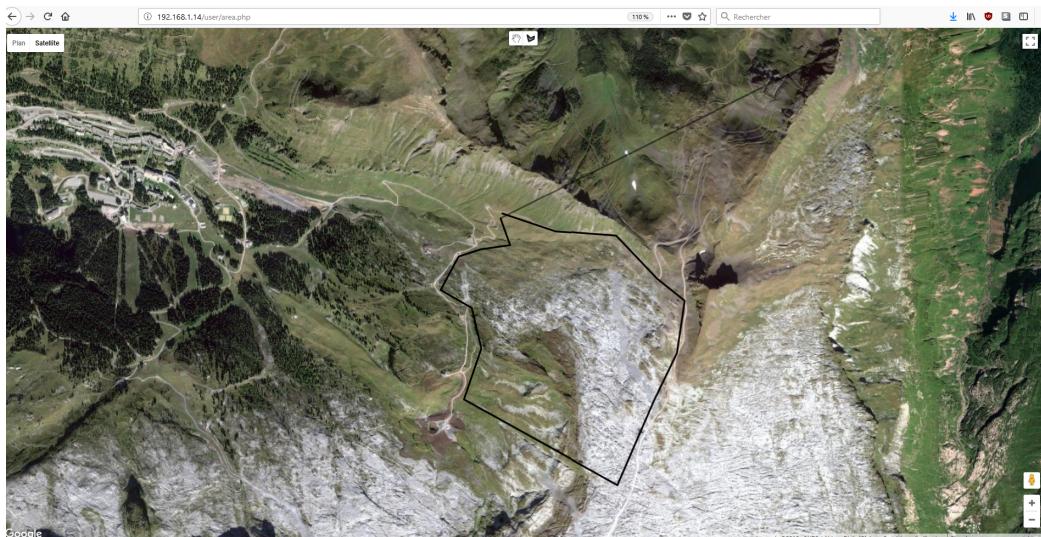
// récupération des dernières info du mouton
$reponse = $bdd->query('select `lat`, `lng`, `idmoutton`, max(`datation`) from')
```

```

positions group by `idmoutton`);
    while ($donnees = $reponse->fetch())
    {
        // boucle de génération du tableau
    ?
        <tr>
            <td><?php echo $donnees['idmoutton']; ?></td>
            <td><?php echo $donnees['max(`datation`)]; ?></td>
                <td><?php echo "Sheep " . $pointLocation->pointInPolygon("") .
$donnees['lat']."' . $donnees['lng']."' . $limite) . " the field" ?></td>
                    <td><a href='history.php?idmoutton=<?php echo $donnees['idmoutton']; ?>'>click here</a></td>
                </tr>
            <?php
        }
        $reponse->closeCursor();
    ?

```

La page de configuration du champ



Cette page permet à l'agriculteur de définir les limites du champ afin d'être notifié quand le bétail sort de cette zone.

Pour insérer la carte dans la page j'utilise l'API Google map. Je définis toutes les propriétés de celle-ci dans la fonction `initialize()` (niveau de zoom, centre de la carte, option de dessin pour définir le polygone...).

Je définis ensuite deux événements:

- un pour charger la carte au chargement de la page et
- un autre qui appelle la fonction `save_coordinates_to_array` quand l'utilisateur a fini de tracer les limites du champ. Cette fonction va récupérer les coordonnées de chaque point et les mettre dans une liste. Cette liste, grâce à la fonction `postToURL(a, b, c)` est mise dans un formulaire caché aux yeux de l'utilisateur et est envoyée au serveur.

De la même manière que pour le listener, le serveur va récupérer la liste et l'insérer dans la base de données après avoir supprimé l'ancien contenu de la table "area"

```

<script src="https://maps.googleapis.com/maps/api/js?key=AI[REDACTED]&libraries=drawing"></script>

<script>
//This variable gets all coordinates of polygon
var coordinates = [];
//This variable saves the polygon.
var polygons = [];
</script>

<script>
//save latitude and longitude to the polygons[] variable
function save_coordinates_to_array(polygon)
{
    polygons.push(polygon);
    var polygonBounds = polygon.getPath();
    for(var i = 0 ; i < polygonBounds.length ; i++)
    {
        coordinates.push(polygonBounds.getAt(i).lat(), polygonBounds.getAt(i).lng());
    }
    console.log(coordinates);
    postToURL("areasubmit.php", "array1", coordinates);
}
</script>

[.....]

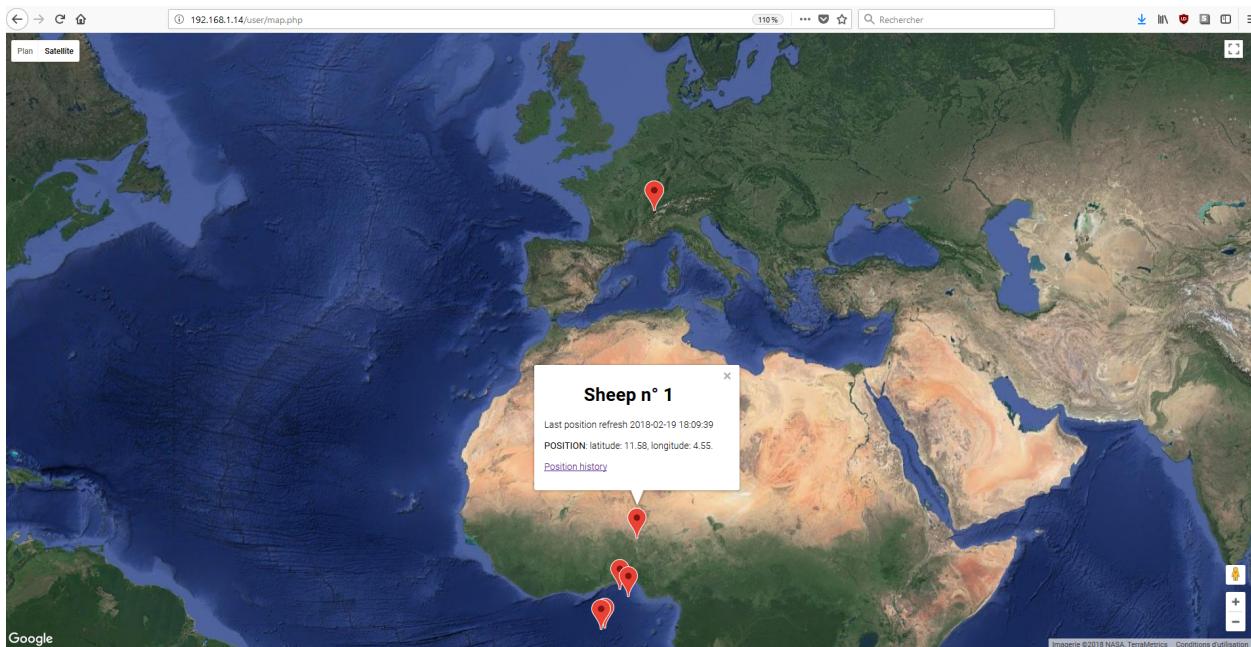
<div id="Carte"></div>

<form id="zone" action="areasubmit.php" method="post">
<input id="array1" type="hidden" name="array1" value="a">
</form>

<script>
function postToURL(a,b,c){
    document.getElementById("zone").action= a;
    document.getElementById("array1").name = b;
    document.getElementById("array1").value = c;
    document.getElementById("zone").submit();
}
</script>
[....]
<script>
function initialize()
{
    //Create map.
    var map = new google.maps.Map(document.getElementById('Carte'), {zoom: 15, mapTypeId: 'satellite', center: new google.maps.LatLng(46.0001577, 6.7202198, 424)});
    //Create drawing manager panel
    var drawingManager = new google.maps.drawing.DrawingManager({
        drawingControlOptions: {
            position: google.maps.ControlPosition.TOP_CENTER,
            drawingModes: ['polygon']
        });
    drawingManager.setMap(map);
    // event creation of polygon completed
    google.maps.event.addListener(drawingManager, 'polygoncomplete', function(polygon) {
        polygon.setEditable(false);
        save_coordinates_to_array(polygon);
    });
}
google.maps.event.addDomListener(window, 'load', initialize);

```

La page de visualisation des positions



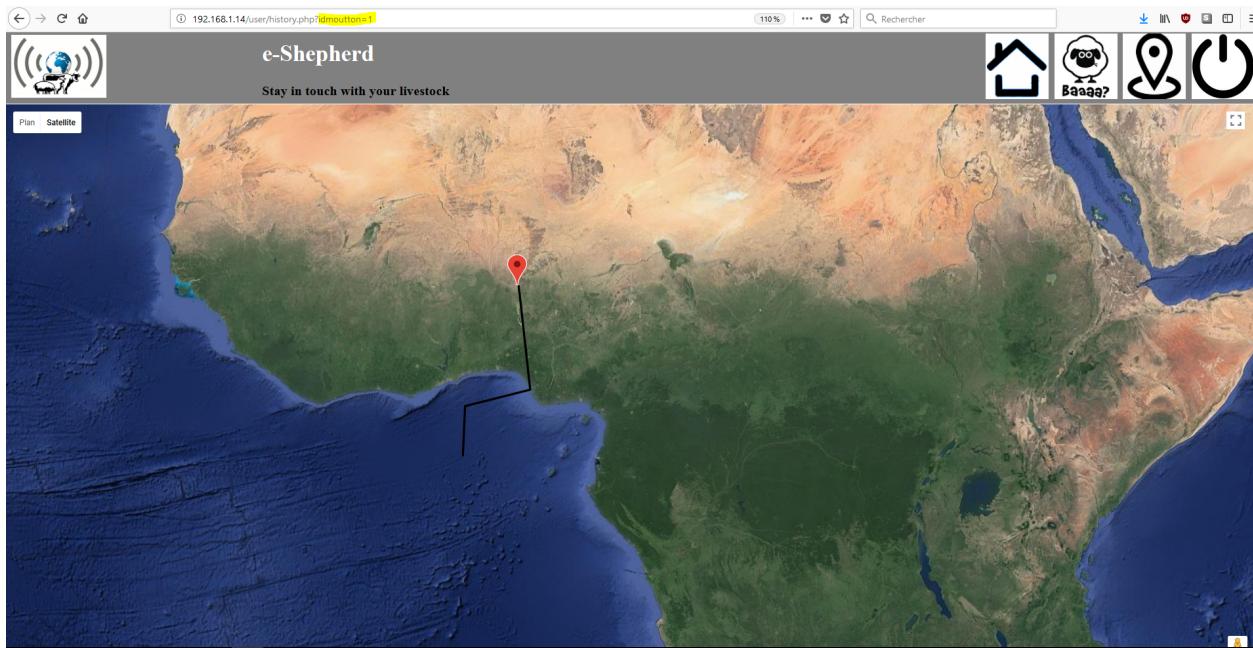
Cette page va afficher la position des animaux en temps réel. Pour ce faire, comme pour la page d'accueil, le PHP va récupérer les dernières positions du bétail et les sommets du champ. Les informations des animaux sont sous forme d'une liste de dictionnaire. Cette liste est parcourue pour ajouter à la carte les points et les info-bulles.

```
for( var i = 0, I = tableauMarqueurs.length; i < I; i++ ) {
    // tableau de lat lng d'un seul point + info
    var latlng = tableauMarqueurs[i],
        latitude = latlng["lat"],
        longitude = latlng["lng"];
    title = latlng["title"];
    refresh = latlng["refresh"];

    // définition et ajout des points sur la carte
    var optionsMarqueur = {
        map: maCarte,
        position: new google.maps.LatLng( latitude, longitude),
        title: "Sheep n° "+ title,
    };
    var marqueur = new google.maps.Marker( optionsMarqueur );
    zoneMarqueurs.extend( marqueur.getPosition() );

    (function(marker, data) {
        // Attaching a click event to the current marker
        // source : stack exchange
        google.maps.event.addListener(marker, "click", function(e) {
            infowindow.setContent(data);
            infowindow.open(maCarte, marker);
        });
    })(marqueur, tableauMarqueurs[i]["content"]);
}
```

La page d'historique des déplacements



Cette dernière page va afficher l'historique des position de l'animal choisi. Cette page prend le numéro de l'animal en argument dans l'URL. J'ai repris le même code que la page précédente en modifiant la requête pour ne retourner que les points correspondant à l'identifiant de l'animal au lieu des derniers points reçus pour tous les animaux.

```
<?php
$req = $bdd->prepare('SELECT * FROM `positions` WHERE idmoutton=:idmoutton');
$req->execute(array(
    'idmoutton' => htmlspecialchars($_GET['idmoutton']),
));
while ($donnees1 = $req->fetch())
{
?>
{ lat:<?php echo $donnees1 ['lat'];?>,  lng:<?php echo $donnees1 ['lng'];?>,
<?php
}
$req->closeCursor();
?>
```

Conclusion

Ce projet m'a permis d'acquérir de nombreuses nouvelles connaissances notamment :

- dans l'utilisation de Google Map sur une page,
- sur les bases de données et
- sur l'utilisation de Debian et les logiciels libres du monde professionnel.

Mais il reste encore des améliorations à faire. Il faudrait:

- remplacer l'API Google Map, qui deviendra payant le mois prochain, par son équivalent libre et gratuit Openstreetmap mais ceci demandera d'utiliser un serveur plus puissant et plus d'espace de stockage.
- envoyer des notifications sur le téléphone de l'éleveur pour qui n'ait pas en permanence à actualiser la page pour savoir si ses animaux ne sont pas sortis du champ
- trouver des solutions pour que le système fonctionne sans accès à internet tout en étant autosuffisant en énergie.

Bien que seul sur le projet, j'ai utilisé des outils collaboratifs tels que Git. J'ai aussi utilisé les forums stackoverflow pour poser des questions .