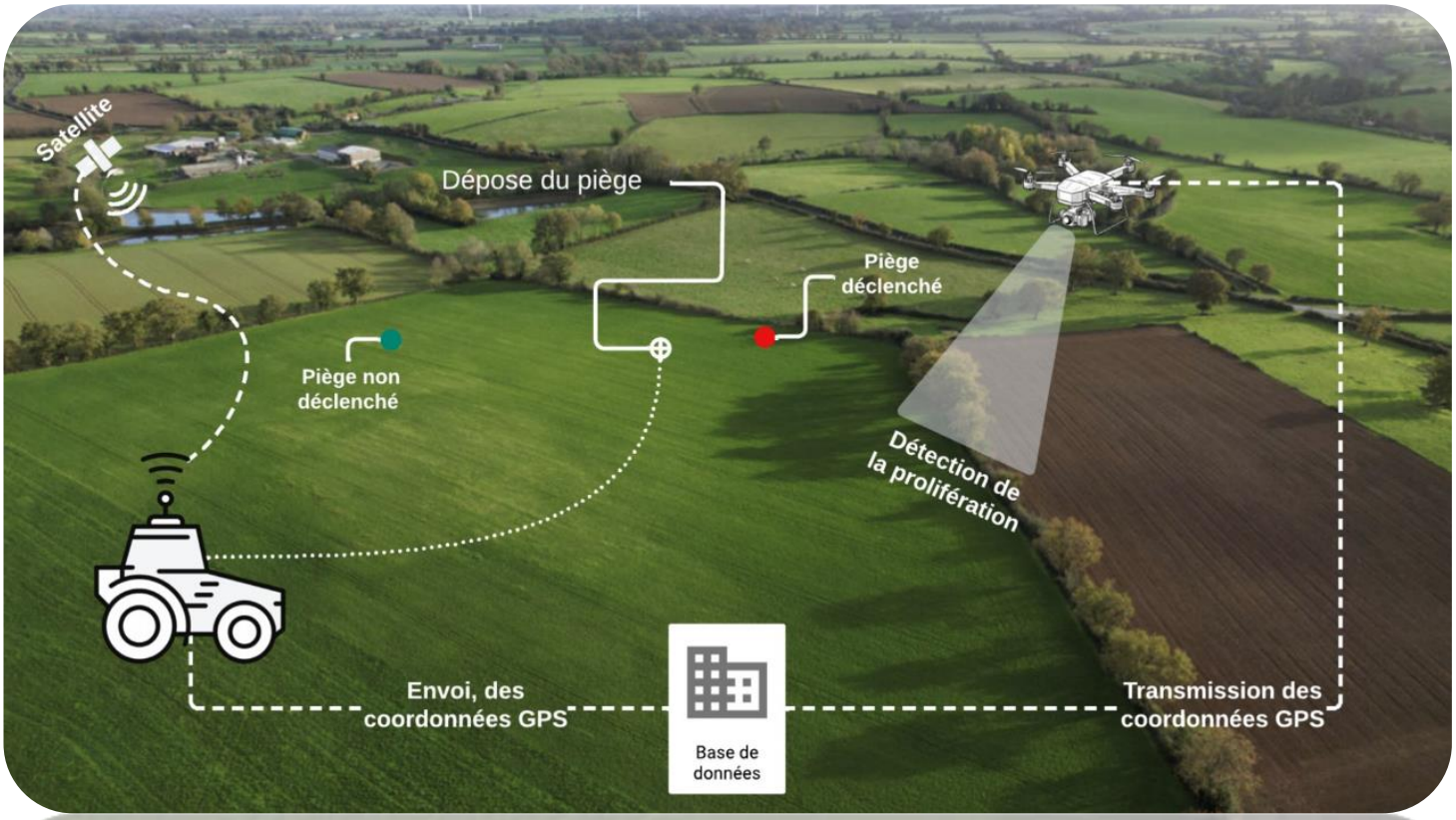
RAPPORT DE PROJET DE FIN D'ANNEE

PROJET ROBOCAT

EN COLLABORATION AVEC INRAE



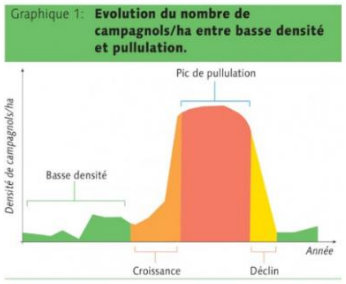
Etablissement : Lycée Monnet-Mermoz Aurillac *(15000)*
Enseignant : VIGUIER Christophe
Etudiants : GAILLARD Mathéo, FONTA Maxime, PHILIPPE Mehdi

SOMMAIRE :

Présentation générale du système supportant le projet :

La régulation des campagnols terrestres est un enjeu pour l'agriculture. La population de campagnols nuit au développement des prairies lorsque celle-ci est trop importante. Aujourd'hui la régulation est réalisée par du piégeage manuel.

L'INRAE de Clermont-Ferrand travaille sur une solution autonome pour réguler les populations de rongeurs et réduire ou supprimer les pics de population (figure ci-contre).



A ce titre, ils ont créé le projet **ROBOCATS** :
La robotique sera au service du contrôle de la prolifération des campagnols terrestres. Ce projet repose sur l'association entre un drone qui, par le survol des parcelles, localise automatiquement les foyers actifs pour transmettre leurs emplacements (coordonnées GPS) à un robot terrestre. Celui-ci aura pour objectif d'aller déposer des pièges et/ou des appâts dans des galeries tout en optimisant ses déplacements.

Description du projet de recherche de l'INRAE :
... Enfin, l'INRAE a présenté à l'ensemble de ses partenaires qui travaillent de près ou de loin sur le sujet des campagnols, l'avancement des travaux de recherche sur la robotisation de la lutte contre les campagnols « le ROBOCATS », pour lequel Monsieur Christian MUNIER, Vice-Président de FREDON AURA est partie prenante. Cela fait maintenant quelques années que nous soutenons dans ce projet Madame Myriam Chanet, Scientifique à l'INRAE (Equipe Robotique et mobilité pour l'environnement et l'agriculture (ROMEA). Par notre implication, nous avons permis le rapprochement des chercheurs de l'INRAE et de nos collègues de FREDON Bourgogne-Franche-Comté qui travaillait de leur côté au même objectif avec le développement d'un bras articulé.

L'objectif de cette recherche est de pallier le manque de main d'œuvre sur les exploitations pour mener à bien les luttes collectives contre les campagnols terrestres en prairie. On peut espérer que cette recherche, quand elle aura abouti, pourra être déployée en arboriculture pour le campagnol provençal et le campagnol des champs en grandes cultures.

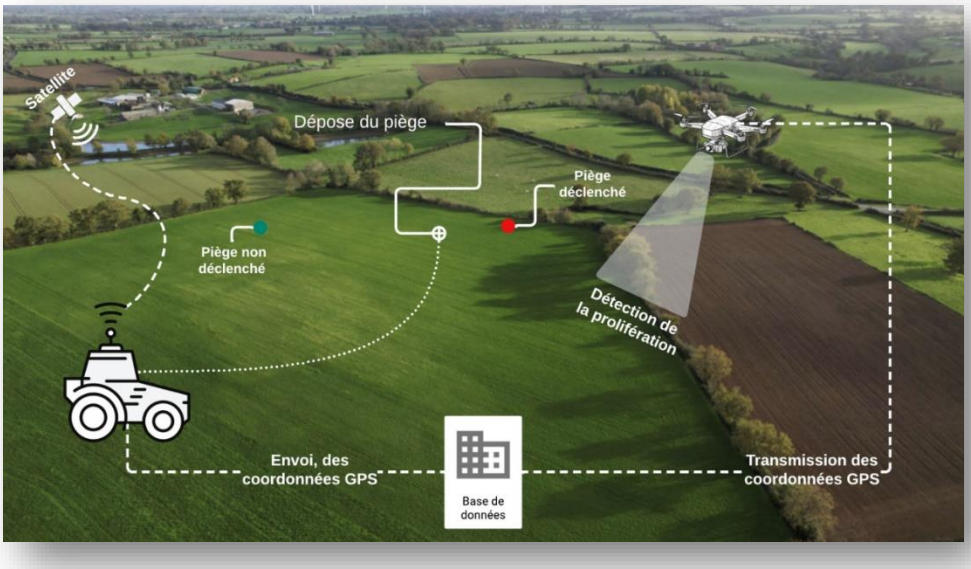
Myriam CHANET nous a présenté en salle, l'ensemble du projet. Le projet ROBOCATS est composé de deux options : un drone et un robot, robot adaptable au matériel existant sur l'exploitation.
L'objectif est que le drone établisse une carte de l'infestation de la parcelle par les campagnols.
Cela devrait permettre au robot de suivre une planification de mise en œuvre de la lutte, la rendant optimale en détectant les tumuli et galeries où il insérera pièges ou appâts, en toute autonomie.

Travail réalisé par les étudiants du BTS SN IR qui s'inscrira dans ce projet à deux niveaux :

1. Identification du déclenchement des pièges préalablement déposés par le robot et transmission de l'information à l'utilisateur. Ce dernier prendra la décision d'envoyer le robot pour les récupérer lorsqu'un certain nombre de pièges seront déclenchés ;
2. Création d'une application permettant d'informer l'utilisateur sur :
   - la cartographie des pièges déposés ;
   - la cartographie des pièges déclenchés ;
   - les difficultés éventuelles du robot ;
   - la nécessité d'aller relever les pièges ; ...

Descriptif du projet :

Il nous a été demandé de proposer le prototype d'un système permettant de recevoir une information du déclenchement d'un piège, de la traiter et de la transmettre à une application afin d'informer l'agriculteur de l'avancée du traitement de sa parcelle.
Celui-ci devra décider d'envoyer le robot afin de récupérer les pièges déclenchés.

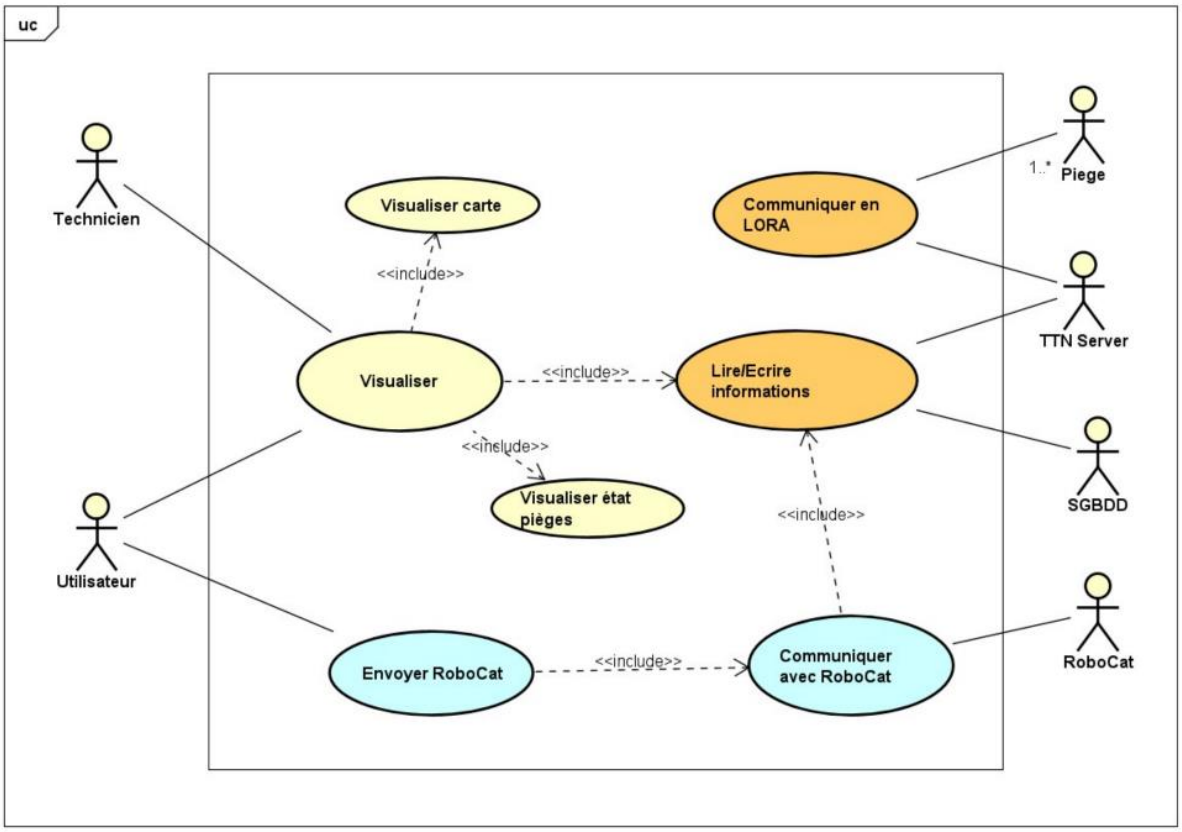


NB : Le déplacement du robot ne sera pas traité dans ce projet.

Expression du besoin :

Diagramme des cas d'utilisation :

Le système étudié s'inscrit dans un projet plus global. Le diagramme des cas d'utilisations ne représentera que la partie étudiée.



Description des acteurs/cas d'utilisation :

Acteur/use case	Rôle de cet acteur
Utilisateur	L'utilisateur est le propriétaire de la parcelle. Il suit l'avancée de la régulation et traite les alertes générées par l'application (envoi du robot).
ROBOCAT	Reçoit les coordonnées pour aller récupérer les pièges
Piège	Émet un signal LoRa lorsque celui-ci se déclenche.
Technicien	Le Technicien reçoit les informations de traitement des parcelles. Pourra établir des statistiques de traitement des différentes parcelles avec un historique.
TTN Server	Traite les données reçues par une passerelle et transmet sous format « ON/OFF » pour chacun des pièges à la base de données.
SGBDD	Stocke les données.

Solutions envisagées pour répondre aux besoins :

L'utilisation de matériel du commerce doit permettre la mise en œuvre de l'ensemble du système proposé. Pour cela le groupe projet devra utiliser un passerelle LoRa à disposition et réaliser l'acquisition de capteurs de type capteur de vibrations ELV pour LoRaWAN®.

Figure 1 : ELV LoRaWAN®

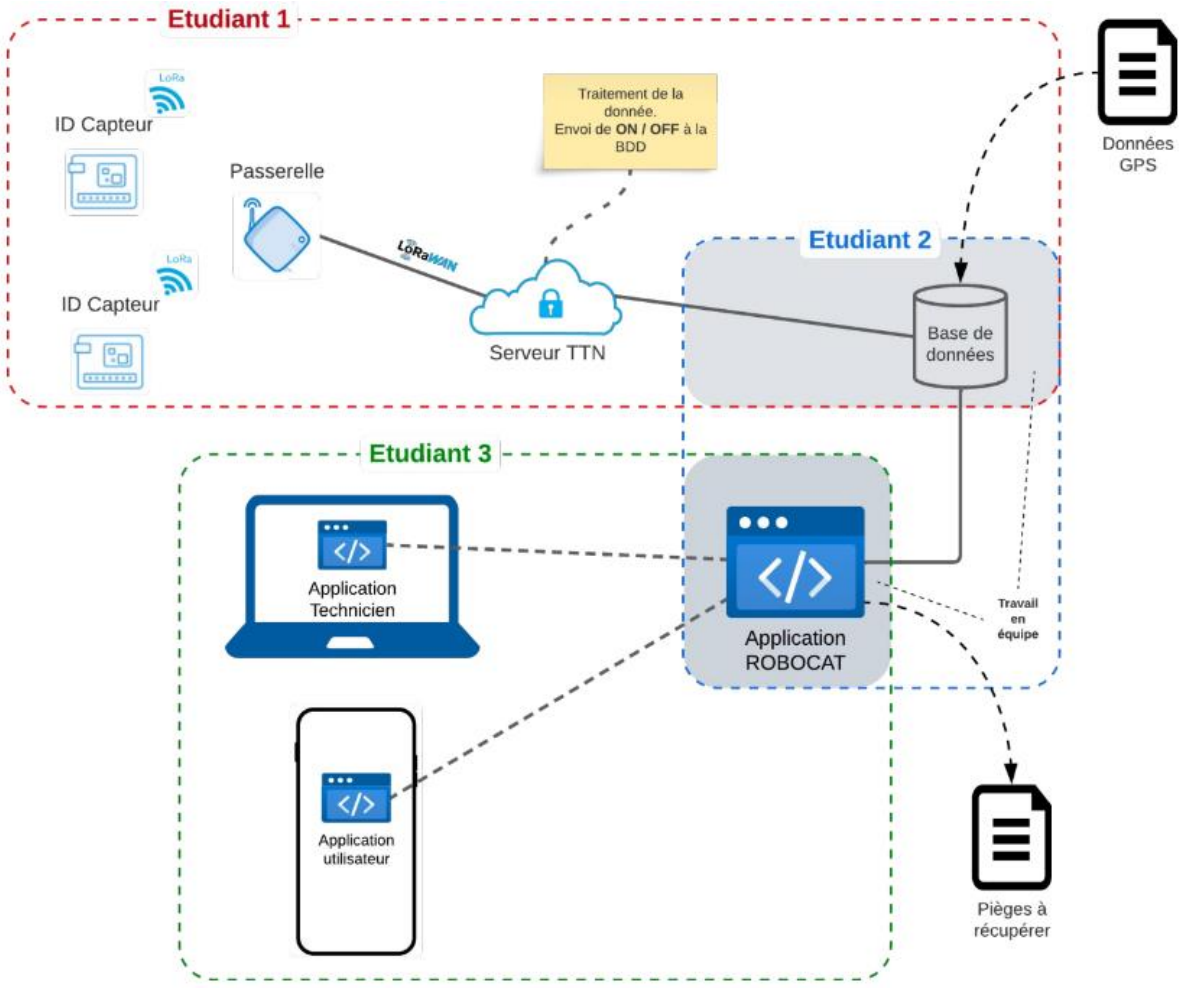
Il faut que le serveur TTN puisse envoyer à la BDD les informations contenant l'ID du capteur ainsi que son état (déclenché ou enclenché). Un protocole MQTT sera utilisé pour cette opération.

La base de données devra être en capacité de recevoir les données GPS d'implantation des pièges et donc la position GPS des différents capteurs LoRa.

La base de données devra pouvoir mettre à disposition des données afin de pouvoir les afficher à l'utilisateur d'une part puis au technicien d'autre part. Ils devront pouvoir visualiser sur une carte la position des pièges identifiés par des ronds qui seront verts lorsqu'ils sont enclenchés et rouges lorsqu'ils sont déclenchés. Le technicien devra être en capacité de réaliser des statistiques (à

définir avec l'INRAE). L'utilisateur devra pouvoir, l'aide de son application, valider l'envoi du robot pour aller récupérer les pièges.

Synoptique correspondant :

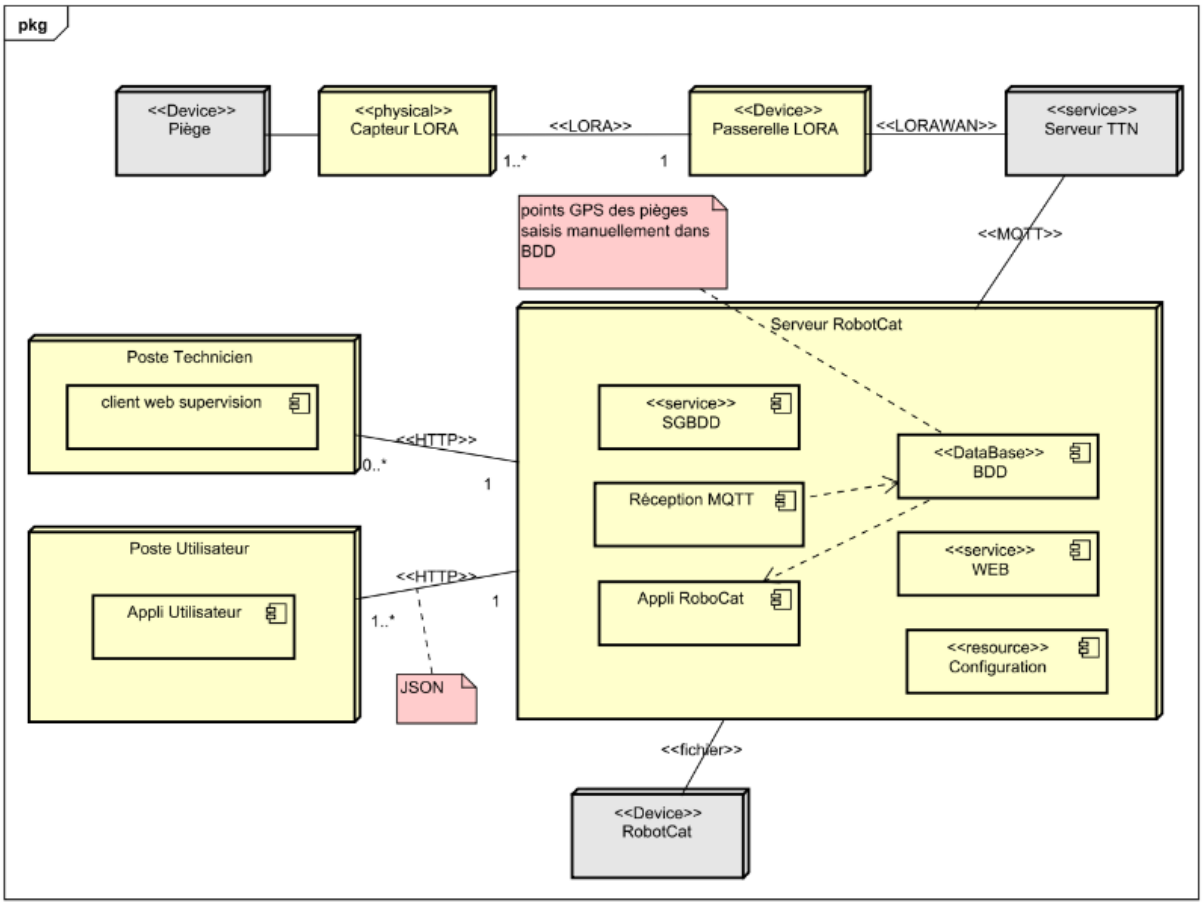


Contraintes technologiques :

- Communication avec ROBOCAT *(échange de fichiers)*
- Passerelle LoRa : utilisation d'un élément du commerce
- Serveur TTN
- Echange TTN serveur ROBOCAT en MQTT
- Gestion de base de données : MariaDB
- Application utilisateur : QT
- Application technicien : QT

Architecture du système :

- Diagramme de déploiement :



Description structurelle du système :

Principaux constituants :	Caractéristiques techniques :
Serveur TTN	Transmission d'une donnée 0 ou 1 associé à l'ID du capteur à la BDD.
Broker MQTT	Serveur MQTT public ou local
Service WEB	Serveur WEB public ou local
Application ROBOCAT	C++/QT, permet une collecte des différentes données pour les charger dans la BDD ou bien les lire afin de les mettre à disposition de l'utilisateur et du technicien.

	C++/QT, configure un fichier qui sera transmis et permettant de déclencher la récupération des pièges.
Process application utilisateur	Sur PC, du technicien, afficher les données liées au traitement d'une parcelle au travers d'une page WEB – **possibilité de réaliser des statistiques**.
Process application utilisateur	QT, Sur Smartphone de l'utilisateur affichage de la position des pièges et suivi de leur déclenchement. L'utilisateur doit pouvoir déclencher l'envoi du Robot pour récupérer les pièges.
ConfigRoboCat	Fichier texte de configuration du système.

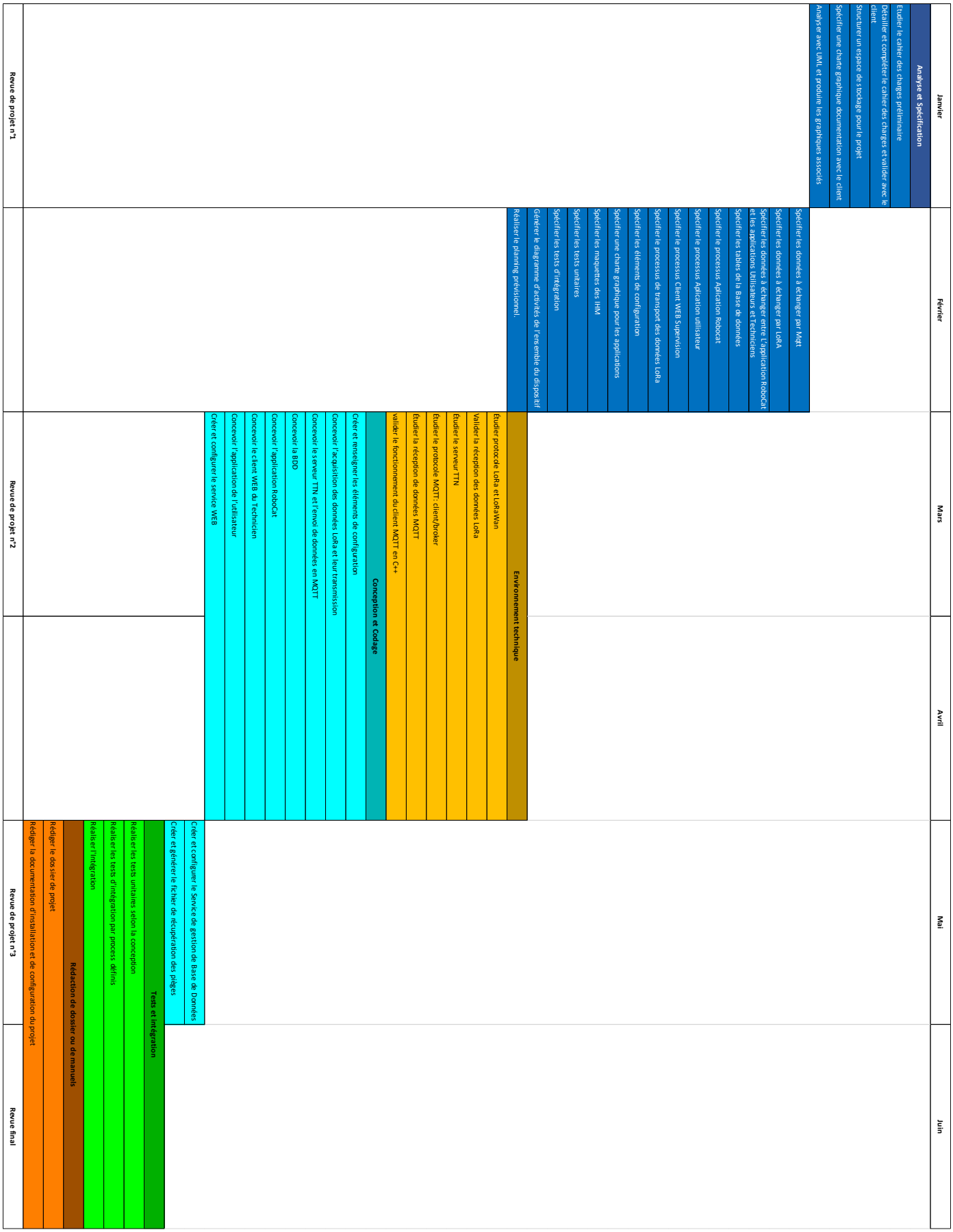
Énoncé des tâches à réaliser par les étudiants :

Etudiant	Partie	Système / Outils
N°1	Etude du protocole LoRa et MQTT Etude du serveur TTN Mise en œuvre d'une acquisition d'informations à partir d'un capteur LoRa et transmission puis structuration de l'information à une BDD suivant un protocole MQTT	- LoRaWan - MQTT - MariaDB
N°2	Création et mise en œuvre de l'application ROBOCAT Mise en œuvre de l'acquisition de données de la BDD à partir du protocole MQTT Configuration et déploiement d'un service WEB Configuration et déploiement d'un SGBDD	- C++/QT - MQTT - WEB
N°3	Création de l'application utilisateur Création de l'application client Web Supervision	- C++/QT - WEB - HTTP & JSON

	Création et mise en œuvre de l'application ROBOCAT Générer le fichier pour commander envoi récupération des pièges	

NB : Le projet n'étant pas figé, des tâches supplémentaires peuvent apparaître en cours de conception.

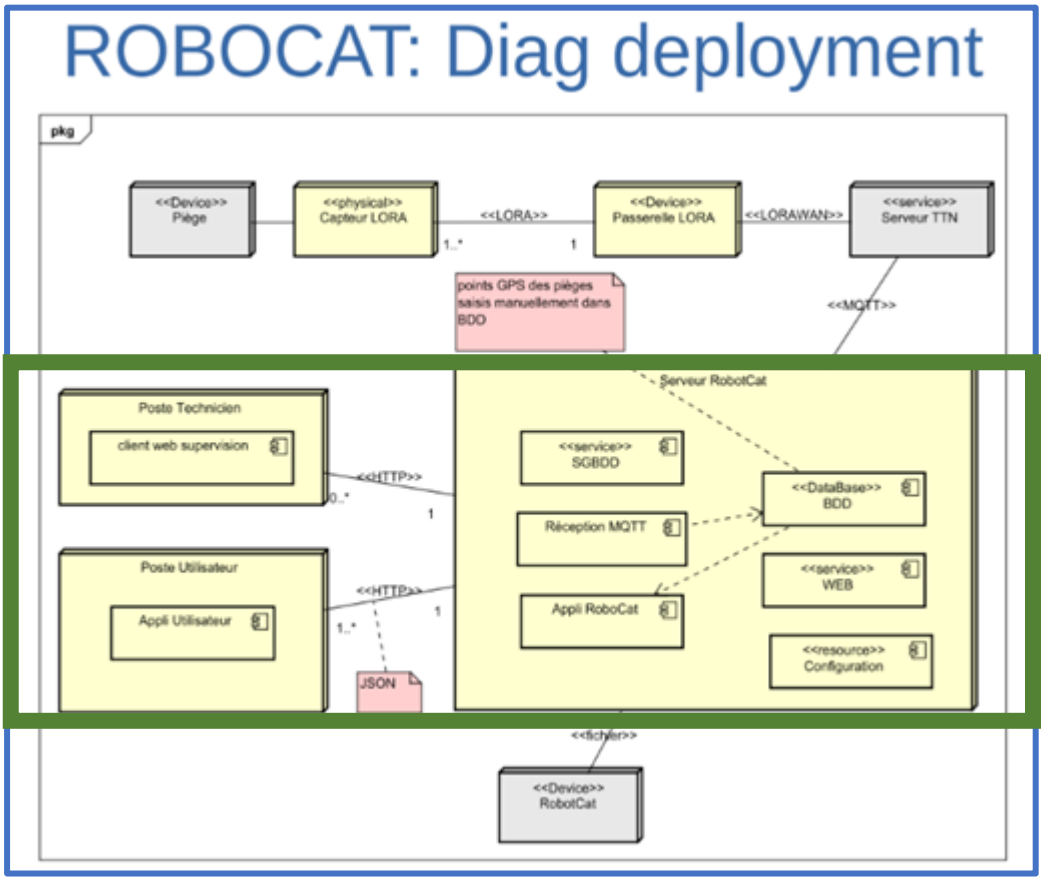
Calendrier Prévisionnel :



Partie étudiant (GAILLARD Mathéo)

Introduction :

Ma mission consiste à développer une version mobile et une version PC de l'application ROBOCAT Management, comprenant des interfaces pour les utilisateurs et les superviseurs. Je dois également mettre en place la communication entre une base de données et le site web. En outre, j'ai créé le fichier nécessaire pour ordonner l'envoi des pièges à récupérer. J'ai également configuré la passerelle LoRa pour qu'elle communique avec le service The Things Network, en utilisant une carte ESP32 LoRa avec l'aide de mon collègue Mehdi PHILIPPE.

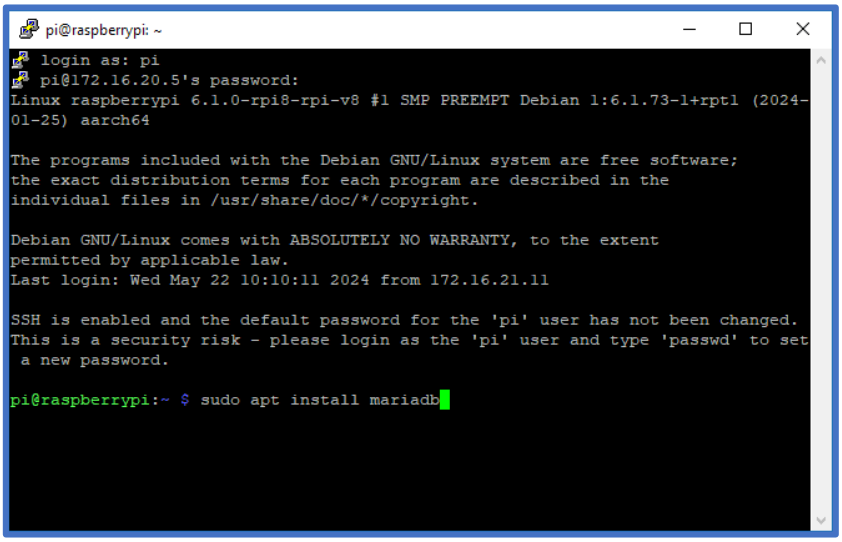


Mise en œuvre :

1. Raspberry Pi 4 :

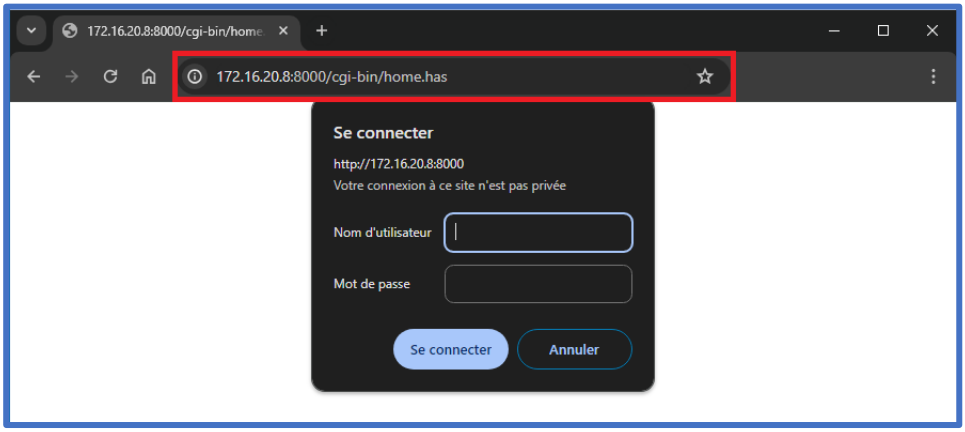
Avant tout chose, il était nécessaire de préparer la Raspberry en lui installant un système d'exploitation (OS) léger sans interface graphique car elle ne sera pas utilisée avec un écran. J'ai utilisé le logiciel « Raspberry

Pi Imager » (voir Annexe 2). Dès que le système est installé j'ai dû installer les services MariaDB et PHPMyAdmin (voir Annexe 1) pour pouvoir héberger la page Web et mettre la base de données. Cette installation a été faite à distance avec Putty car il est un client SSH open-source utilisé pour se connecter à des serveurs distants, notamment des serveurs Linux, depuis des ordinateurs Windows. Il permet aux utilisateurs d'établir des connexions sécurisées avec des serveurs distants pour exécuter des commandes à distance, transférer des fichiers et gérer des systèmes à distance.



2. Passerelle Dragino DLOS8 :

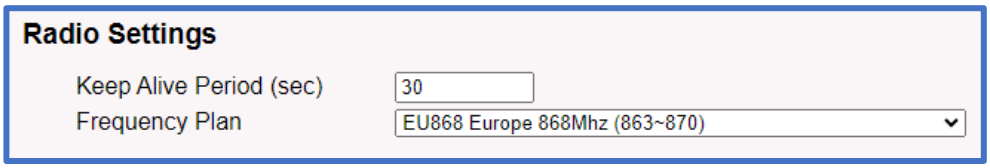
Pour commencer, j'ai dû brancher la passerelle LORA (Dragino DLOS8) au réseau Internet à l'aide d'un câble RJ45, elle pouvait également se connecter en Wi-Fi mais j'ai préféré utiliser le branchement par câble (plus stable). Ensuite je me suis connecté à la passerelle via son adresse IP que j'ai entré dans un navigateur Internet.



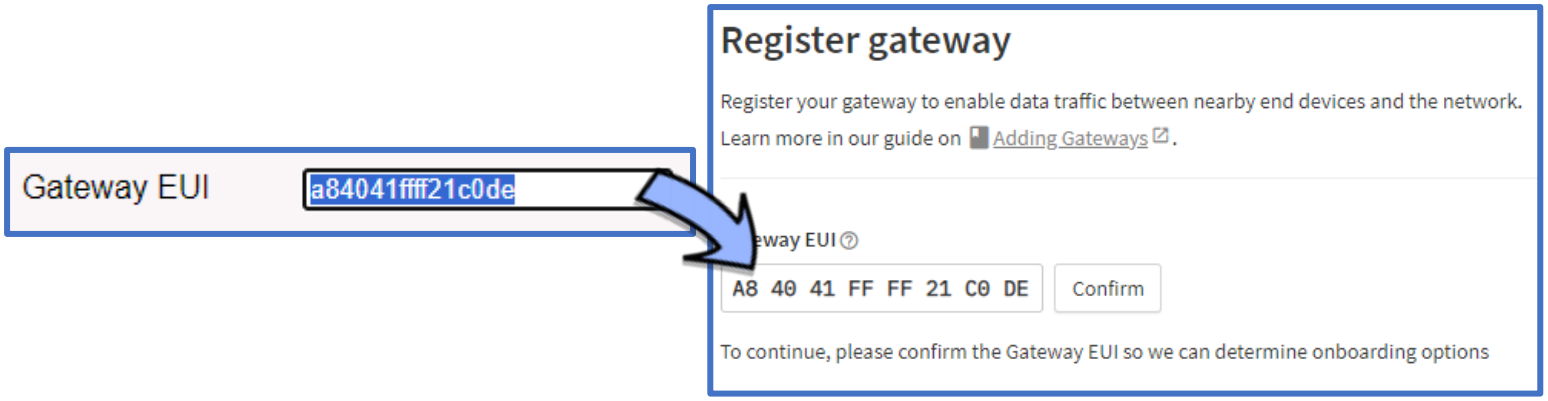
Il faut ensuite s'identifier en utilisant un nom d'utilisateur et un mot de passe qui par défaut sont les suivants :

- Nom d'utilisateur : *root*
- Mot de passe : *dragino*

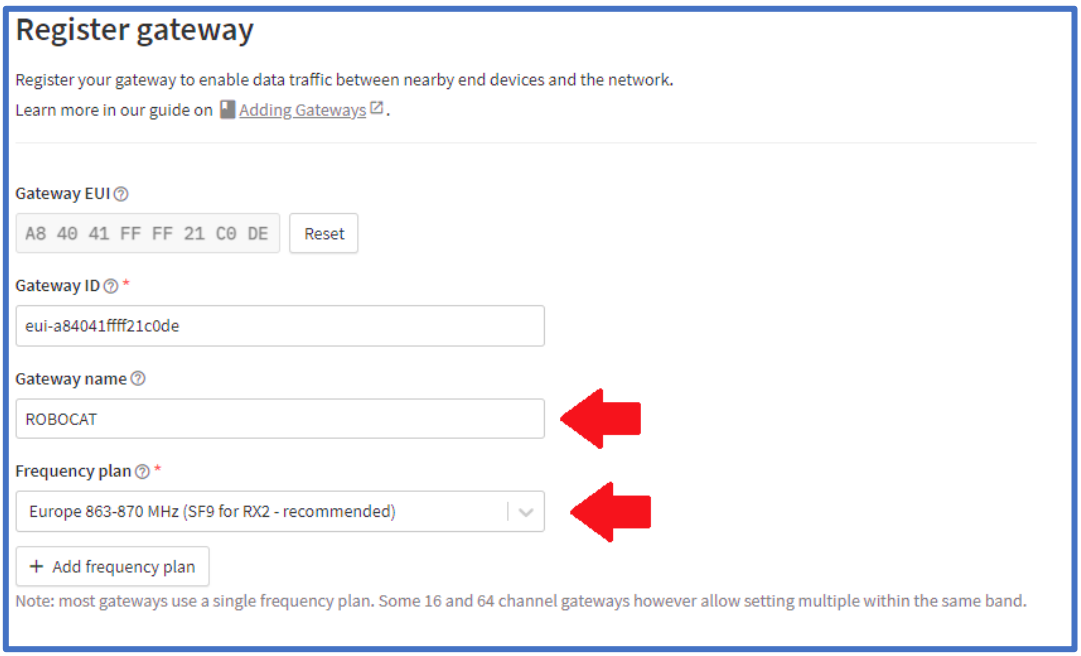
Le premier réglage nécessaire à faire est de sélectionner la bonne fréquence à utiliser pour la technologie LORA car chaque continent à sa propre fréquence qui lui est réservée, dans notre cas c'est la fréquence 868MHz qui est dédié à l'Europe.



Ensuite, le second réglage à faire est de récupérer l'EUI de la passerelle (Extended Unique Identifier, soit en français : identifiant unique étendu) pour les ajouter dans notre Gateway qu'on va créer sur TTN.

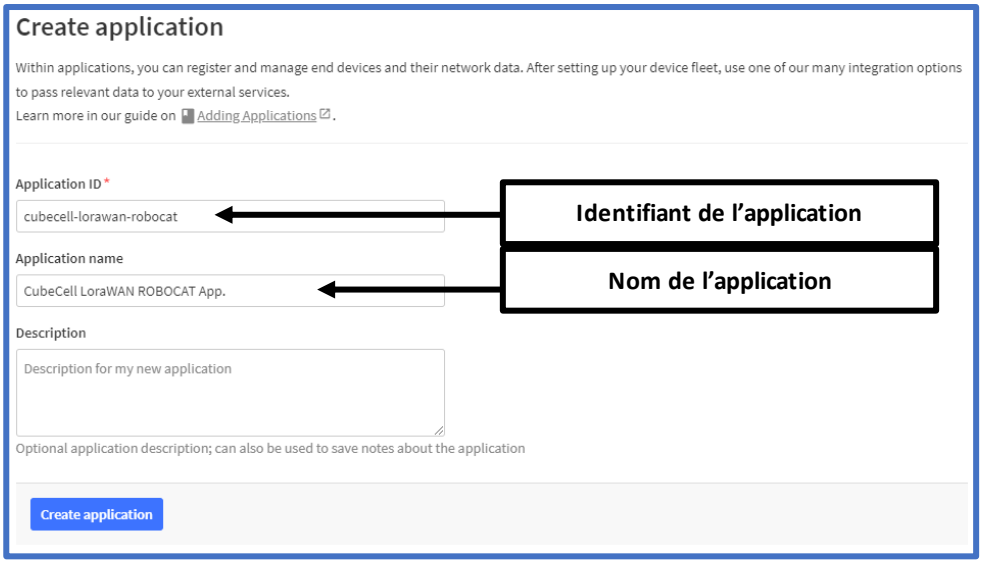


➔ Après il faut simplement compléter les dernières informations…

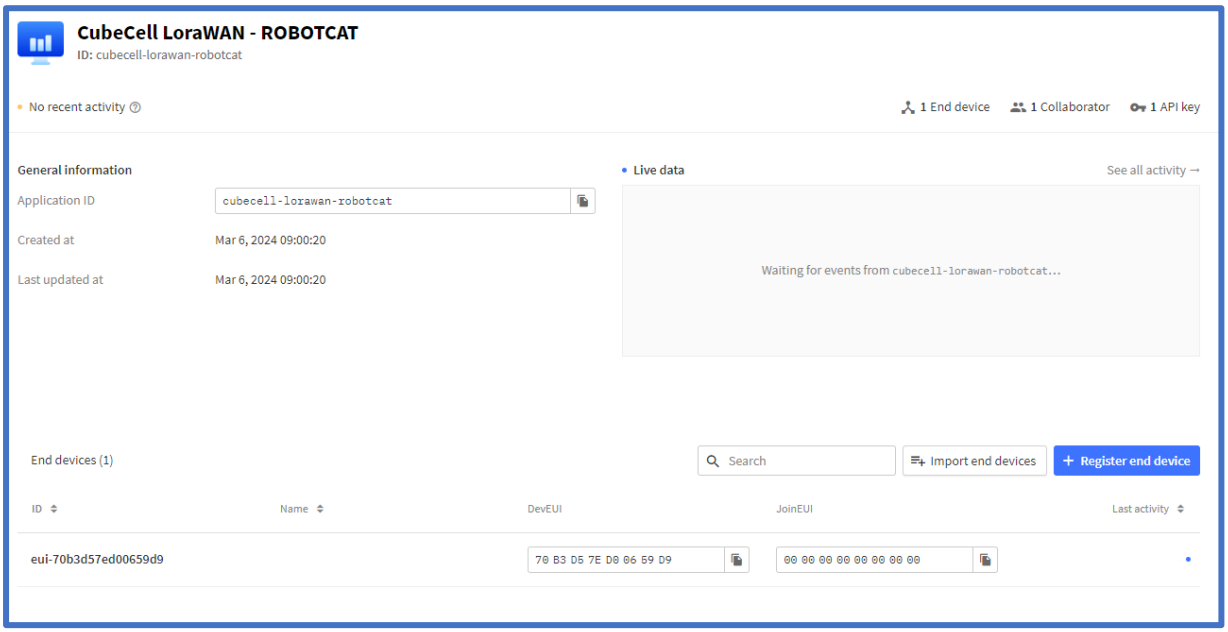


3. Ajout de la carte ESP32 CubeCell HTCC-AB02S sur The Things Network :

➔ Pour pouvoir connecter notre ESP32, il est obligatoire de créer une application sur TTN pour faire un lien :



➔ On se retrouve alors avec une application…



…dans laquelle on pourra ajouter du coup notre carte ESP32 (HELTEC Automation CubeCell HTCC-AB02S 868-928MHz) :

Il a été facile d'ajouter notre carte ESP32 dans TTN car celui-ci dispose d'une bibliothèque d'appareils qui utilise la technologie LoRaWAN.

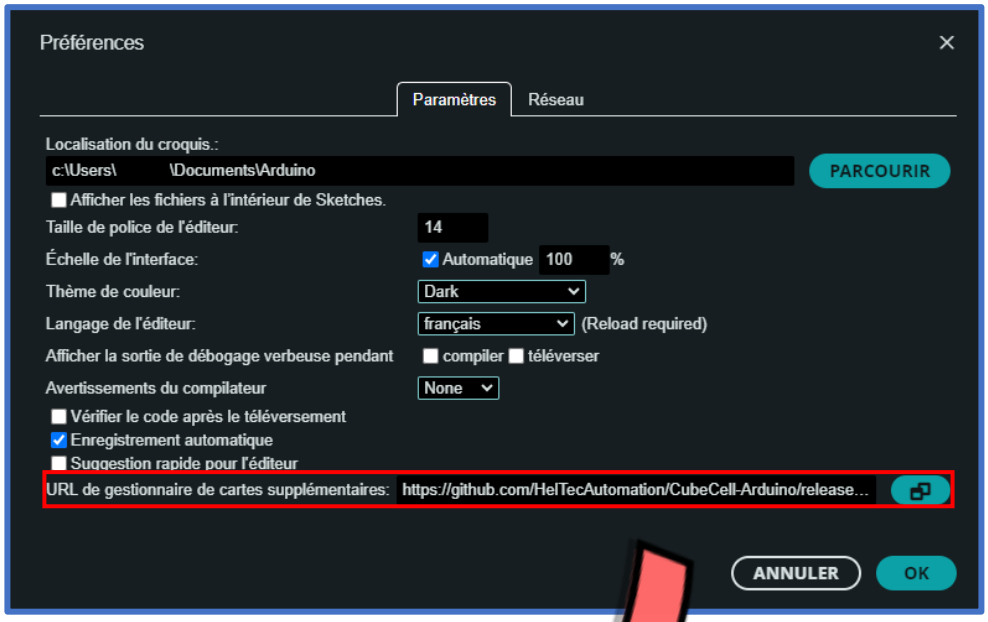
Il faut également de pas oublier de sélectionner la bonne fréquence !

Puis, nous devons générer des valeurs aléatoires qui seront utilisés pour récupérer des « Payloads » envoyé par la carte et la passerelle…

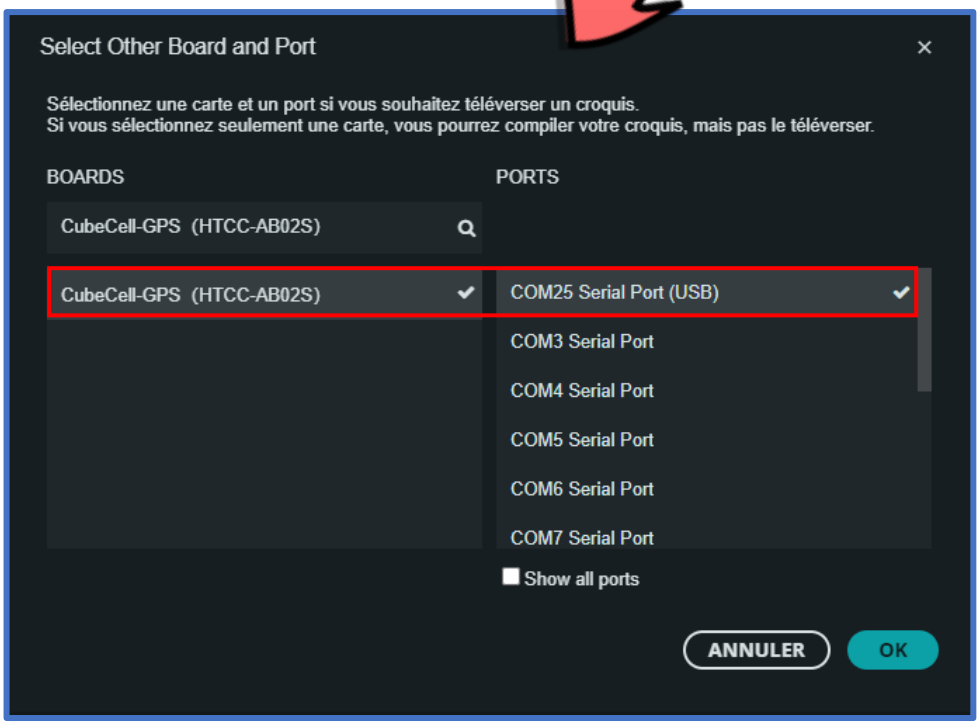
➔ ***NOTRE CARTE SERA DONC ENREGISTREE SUR TTN.***

4. Programmation de la carte ESP32 CubeCell HTCC-AB02S à l'aide d'Arduino :

➔ Pour pouvoir utiliser cette carte, j'ai dû utiliser le logiciel Arduino. Avant toute chose j'ai dû installer le composant propriétaire via un lien GitHub pour que la carte soit bien reconnue sur mon ordinateur.

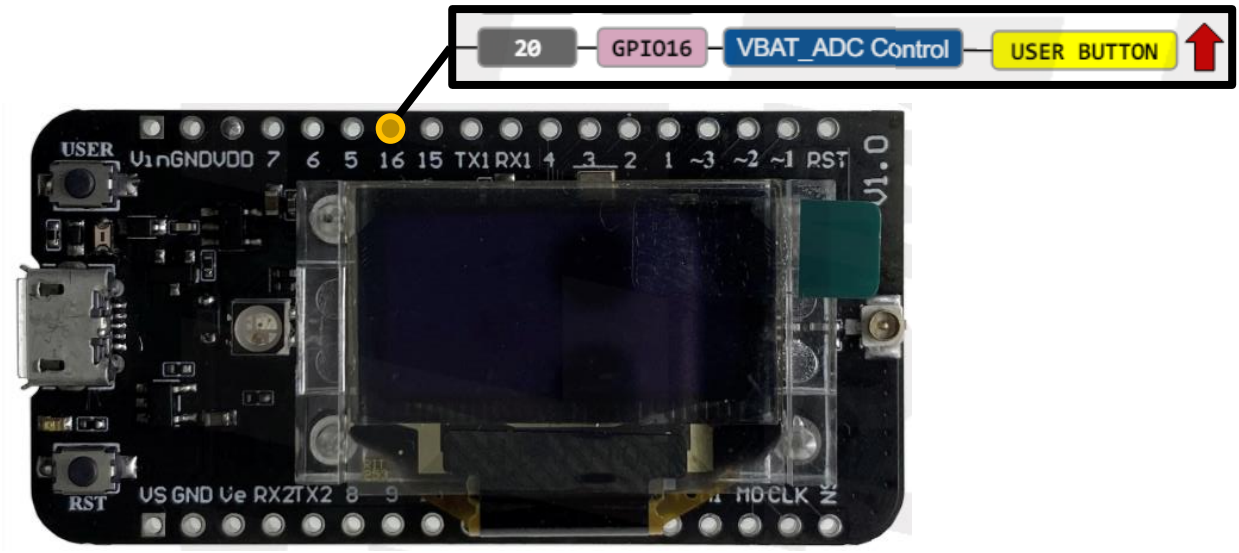


➔ **La carte sera donc sélectionnable et détectée par Arduino :**



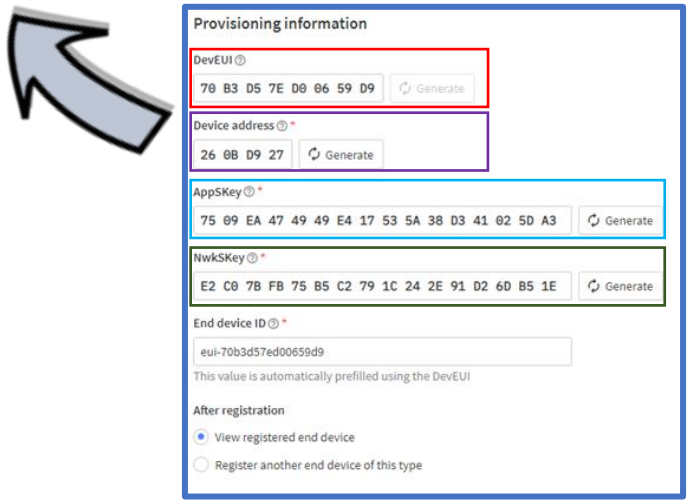
➔ Grâce à Arduino et au constructeur de la carte, j'ai pu utiliser un exemple de communication entre la carte ESP32 et TTN. J'ai analysé le code et je l'ai adapté pour qu'il soit utilisable dans notre cas : envoyer une valeur sur TTN pour savoir si un piège a été déclenché. Pour faire ce

changement d'état j'ai utilisé le bouton « USER » de la carte en codant le PIN, pour qu'un Payload soit envoyé seulement quand le piège est déclenché.



➔ Il fallait aussi ajouter dans le code des valeurs qu'on à précédemment générer avec TTN :

```
/* OTAA para*/
uint8_t devEui[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x06, 0x59, 0xD9 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appKey[] = { 0xCE, 0x8E, 0x0B, 0x0E, 0x20, 0xDF, 0x2A, 0xF3, 0x5B,
0x19, 0x7B, 0xE9, 0xE8, 0xEF, 0xA6, 0x2F };

/* ABP para*/
uint8_t nwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d,
0x11, 0x18, 0x1e, 0x1e, 0xc7, 0xda,0x85 };
uint8_t appSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf, 0x55,
0xee, 0x4a, 0x77, 0x8d, 0x16, 0xef,0x67 };
uint32_t devAddr =  ( uint32_t )0x007e6ae1;
```



Conclusion et état d'avancement :

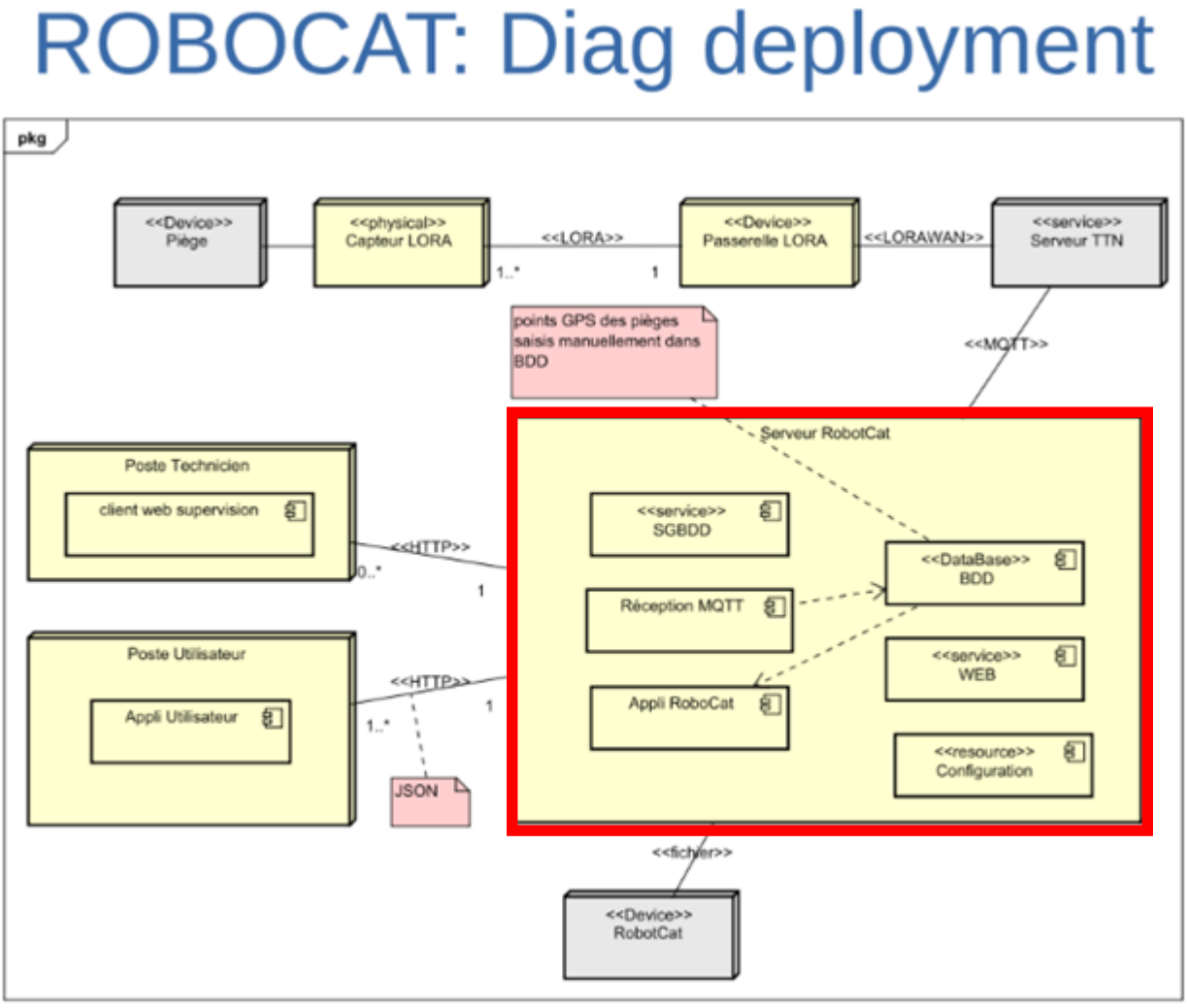
En conclusion, le projet a été un atout important pour découvrir de nouvelles technologies et de pouvoir travailler en équipe de manière optimale à l'aide du calendrier prévisionnel. J'ai pu découvrir de nouvelles technologies comme The Things Network, LoRa / LoRaWAN, Raspberry Pi OS, PuTTy et comprendre leur fonctionnement.

Pour ce qui est de l'état de l'avancement du projet c'est de pouvoir récupérer les données stockées sur TTN et de les mettre sur la base de données qui se trouve sur la Raspberry Pi pour mettre à jour la carte interactive.

Partie individuelle (FONTA Maxime)

Introduction :

Pour pouvoir réaliser ma partie, j'ai dû créer la base de données dans laquelle toutes les informations relatives aux différents pièges sont stockées. Ensuite, j'ai dû créer une page web sur laquelle sont affichés les différents pièges sur une carte, leur position étant indiquée à partir d'un fichier CSV, ainsi que leurs différentes informations.

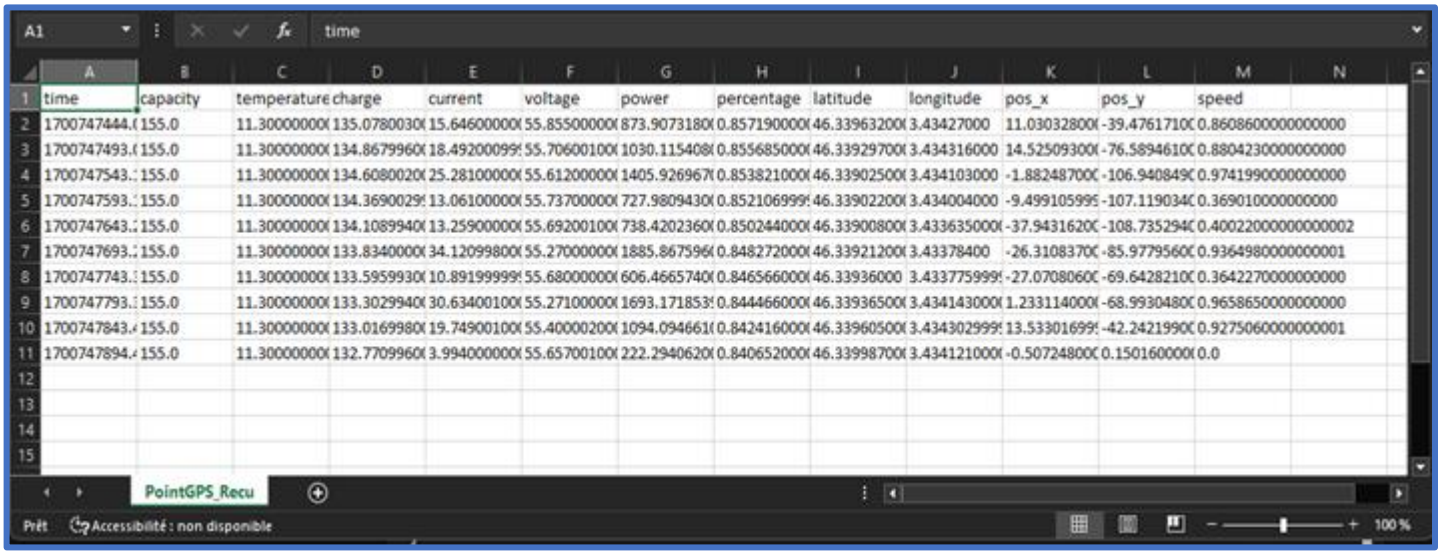


Mise en oeuvre :

1. Création et gestion de la base de données :

J'ai donc commencé par créer une base de données MySQL avec phpMyAdmin. Il m'a donc aussi fallu déterminer quels sont les différents champs dont on aura besoin pour stocker les différentes données relatives aux pièges. Les pièges envoient deux informations principales : un ID pour identifier chaque

piège et un état (déclenché ou non déclenché). Grâce à un fichier CSV, on a aussi la position de chaque piège, donnée par le robot qui les pose, avec la latitude et la longitude.



time	capacity	temperature	charge	current	voltage	power	percentage	latitude	longitude	pos_x	pos_y	speed
1700747444.(	155.0	11.30000000(	135.0780030(	15.64600000(	55.85500000(	873.9073180(	0.857190000(	46.33963200(	3.43427000	11.03032800(	-39.4761710(	0.8608600000000000
1700747493.(	155.0	11.30000000(	134.8679960(	18.49200099(	55.70600100(	1030.115408(	0.855685000(	46.33929700(	3.434316000	14.52509300(	-76.5894610(	0.8804230000000000
1700747543.:	155.0	11.30000000(	134.6080020(	25.28100000(	55.61200000(	1405.926967(	0.853821000(	46.33902500(	3.434103000	-1.88248700(	-106.940849(	0.9741990000000000
1700747593.:	155.0	11.30000000(	134.3690029(	13.06100000(	55.73700000(	727.9809430(	0.852106999(	46.33902200(	3.434004000	-9.499105995	-107.119034(	0.369010000000000
1700747643.:	155.0	11.30000000(	134.1089940(	13.25900000(	55.69200100(	738.4202360(	0.850244000(	46.33900800(	3.433635000(	-37.9431620(	-108.735294(	0.4002200000000002
1700747693.:	155.0	11.30000000(	133.8340000(	34.12099800(	55.27000000(	1885.867596(	0.848272000(	46.33921200(	3.43378400	-26.3108370(	-85.9779560(	0.9364980000000001
1700747743.:	155.0	11.30000000(	133.5959930(	10.89199999(	55.68000000(	606.4665740(	0.846566000(	46.33936000	3.433775999(	-27.0708060(	-69.6428210(	0.3642270000000000
1700747793.:	155.0	11.30000000(	133.3029940(	30.63400100(	55.27100000(	1693.171853(	0.844466000(	46.33936500(	3.434143000(	1.233114000(	-68.9930480(	0.9658650000000000
1700747843.(	155.0	11.30000000(	133.0169980(	19.74900100(	55.40000200(	1094.094661(	0.842416000(	46.33960500(	3.434302999(	13.53301699(	-42.2421990(	0.9275060000000001
1700747894.(	155.0	11.30000000(	132.7709960(	3.994000000(	55.65700100(	222.2940620(	0.840652000(	46.33998700(	3.434121000(	-0.50724800(	0.150160000(	0.0

Enfin, on retrouve la batterie du module.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	int			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
2	etat_piege	tinyint(1)			Non	*Aucun(e)*			Modifier Supprimer Plus
3	etat_batterie	int			Non	*Aucun(e)*			Modifier Supprimer Plus
4	latitude	float			Non	*Aucun(e)*			Modifier Supprimer Plus
5	longitude	float			Non	*Aucun(e)*			Modifier Supprimer Plus
6	etat_import	tinyint(1)			Oui	0			Modifier Supprimer Plus

L'ID est une clé étrangère (numéro unique) de type int (nombre entier) et est renvoyé par chaque piège via une requête SQL :

```
// Insérer les données dans la base de données $insertStmt = $bdd-
      >prepare("INSERT INTO robotcat_table (id,,etat_piege, etat_import) VALUES
(?, ?, 0)");
       $insertStmt->execute([$id, $latitude, $longitude, $etat_piege]);

echo "Équipement ajouté avec succès !";
```

`etat_piege` indique si le piège est actif ou non et est de type tinyint : soit 0 si le piège n'est pas déclenché, soit 1 si le piège est déclenché.

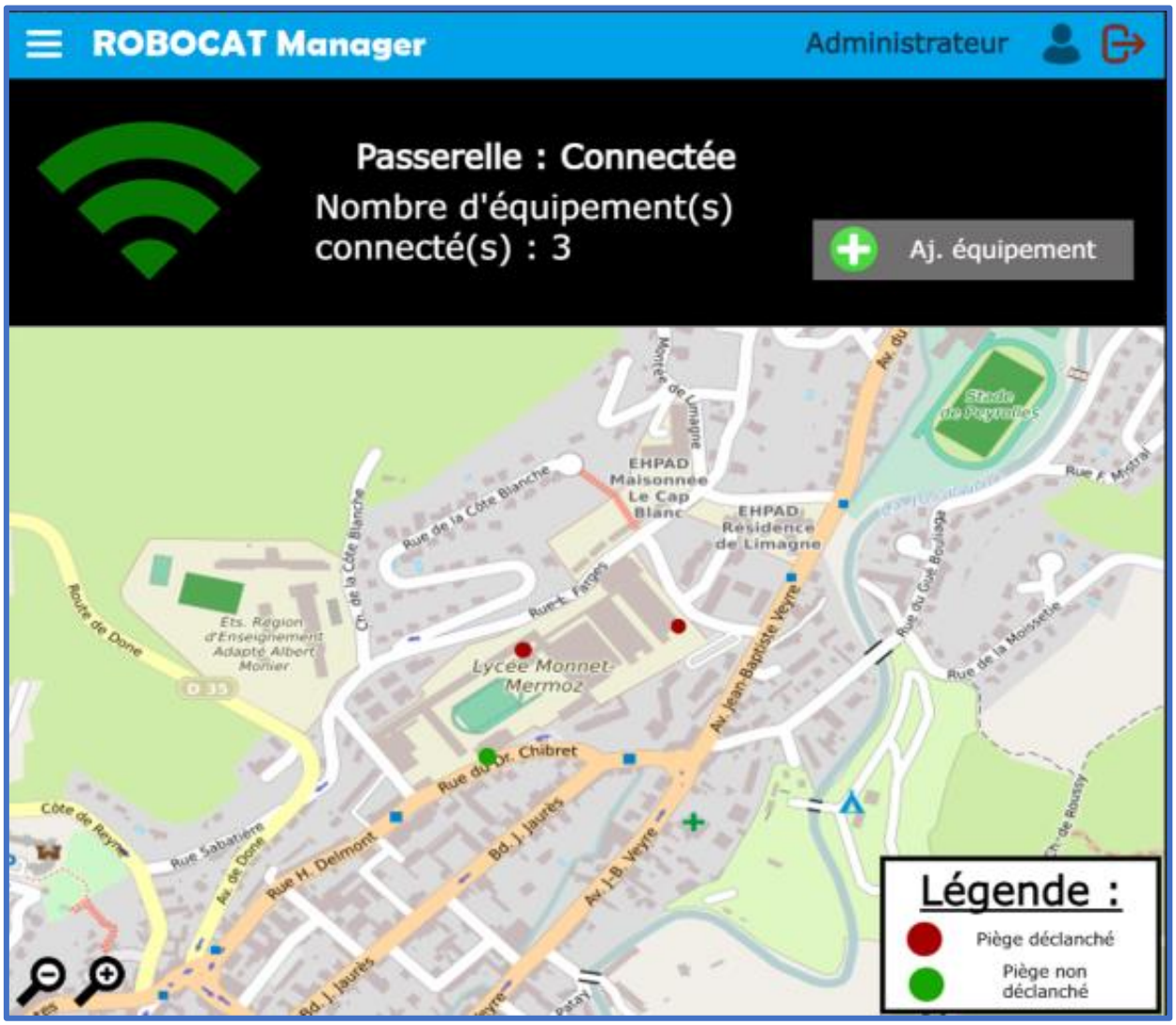
`etat_batterie` est un entier qui indique le pourcentage de la batterie (actuellement non opérationnel).

`longitude` et `latitude` sont de type float (nombre à virgule) et indiquent les coordonnées géographiques de chaque piège.

`etat_import` sert à afficher correctement sur la carte du site web l'état de certains pièges.

2. Création d'un site web :

Dans un second temps, j'ai dû réfléchir à un moyen de pouvoir visualiser les pièges et de pouvoir facilement voire où ils sont placés ainsi que les différentes informations les concernant. J'ai donc commencé par créer un visuel pour déterminer quels étaient les différents éléments à avoir.



Un des éléments est une carte sur laquelle les pièges sont affichés. Il m'a donc fallu faire des recherches sur les moyens d’implémenter une carte interactive et de l’utiliser :

Après plusieurs recherches, j'ai utilisé OpenStreetMap étant donné que le code pour l’utiliser est open source et que son utilisation ainsi que les ajouts que l'on peut y faire sont expliqués sur leur site.

```
<!DOCTYPE html>
<html>
<head>
    <title>Carte OpenStreetMap</title>
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"/>
    <style>
        #osm-map {
            height: 300px;
        }
    </style>
</head>
<body>
    <div id="osm-map"></div>
    <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
    <script>
        var element = document.getElementById('osm-map');
        var map = L.map(element);
        L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
            attribution: '© <a href="http://osm.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(map);
        var target = L.latLng('47.50737', '19.04611');
        map.setView(target, 14);
        L.marker(target).addTo(map);
    </script>
</body>
</html>
```

Après cela, j'ai créé un site-test en PHP pour apprendre à ajouter un point rouge sur la carte avec un clic de souris, grâce à des liens GitHub.

```
 // Créer un icône rouge
    var redIcon = new L.Icon({
      iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-red.png',
      shadowUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    var marker = L.marker(target, {icon: redIcon}).addTo(map);
    markers.push(marker); // Ajouter le marqueur à la liste

    // Ajouter un écouteur d'événements pour les clics sur la carte
    map.on('click', function(e) {
        var target = e.latlng;
        var newMarker = L.marker(target, {icon: redIcon}).addTo(map);
        markers.push(newMarker); // Ajouter le nouveau marqueur à la liste
    });

    // Définir la vue initiale
    var initialTarget = L.latLng('44.93711027433447', '2.4543199680414722');
    map.setView(initialTarget, 14);
```

Cette partie est en HTML. La partie en PHP m'a permis de connecter la base de données se trouvant alors en localhost. Je précise qu'à cette étape, le point n'est pas encore déterminé par la base de données mais à chaque clic de souris :

```php
<?php

$host = "localhost";
$dbname = "robo";
$username = "root";
$password = "";



try {

    $bdd = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    echo "Connected to $dbname at $host successfully.";
} catch (PDOException $pe) {
    die('Erreur de connexion à la base ' . $dbname . ': ' . $pe->getMessage());
}
?>
```

Ensuite, j'ai fait en sorte que l'on puisse ouvrir le fichier CSV, récupérer les données de latitude et longitude, et les inscrire dans la base de données.

```
  // Vérifier si le fichier CSV a déjà été traité
    $checkImportStmt = $bdd->query("SELECT COUNT(*) FROM robotcat_table WHERE
etat_import = 1");
    $imported = $checkImportStmt->fetchColumn();

    if ($imported == 0) {
        // Lire le fichier CSV et insérer les nouveaux points GPS uniquement si ce
n'est pas encore fait
        $filePath = 'C:\wamp64\www\html\page_principale\PointGPS_Recu.CSV';
        if (file_exists($filePath) && filesize($filePath) > 0) {
            $file = fopen($filePath, 'r');
            if ($file) {
                // Lecture de la première ligne (en-tête)
                $header = fgetcsv($file, 0, ';');
                if ($header !== FALSE) {
                    // Déterminer les indices des colonnes latitude et longitude
                    $latitudeIndex = 8;  // Colonne I
                    $longitudeIndex = 9; // Colonne J

                    // Préparation de la requête SQL pour vérifier l'existence des
points
                    $checkStmt = $bdd->prepare("SELECT COUNT(*) FROM
robotcat_table WHERE latitude = ? AND longitude = ?");
                    // Préparation de la requête SQL pour l'insertion
                    $insertStmt = $bdd->prepare("INSERT INTO robotcat_table
(latitude, longitude, etat_piege, etat_import) VALUES (?, ?, 0, 1)");

                    // Lecture du fichier CSV ligne par ligne
                    while (($row = fgetcsv($file, 0, ';')) !== FALSE) {
                        // Nettoyer et vérifier les valeurs de latitude et de
longitude
                        $latitude = isset($row[$latitudeIndex]) ?
trim($row[$latitudeIndex]) : '';
                        $longitude = isset($row[$longitudeIndex]) ?
trim($row[$longitudeIndex]) : '';

                        // Remplacer les virgules par des points si nécessaire
                        $latitude = str_replace(',', '.', $latitude);
                        $longitude = str_replace(',', '.', $longitude);

                        if (is_numeric($latitude) && is_numeric($longitude)) {
                            // Vérifier si les coordonnées existent déjà
                            $checkStmt->execute([$latitude, $longitude]);
                            $count = $checkStmt->fetchColumn();

                            if ($count == 0) {
                                // Exécution de la requête SQL pour l'insertion
                                $insertStmt->execute([$latitude, $longitude]);
                            }
                        }
                    }
                }
                // Fermeture du fichier CSV
                fclose($file);
            }
```

Une fois cela fait, j'ai programmé de sorte que selon l'état du piège, le point sur la carte soit rouge ou vert :

PHP :

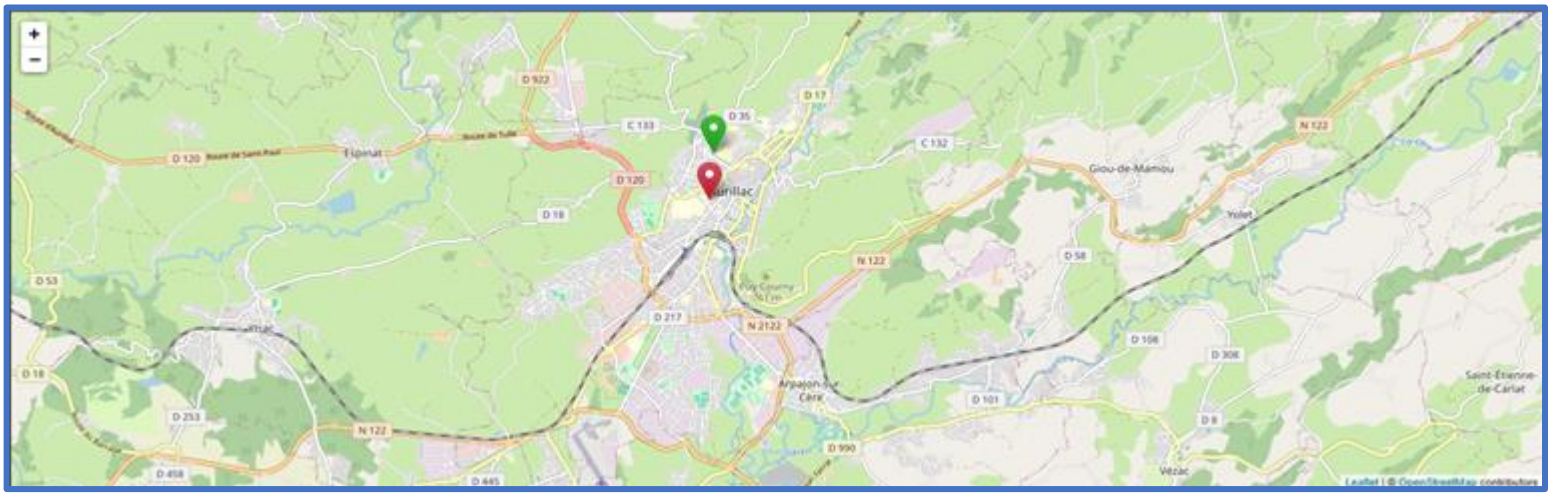
```
// Requête SQL pour récupérer les données avec état du piège
    $sql = "SELECT id, latitude, longitude, etat_piege FROM robotcat_table";
    $result = $bdd->query($sql);

    // Création d'un tableau pour stocker les coordonnées, l'état du piège et l'id
    $data = [];
    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {
        $data[] = [$row['id'], $row['latitude'], $row['longitude'],
$row['etat_piege']];
    }

    // Calculer le nombre total d'équipements
    $totalEquipment = count($data);

} catch (PDOException $e) {
    // Gestion des erreurs de connexion à la base de données ou d'exécution de
requête
    echo "Erreur : " . $e->getMessage();
    die();
}
?>
```

HTML :

```
  var redIcon = new L.Icon({
      iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-red.png',
      shadowUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    var greenIcon = new L.Icon({
      iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-green.png',
      shadowUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    // Parcourir le tableau de données et ajouter les marqueurs à la carte
    var data = <?php echo json_encode($data); ?>;
    for (var i = 0; i < data.length; i++) {
        var id = data[i][0];
        var lat = parseFloat(data[i][1]);
```

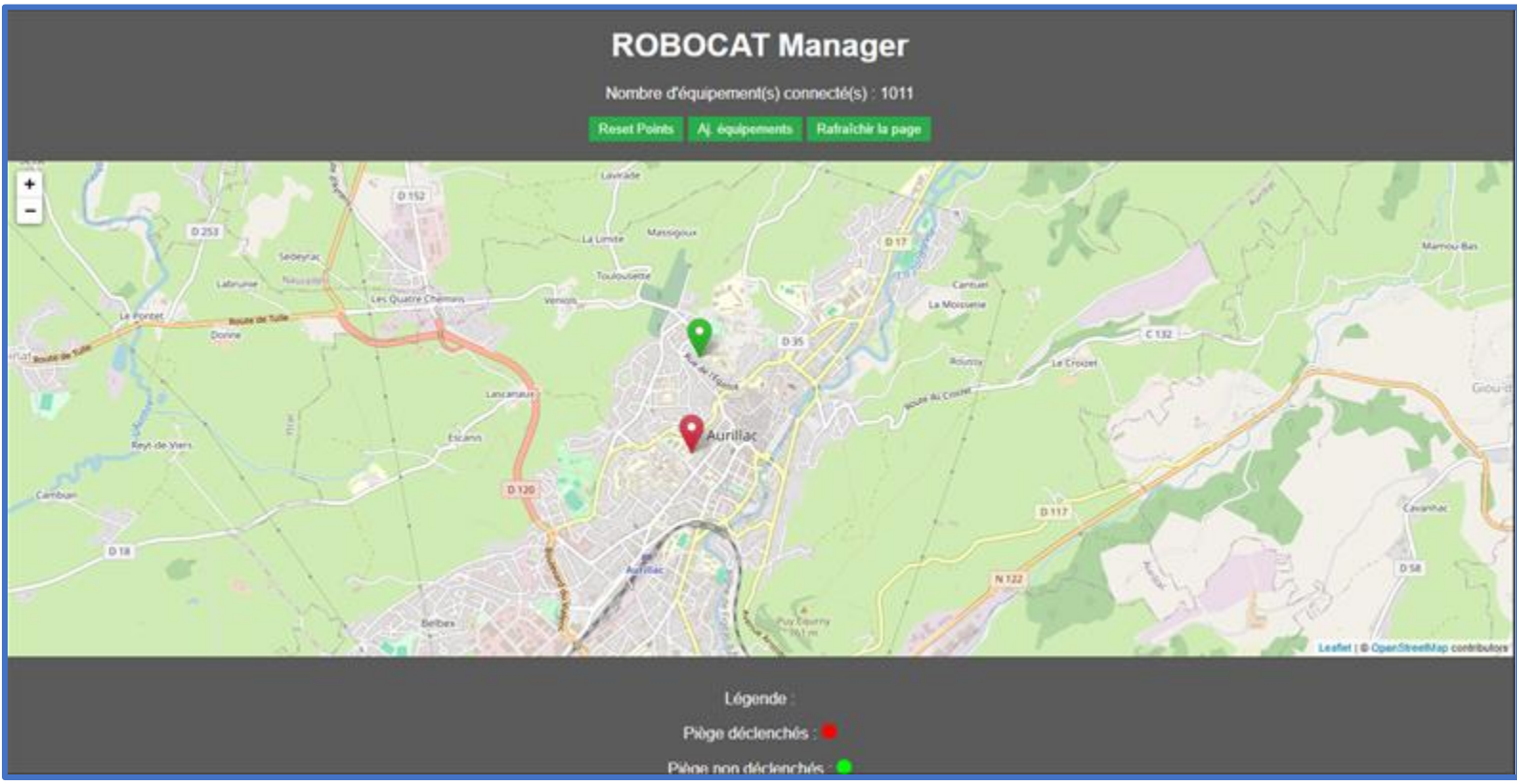
```
        var lon = parseFloat(data[i][2]);
        var etat_piege = parseInt(data[i][3], 10);
        var icon = (etat_piege === 1) ? redIcon : greenIcon; // Red icon for
etat_piege 1, green icon for 0
        var marker = L.marker([lat, lon], {icon: icon}).addTo(map);
        marker.bindPopup("ID: " + id + "<br>Latitude: " + lat + "<br>Longitude: "
+ lon);
        markers.push(marker);
    }
```

Points ajoutés manuellement dans la base de données :



Points fournis par l'INRAE (fichier CSV) :

Ensuite, j'ai travaillé sur le visuel du site :



J'ai ajouté un bouton "Ajout d'équipement" en PHP et JavaScript :

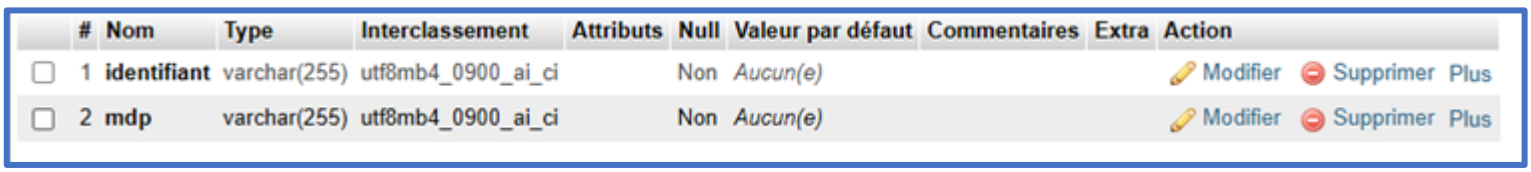
```
  <button id="add-equipment">Aj. équipements</button>
  <button onclick="location.reload()">Rafraîchir la page</button>
</div>

<div id="add-equipment-popup" style="display: none;">
  <form id="add-equipment-form">
    <label for="id">ID :</label><br>
    <input type="text" id="id" name="id"><br>
    <label for="latitude">Latitude :</label><br>
    <input type="text" id="latitude" name="latitude"><br>
    <label for="longitude">Longitude :</label><br>
    <input type="text" id="longitude" name="longitude"><br>
    <input type="submit" value="Ajouter">
  </form>
</div>

<script>
    // Afficher le popup lorsque vous cliquez sur le bouton "Aj. équipements"
    document.getElementById('add-equipment').addEventListener('click', function()
{
        document.getElementById('add-equipment-popup').style.display = 'block';
    });

    // Envoyer une requête POST lorsque le formulaire est soumis
    document.getElementById('add-equipment-form').addEventListener('submit',
function(e) {
        e.preventDefault();

        var id = document.getElementById('id').value;
        var latitude = document.getElementById('latitude').value;
        var longitude = document.getElementById('longitude').value;

        $.post('add_equipment.php', {id: id, latitude: latitude, longitude:
longitude, etat_piege: 1}, function(data) {
            alert('Équipement ajouté avec succès !');
            location.reload();
        });
    });
</script>

Avec la méthode Post les information sont envoyer a un script nommée ici
add_equipement.php

<?php

// Configuration de la base de données
$host = "localhost";
$dbname = "robo";
$username = "root";
$password = "";

try {
    // Connexion à la base de données
    $bdd = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
    // Récupérer les données du formulaire
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $id = $_POST['id'];
        $latitude = $_POST['latitude'];
        $longitude = $_POST['longitude'];
        $etat_piege = 1;

        // Insérer les données dans la base de données
        $insertStmt = $bdd->prepare("INSERT INTO robotcat_table (id, latitude,
longitude, etat_piege, etat_import) VALUES (?, ?, ?, ?, 0)");
        $insertStmt->execute([$id, $latitude, $longitude, $etat_piege]);

        echo "Équipement ajouté avec succès !";
    }

} catch (PDOException $e) {
    // Gestion des erreurs de connexion à la base de données ou d'exécution de
requête
    echo "Erreur : " . $e->getMessage();
    die();
}
?>
```

Un bouton pour rafraîchir la page :

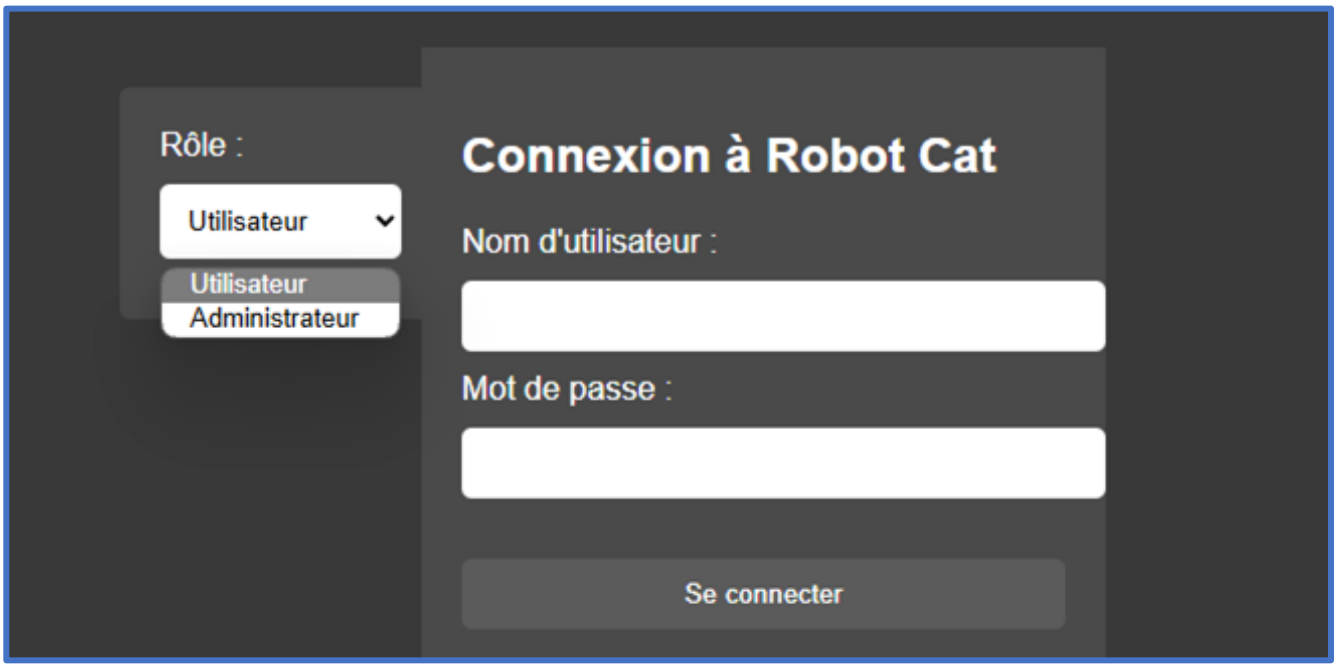
```
<button onclick="location.reload()">Rafraîchir la page</button>
</div>
```

Pour accéder à cette page, j'ai créé une page de connexion avec un identifiant et un mot de passe. Pour cela, j'ai créé deux nouvelles tables dans la base de données : `connexion_administrateur` et `connexion`. Pour différencier les utilisateurs classiques d'un administrateur, dans ces deux tables, on retrouve un champ identifiant et mdp pour mot de passe.



	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	identifiant	varchar(255)	utf8mb4_0900_ai_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	2	mdp	varchar(255)	utf8mb4_0900_ai_ci		Non	*Aucun(e)*			Modifier Supprimer Plus

Page de connexion :




```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Connexion à Robot Cat</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #333; /* Gris foncé */
            color: #fff; /* Blanc */
        }
        .login-container {
            width: 300px;
            padding: 20px;
            margin: 100px auto;
            background-color: #444; /* Gris */
            position: relative;
        }
        input[type="text"], input[type="password"], select {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: none;
            border-radius: 5px;
        }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            margin: 20px 0;
            border: none;
            border-radius: 5px;
            background-color: #555; /* Gris plus clair */
            color: #fff; /* Blanc */
            cursor: pointer;
        }
```

```
        a {
            color: #aaa; /* Gris clair */
            text-decoration: none;
        }
        a:hover {
            text-decoration: underline;
        }
        .role-container {
            position: absolute;
            top: 20px;
            left: -150px;
            width: 120px;
            background-color: #444; /* Gris */
            padding: 20px;
            border-radius: 5px;
        }
        .role-container select {
            width: 100%;
            padding: 10px;
            border: none;
            border-radius: 5px;
        }
        .error {
            color: #ff6666; /* Rouge */
            margin: 10px 0;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <div class="role-container">
            <label for="role">Rôle :</label>
            <select id="role" name="role">
                <option value="utilisateur">Utilisateur</option>
                <option value="administrateur">Administrateur</option>
            </select>
        </div>
        <h2>Connexion à Robot Cat</h2>
        <form id="loginForm" action="traitement_connexion.php" method="post">
            <!-- Message d'erreur -->
            <?php if (isset($_GET['error'])): ?>
                <div class="error"><?php echo
htmlspecialchars($_GET['error']); ?></div>
            <?php endif; ?>
            <!-- Partie connexion -->
            <label for="username">Nom d'utilisateur :</label>
            <input type="text" id="username" name="username" required>

            <!-- Partie mot de passe -->
            <label for="password">Mot de passe :</label>
            <input type="password" id="password" name="password" required>

            <input type="submit" value="Se connecter">
        </form>
        <!-- Hyperliens -->
        <a href="cree_un_compte.php">Créer un compte</a> |
        <a href="mot_de_passe_oublie.php">Mot de passe oublié?</a>
    </div>
</body>
```

```
</html>
```

Une fois les informations entrées, l'identifiant et le mot de passe sont envoyés au script traitement_connexion.php qui compare les informations entrées avec celles présentes dans la base de données.

```
<?php
// Démarrage de la session
session_start();

// Connexion à la base de données
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "robo";

// Création de la connexion
$conn = new mysqli($servername, $username, $password, $dbname);

// Vérification de la connexion
if ($conn->connect_error) {
    die("Échec de la connexion: " . $conn->connect_error);
}

// Vérifiez si le formulaire a été soumis
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Récupération des valeurs du formulaire
    $login = isset($_POST['username']) ? $_POST['username'] : '';
    $mdp = isset($_POST['password']) ? $_POST['password'] : '';
    $role = isset($_POST['role']) ? $_POST['role'] : '';

    // Diagnostic: Affichage des valeurs récupérées
    error_log("Login: $login, MDP: $mdp, Role: $role");

    // Vérification des valeurs récupérées
    if ($login !== '' && $mdp !== '' && $role !== '') {
        // Préparation de la requête SQL en fonction du rôle
        if ($role === 'administrateur') {
            $sql = "SELECT * FROM connexion_administrateur WHERE identifiant = ?
AND mdp = ?";
        } else {
            $sql = "SELECT * FROM connexion WHERE identifiant = ? AND mdp = ?";
        }

        $stmt = $conn->prepare($sql);

        // Vérification si la préparation a réussi
        if ($stmt === false) {
            die("Erreur de préparation de la requête: " . $conn->error);
        }

        // Liaison des paramètres
        $stmt->bind_param("ss", $login, $mdp);

        // Exécution de la requête
        $stmt->execute();
```

```
        // Stockage du résultat
        $stmt->store_result();

        // Vérification si un utilisateur correspond
        if ($stmt->num_rows == 1) {
            // L'utilisateur existe dans la base de données
            $_SESSION['user_login'] = $login;
            if ($role === 'administrateur') {
                header("Location: page_principale2.php"); // Redirection vers la
page administrateur
            } else {
                header("Location: page_principale1.php"); // Redirection vers la
page utilisateur
            }
            exit();
        } else {
            // L'utilisateur n'existe pas
            $error = "Identifiant ou mot de passe incorrect.";
            header('Location: connexion.php?error=' . urlencode($error));
            exit();
        }

        // Fermeture de la requête
        $stmt->close();
    } else {
        // Diagnostic: Affichage des valeurs vides
        error_log("Valeurs manquantes - Login: $login, MDP: $mdp, Role: $role");

        // En cas de valeurs manquantes
        $error = "Veuillez remplir tous les champs.";
        header('Location: connexion.php?error=' . urlencode($error));
        exit();
    }
}

// Fermeture de la connexion
$conn->close();
?>
```

Si les identifiants correspondent, elle affiche la page contenant la carte et les points.

Difficultés rencontrées :

Pendant ma partie, la principale difficulté rencontrée a été d'intégrer et de manipuler la carte interactive, ainsi que de faire afficher et correspondre les différentes informations entre la base de données et les points sur la carte.

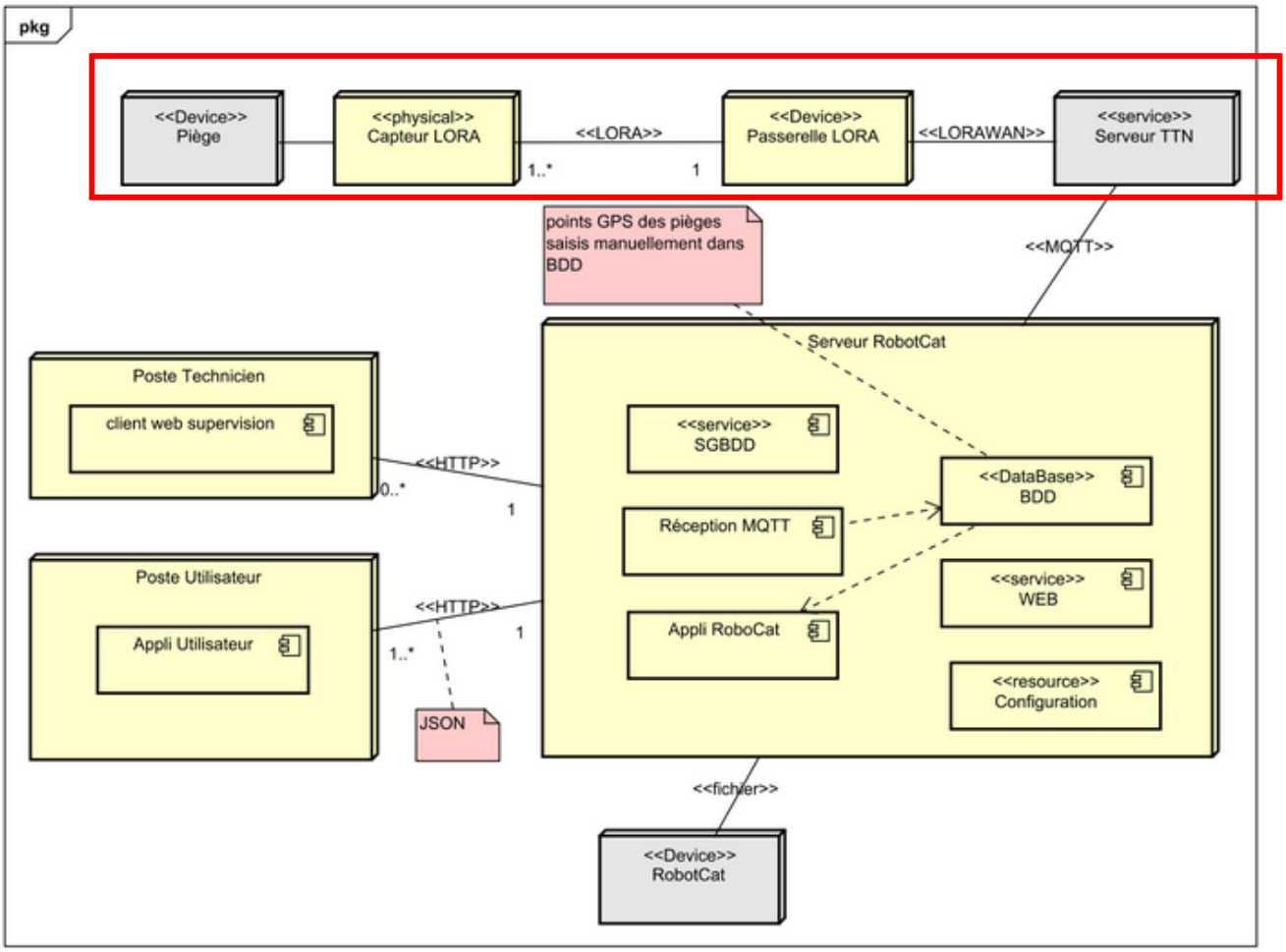
Conclusion :

Grâce à ce projet, j'ai pu apprendre à intégrer sur un site différentes fonctionnalités et à comprendre leur fonctionnement. J'ai également appris à mieux travailler en équipe sur des projets concrets.

Partie individuelle (PHILIPPE Mehdi)

Introduction :

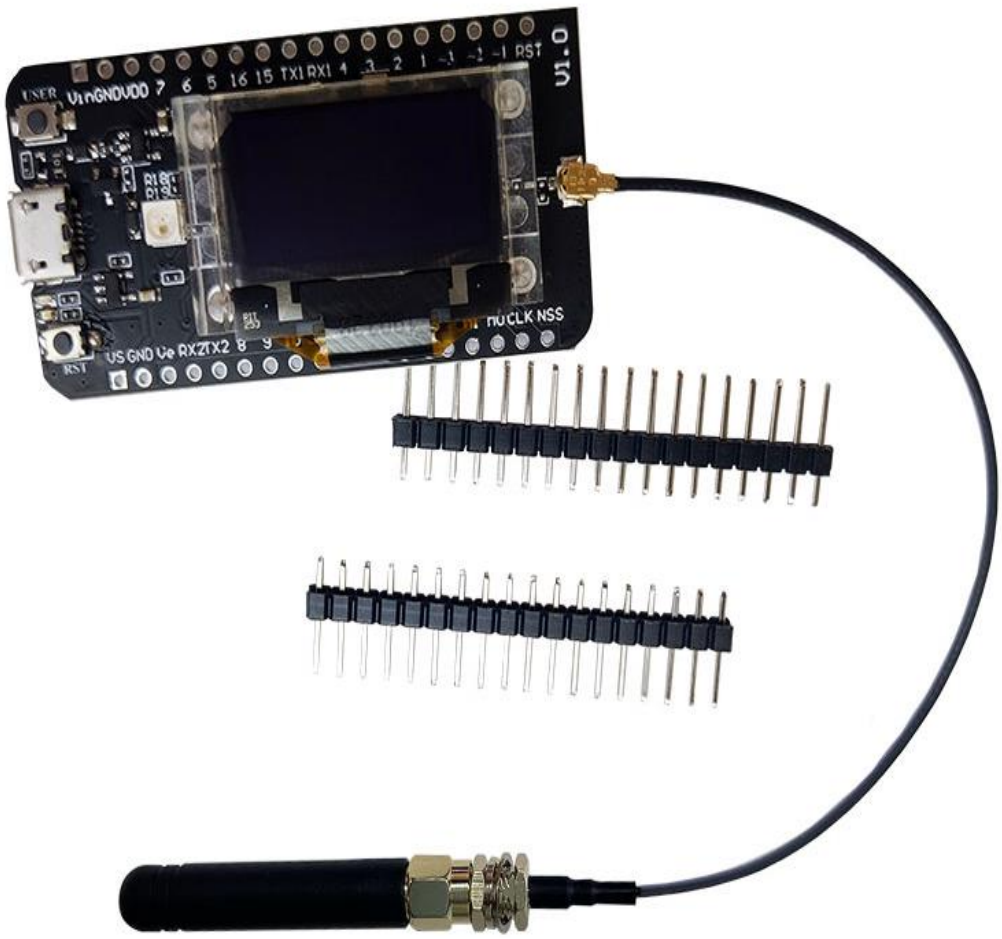
Dans ma partie, j'ai dû gérer la liaison entre les différents capteurs et la base de données, ainsi que la collecte des informations nécessaires (identifiant, état). Pour cela, on nous a fourni une passerelle Dragino LoRa et j'ai donc dû trouver des capteurs adaptés pour LoRa, afin d'acheminer les informations dans la base de données via un serveur TTN.



Analyse / Matériels utilisés :

➔ Le capteur :

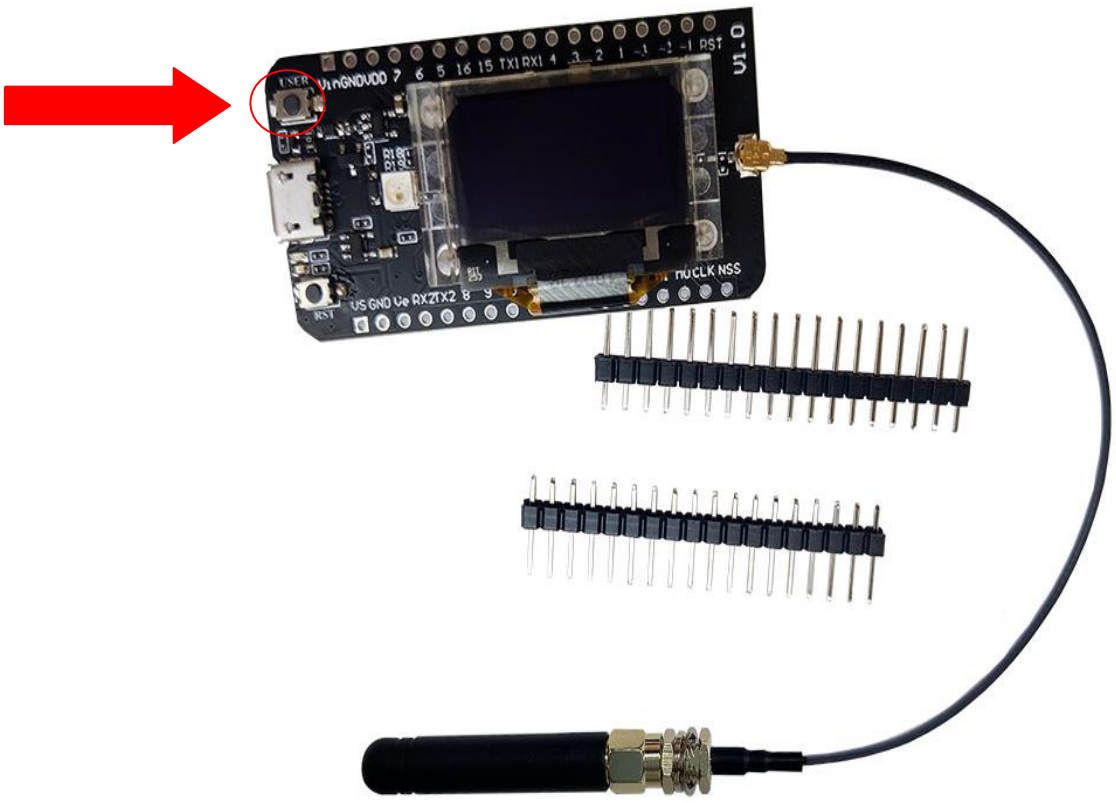
Pour ce faire, nous avons trouvé une carte, la CubeCell GPS-6502 (HTCC-AB02S), à laquelle nous avons branché un capteur ainsi qu'une antenne LoRa qui fera le lien entre le capteur et la passerelle.



Pour ce qui est du capteur, nous n'avons pas besoin de quelque chose de très sophistiqué. Pour cela, nous avons simplement utilisé un capteur de vibration Piezo en tant que solution provisoire, qui sera placé directement sur le piège afin de détecter les changements d'état du piège par les vibrations émises lors de son déclenchement.

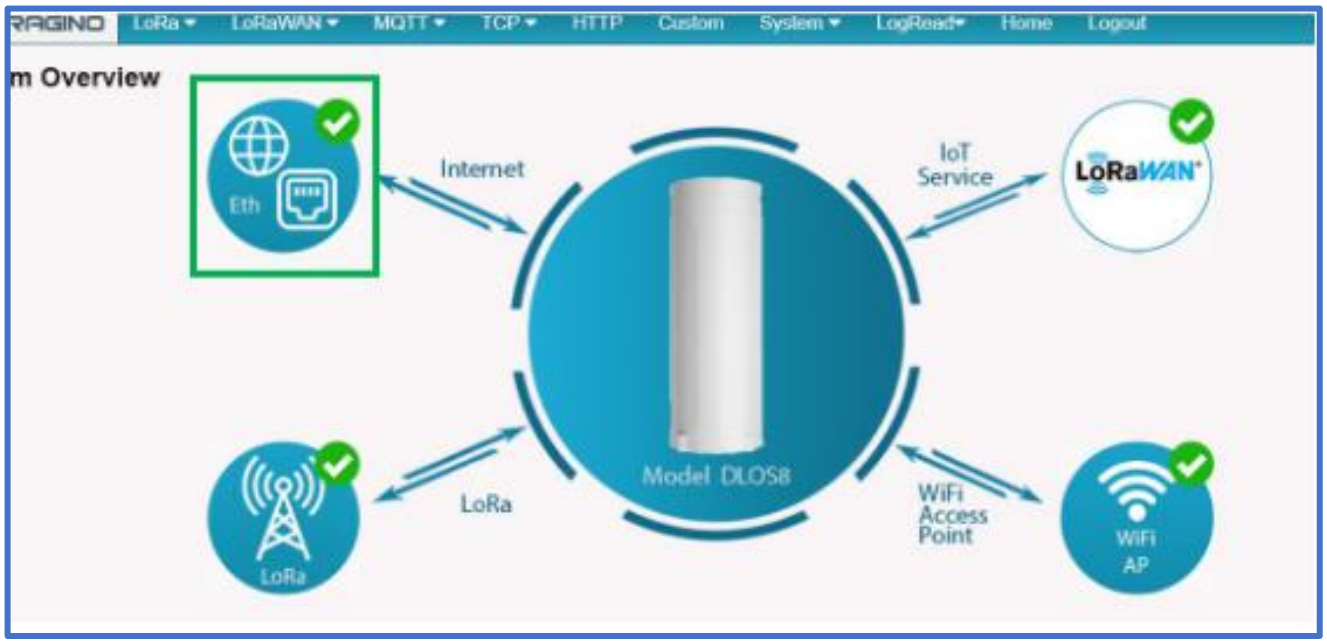
Seulement, nous avons changé de solution pour finalement choisir un bouton utilisateur présent sur la carte afin de simuler le changement d'état du piège.

Bouton **USER** :



➔ La passerelle LoRaWan

Donc, comme dit précédemment, nous avons utilisé une passerelle Dragino DL0S8. Et via l'interface de gestion, nous avons pu régler et vérifier le bon fonctionnement de la passerelle en réseau grâce à un accès Internet via un câble RJ45.



- Principales caractéristiques :

Technologie LoRaWAN : La DL0S8 utilise la technologie LoRaWAN (Long Range Wide Area Network), qui permet une communication longue portée et une consommation d'énergie faible. Cette technologie est idéale pour les applications IoT (Internet des objets) nécessitant une couverture étendue et une autonomie prolongée des capteurs.

Canaux : La DL0S8 est capable de gérer plusieurs canaux de communication, ce qui permet de recevoir et de transmettre des données de nombreux capteurs simultanément.

Compatibilité : Elle est compatible avec divers réseaux de serveurs LoRaWAN publics et privés, permettant une grande flexibilité d'intégration dans différents systèmes IoT.

- Les différentes fonctionnalités :

Gestion des données : La passerelle collecte les données des capteurs LoRaWAN et les envoie aux serveurs de données pour traitement et analyse. Cela permet une surveillance en temps réel et une gestion efficace des dispositifs IoT.

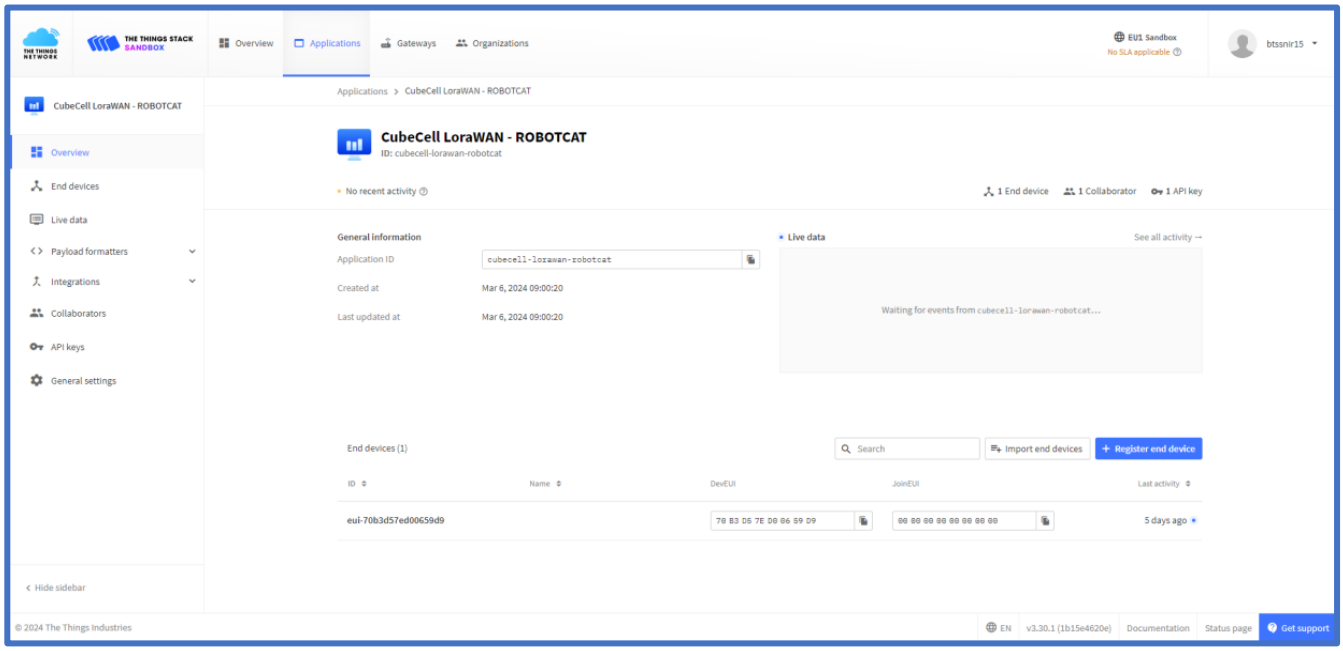
Sécurité : Elle offre des fonctionnalités de sécurité pour assurer que les données transmises sont protégées contre les accès non autorisés et les interférences.

Configuration et gestion : La passerelle peut être configurée et gérée à distance via une interface utilisateur, comme vue précédemment, facilitant l'installation et la maintenance.

En résumé, la passerelle Dragino DL0S8 est un dispositif essentiel pour les réseaux IoT, offrant une connectivité fiable, une gestion efficace des données et des capacités de communication étendues grâce à la technologie LoRaWAN. Elle est polyvalente et s'adapte à une multitude de scénarios d'utilisation dans différents secteurs. Ce qui dans notre cas se trouve être un véritable atout.

➔ Serveur TTN

Une fois cette plateforme mise en place, elle nous sert de lien avec le serveur TTN qui s'occupera de fournir la base de données.



Ici nous sommes sur l'onglet "Applications" sur The Things Network (TTN) qui est une section clé de la console TTN, permettant aux utilisateurs de gérer et d'interagir avec leurs applications IoT.

Création d'Applications :

- Les utilisateurs peuvent créer de nouvelles applications en fournissant un nom et des informations spécifiques. Et chaque application est identifiée par un identifiant unique (Application ID).

Gestion des Dispositifs :

- Les utilisateurs peuvent ajouter des dispositifs (capteurs ou actionneurs) à une application. Chaque dispositif est identifié par un Device ID.
- Il est possible de configurer les paramètres de chaque dispositif, tels que l'AppEUI (Application EUI) et le AppKey (Application Key), nécessaires pour l'authentification et la sécurité

Visualisation des Données :

- Les utilisateurs peuvent surveiller les données reçues des dispositifs en temps réel.
- Les données peuvent être affichées sous forme de graphiques ou de flux de données brutes.

Configuration des Fonctions :

- Les utilisateurs peuvent définir des fonctions spécifiques pour traiter les données des dispositifs, comme le décodage des payloads (données envoyées par les capteurs) ou la conversion des formats de données.

Sécurité :

- En permettant la configuration de clés de sécurité et d'authentification pour chaque dispositif, TTN assure la sécurité des données transmises.

- Conclusion personnelle :

En conclusion j'ai étudié la technologie LoRa WAN, j'ai mis en place un système reliant un capteur à une carte ESP32 et envoyant des données via Lora à une base de données auto-incrémentée, via un serveur TTN programmé pour extraire et envoyé un id et un état, ceux des pièges mis en place par le robot. Ainsi j'ai acquis de l'expérience dans le travail d'équipe ainsi que dans la gestion de mon temps de travail. Aussi, j'ai pu découvrir les difficultés de réaliser un projet de A à Z et cela m'a permis accroître mes connaissances et mes compétences.

Conclusion générale :

En conclusion, nous avons apprécié le projet qui nous été attribué. Mais ce dernier n'est pas fini, il faut seulement trouver une solution pour récupérer les valeurs reçues sur TTN pour pouvoir les envoyer sur la base de données qui est stocké directement sur la Raspberry Pi avec PHPMyAdmin.

De plus, cela nous a permis d'apprendre de nouvelles technologies comme le LoRa, le service The Things Network, la création de page Web avec une carte intéractive.

La phrase d'échange et d'avancement entre tout les membres de l'équipe nous a appris à travailler de manière optimale, de gérer notre temps et de mettre en œuvre nos compétences acquises durant notre formation.

Annexe Partie étudiant Mathéo :

- Code ESP32 LoRaWAN (sur la CubeCell HTCC-AB02S)

```
#include "LoRaWan_APP.h"
#include "Arduino.h"
/*
 * set LoraWan_RGB to Active,the RGB active in loraWan
 * RGB red means sending;
 * RGB purple means joined done;
 * RGB blue means RxWindow1;
 * RGB yellow means RxWindow2;
 * RGB green means received done;
 */
/* OTAA para*/
uint8_t devEui[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x06, 0x59, 0xD9 }; // issue de
TTN partie création de End Devices
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appKey[] = { 0xCE, 0x8E, 0x0B, 0x0E, 0x20, 0xDF, 0x2A, 0xF3, 0x5B, 0x19,
0x7B, 0xE9, 0xE8, 0xEF, 0xA6, 0x2F }; // issue de TTN partie création de End
Devices

/* ABP para*/
uint8_t nwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d, 0x11,
0x18, 0x1e, 0x1e, 0xc7, 0xda,0x85 };
uint8_t appSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf, 0x55, 0xee,
0x4a, 0x77, 0x8d, 0x16, 0xef,0x67 };
uint32_t devAddr =  ( uint32_t )0x007e6ae1;

/*LoraWan channelsmask*/
uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };

/*LoraWan region, select in arduino IDE tools*/
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/*LoraWan Class, Class A and Class C are supported*/
DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 15000;

/*OTAA or ABP*/
bool overTheAirActivation = LORAWAN_NETMODE;

/*ADR enable*/
bool loraWanAdr = LORAWAN_ADR;
```

```
/* set LORAWAN_Net_Reserve ON, the node could save the network info to flash, when
node reset not need to join again */
bool keepNet = LORAWAN_NET_RESERVE;

/* Indicates if the node is sending confirmed or unconfirmed messages */
bool isTxConfirmed = LORAWAN_UPLINKMODE;

/* Application port */
uint8_t appPort = 2;
/*!
* Number of trials to transmit the frame, if the LoRaMAC layer did not
* receive an acknowledgment. The MAC performs a datarate adaptation,
* according to the LoRaWAN Specification V1.0.2, chapter 18.4, according
* to the following table:
*
* Transmission nb | Data Rate
* ----------------|-----------
* 1 (first)       | DR
* 2               | DR
* 3               | max(DR-1,0)
* 4               | max(DR-1,0)
* 5               | max(DR-2,0)
* 6               | max(DR-2,0)
* 7               | max(DR-3,0)
* 8               | max(DR-3,0)
*
* Note, that if NbTrials is set to 1 or 2, the MAC will not decrease
* the datarate, in case the LoRaMAC layer did not receive an acknowledgment
*/
uint8_t confirmedNbTrials = 4;

/* Prepares the payload of the frame */
static void prepareTxFrame( uint8_t port, int E )
{
  /*appData size is LORAWAN_APP_DATA_MAX_SIZE which is defined in
"commissioning.h".
  *appDataSize max value is LORAWAN_APP_DATA_MAX_SIZE.
  *if enabled AT, don't modify LORAWAN_APP_DATA_MAX_SIZE, it may cause system
hanging or failure.
  *if disabled AT, LORAWAN_APP_DATA_MAX_SIZE can be modified, the max value is
reference to lorawan region and SF.
  *for example, if use REGION_CN470,
  *the max value for different DR can be found in MaxPayloadOfDatarateCN470 refer
to DataratesCN470 and BandwidthsCN470 in "RegionCN470.h".
  */
    appDataSize = 4;

    if(E==HIGH){
    appData[0] = 0x01;
```

```
    appData[1] = 0x01;
    appData[2] = 0x01;
    appData[3] = 0x01;
    }
    else
    {
    appData[0] = 0x00;
    appData[1] = 0x00;
    appData[2] = 0x00;
    appData[3] = 0x00;

    }
}

#define entree 16
#define sortie 7

int etat=0;
int Eavant=0;
int frontmontant=0;
int Envoyer=0;
int EtatPiege=0; // piège non déclenché

void setup() {
  Serial.begin(115200);
  pinMode(entree, INPUT);
  pinMode(sortie, OUTPUT);


#if(AT_SUPPORT)
  enableAt();
#endif
  LoRaWAN.displayMcuInit();
  deviceState = DEVICE_STATE_INIT;

  LoRaWAN.ifskipjoin();
}

void loop()
{

int E=digitalRead(entree);

  if (E == HIGH && Eavant == LOW) {
    Serial.println("Front montant sur E");
    frontmontant=1;
    Envoyer=1;
    EtatPiege=1; // piège déclenché
  }
```

```
  if( E== LOW && Eavant==HIGH) {
    Serial.println("Front descendant sur E");
    frontmontant=0;
  }
  Eavant=E;

// faire l'envoie LORA que si front montant
if(Envoyer==1)
{

  switch( deviceState )
  {
    case DEVICE_STATE_INIT:
    {
#if(LORAWAN_DEVEUI_AUTO)
      LoRaWAN.generateDeveuiByChipID();
#endif
#if(AT_SUPPORT)
      getDevParam();
#endif
      printDevParam();
      LoRaWAN.init(loraWanClass,loraWanRegion);
      deviceState = DEVICE_STATE_JOIN;
      break;
    }
    case DEVICE_STATE_JOIN:
    {
      LoRaWAN.displayJoining();
      LoRaWAN.join();
      break;
    }
    case DEVICE_STATE_SEND:
    {
      LoRaWAN.displaySending();
      prepareTxFrame( appPort, EtatPiege );
      LoRaWAN.send();
      deviceState = DEVICE_STATE_CYCLE;

      Envoyer=0;
      break;
    }
    case DEVICE_STATE_CYCLE:
    {
      // Schedule next packet transmission
      txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
      LoRaWAN.cycle(txDutyCycleTime);
      deviceState = DEVICE_STATE_SLEEP;
      break;
    }
```

```
    case DEVICE_STATE_SLEEP:
    {
      LoRaWAN.displayAck();
      LoRaWAN.sleep();

      break;
    }
    default:
    {
      deviceState = DEVICE_STATE_INIT;
      break;
    }
  }

}

//attendre delta t
delay(100); // Un 1/10 seconde de pause

}
```

Annexe 1 : Installation du service MariaDB et PHPMyAdmin sur une Raspberry Pi (3/4)

1) Mise à jour du système :

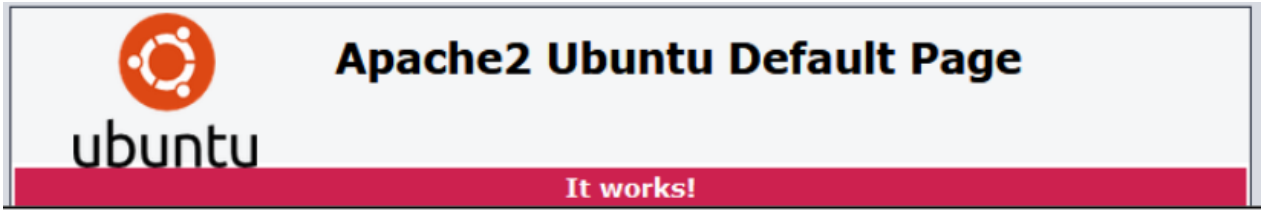
➔ Pour la recherche et le téléchargement des mises à jour, dans la console taper : `sudo apt-get update`

➔ Puis pour effectuer les mises à jour, taper : `sudo apt-get upgrade`

2) Installation de Apache2 :

Apache est un serveur de page web, on l'installe en tapant dans la console, la commande : `sudo apt install apache2`

Pour tester Apache, ouvrir un navigateur et taper dans la barre d'adresse, l'adresse de votre serveur. Si votre serveur Apache fonctionne correctement. Votre navigateur doit afficher la page ci-dessous.



Les pages web que l'on veut que le serveur mette à disposition doivent être déposées dans le répertoire var/www/html, il faut donc modifier les droits de ce répertoire afin que l'on puisse y déposer des fichiers en tapant la commande : `sudo chmod 777 /var/www/html`

3) Installation de PHP :

PHP est un langage de script qui permet de créer des pages web dynamiques interprétées côtés serveur.

On installe un interpréteur PHP en tapant la commande : `sudo apt install php libapache2-mod-php`

Pour tester si PHP fonctionne, on crée le fichier **index.php** qui contient le script **<?php phpinfo(); ?>** et on place ce fichier dans le répertoire /var/www/html en tapant la commande : `echo "<?php phpinfo(); ?>" > /var/www/html/index.php`

Puis dans la barre d'adresse d'un navigateur, **on tape l'adresse de la Raspberry suivie de /index.php**. Votre navigateur doit afficher la page ci-

dessous.

4) Installation de MySql (MariaDB) :

MariaDb est un fork de MySQL. C'est un gestionnaire de base de données qui s'installe en tapant la commande : `sudo apt install mariadb-server php-mysql`

Lors de l'installation, il y a création d'un utilisateur nommé « root » pour l'utilisation du gestionnaire de bases de données. Attention ce n'est pas le même utilisateur « root » que celui du système.

Cet utilisateur n'a pas de mot de passe. Il faut lui en donner un. Dans la console, taper la commande :

`sudo mysql`

Puis effectuer les requêtes afin d'attribuer un mot de passe à l'utilisateur « root ».

```
    USE mysql;
    update user set password = password('mdpDeVotrechoix') where user
= 'root' and host='localhost';
    FLUSH PRIVILEGES;
    QUIT;
```

5) Installation de PHPMyAdmin :

PHPMyAdmin est une interface graphique qui permet de gérer des bases de données plus facilement qu'en ligne de commande.

5-1) Création d'un utilisateur pour PhpMyAdmin :

MySQL autorise l'accès aux informations d'identification de l'utilisateur « root » que si on est super utilisateur (sudo).
PHPMyAdmin ne pourra donc pas utiliser l'utilisateur « root » pour se connecter.

La solution la plus simple est de créer un nouvel utilisateur à qui on octroie les privilèges requis. Dans la console, taper la commande :

`sudo mysql`

Puis taper les requêtes :

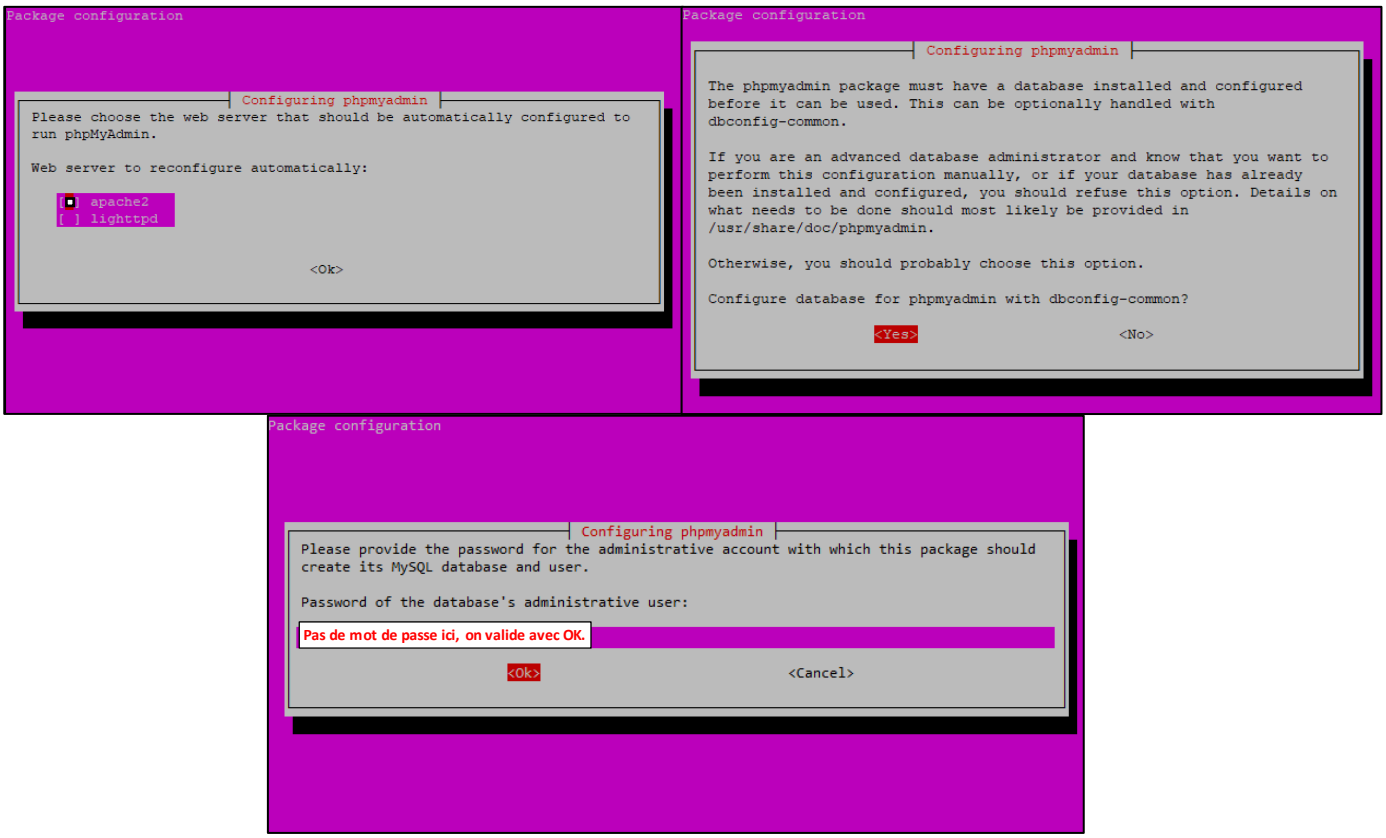
```
    CREATE USER 'piP'@'localhost' IDENTIFIED BY 'raspberry';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'rootpma'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
QUIT ;
```

Remarque : Si on veut autoriser la connexion à distance à PhpMyAdmin**, il faut remplacer le mot localhost par le caractère % dans la requête ci-dessus.**

6-2) Installation de PhpMyAdmin :

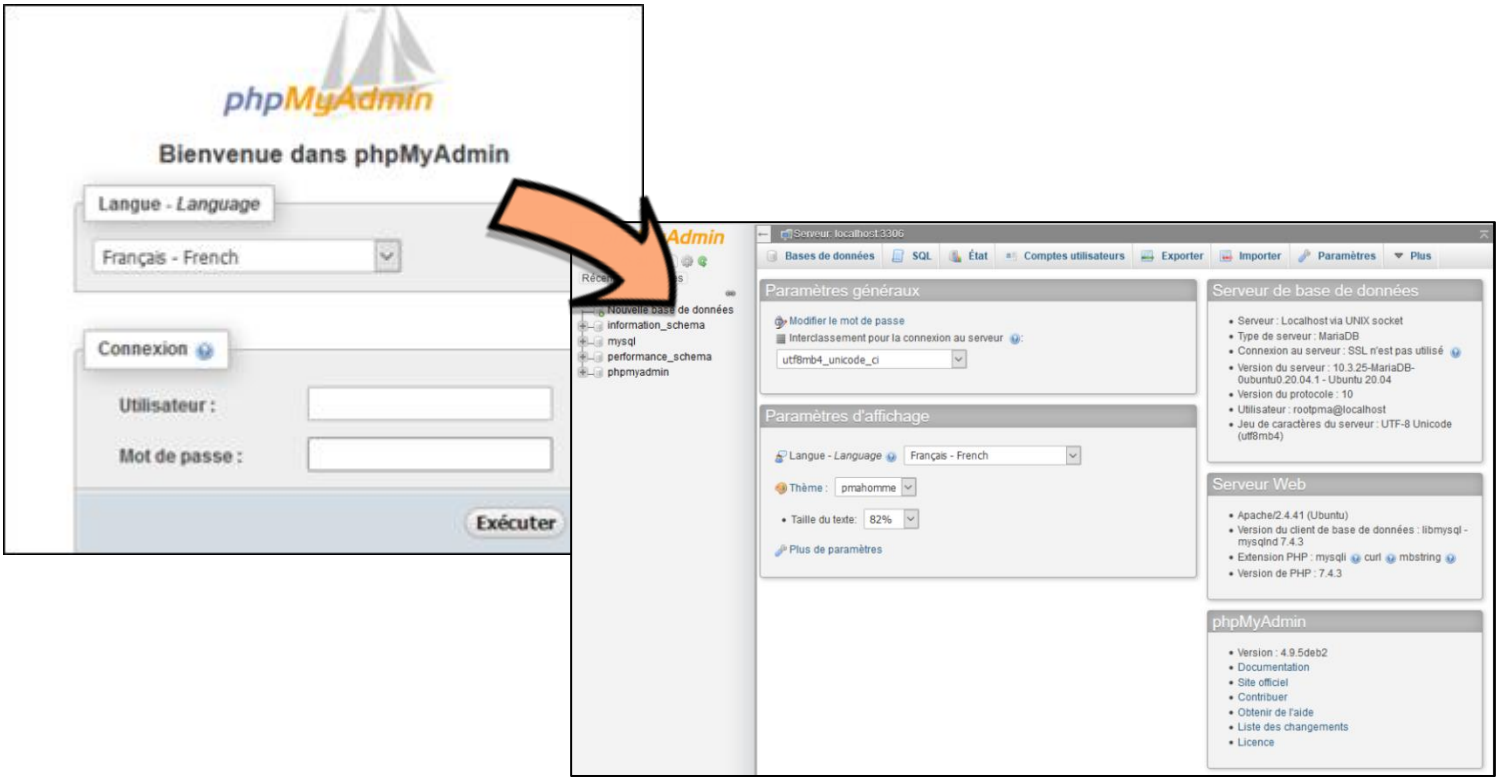
Pour installer PHPMyAdmin, dans la console il faut taper : `sudo apt-get install phpmyadmin`



5-3) Test de PHPMyAdmin :

On se connecte à la page d'accueil de PhpMyAdmin en tapant l'adresse suivante dans la barre d'adresse du navigateur internet :

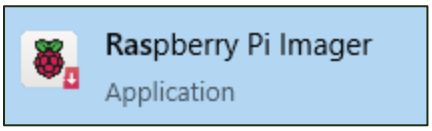
- **localhost/phpmyadmin** (Mode local)
- **adresseServeur/phpmyadmin** (Mode ordinateur distant)

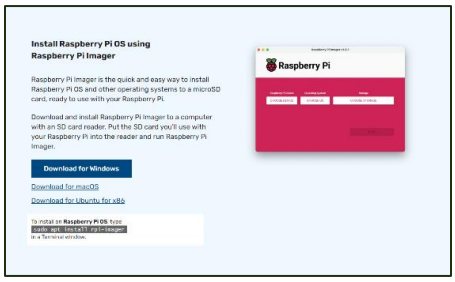


PHPMyAdmin est installé !

- Annexe 2 : Mise en place de la Raspberry Pi (Installation de l'OS)

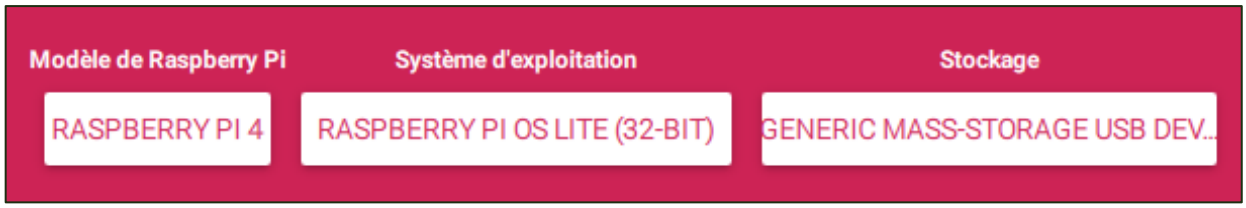
Pour pouvoir installer une image Raspberry Pi OS, il est recommandé d'utiliser le logiciel propriétaire qui est le suivant : Raspberry Pi Imager.





Il peut être téléchargé sur le site Web officiel : https://www.raspberrypi.com/software/

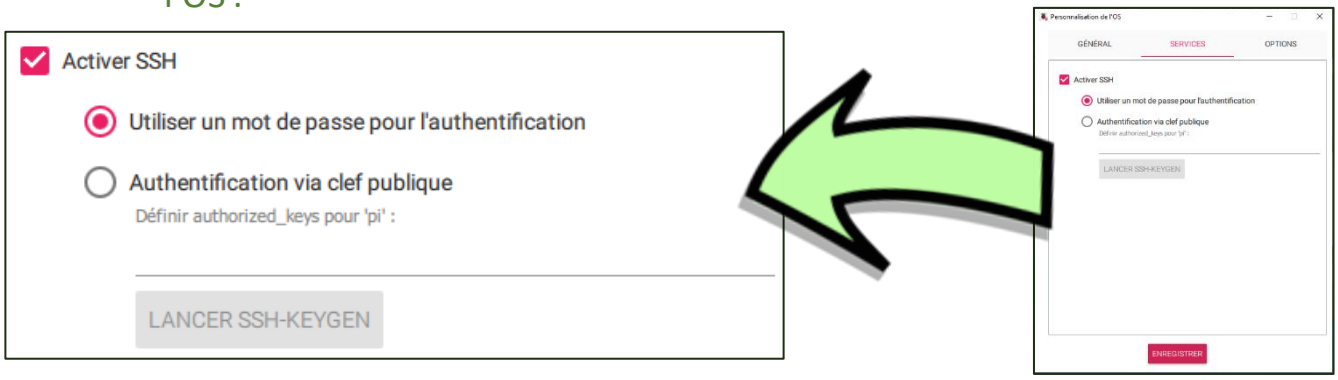
➔ En lançant le logiciel on se retrouve à choisir différentes options :



➔ Le modèle de Raspberry est important et obligatoire à choisir avant de commencer l'installation.

➔ Le système d'exploitation à deux versions différentes… une version « complète » c'est-à-dire avec une interface graphique et une version « Lite » sans interface graphique.

➔ Avant de commencer l'installation passons dans les réglages initiaux de l'OS :



! -- Le service SSH doit être activé pour que la Raspberry puisse être utilisé sans écran et qu'elle soit contrôlable depuis un autre PC avec **Putty** ou **WinSCP**… on peut ensuite lancer l'installation.

Annexe Partie étudiant Maxime :

- Code en PHP, HTML & JAVASCRIPT de la page principale avec la carte interactive :

```
<?php

// Configuration de la base de données
$host = "localhost";
$dbname = "robo";
$username = "root";
$password = "";

try {
    // Connexion à la base de données
    $bdd = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Vérifier si le fichier CSV a déjà été traité
    $checkImportStmt = $bdd->query("SELECT COUNT(*) FROM robotcat_table WHERE
etat_import = 1");
    $imported = $checkImportStmt->fetchColumn();

    if ($imported == 0) {
        // Lire le fichier CSV et insérer les nouveaux points GPS uniquement si ce
n'est pas encore fait
        $filePath = 'C:\wamp64\www\html\page_principale\PointGPS_Recu.CSV';
        if (file_exists($filePath) && filesize($filePath) > 0) {
            $file = fopen($filePath, 'r');
            if ($file) {
                // Lecture de la première ligne (en-tête)
                $header = fgetcsv($file, 0, ';');
                if ($header !== FALSE) {
                    // Déterminer les indices des colonnes latitude et longitude
                    $latitudeIndex = 8;  // Colonne I
                    $longitudeIndex = 9; // Colonne J

                    // Préparation de la requête SQL pour vérifier l'existence des
points
                    $checkStmt = $bdd->prepare("SELECT COUNT(*) FROM
robotcat_table WHERE latitude = ? AND longitude = ?");
                    // Préparation de la requête SQL pour l'insertion
                    $insertStmt = $bdd->prepare("INSERT INTO robotcat_table
(latitude, longitude, etat_piege, etat_import) VALUES (?, ?, 0, 1)");

                    // Lecture du fichier CSV ligne par ligne
                    while (($row = fgetcsv($file, 0, ';')) !== FALSE) {
                        // Nettoyer et vérifier les valeurs de latitude et de
longitude
                        $latitude = isset($row[$latitudeIndex]) ?
trim($row[$latitudeIndex]) : '';
                        $longitude = isset($row[$longitudeIndex]) ?
trim($row[$longitudeIndex]) : '';

                        // Remplacer les virgules par des points si nécessaire
```

```
                        $latitude = str_replace(',', '.', $latitude);
                        $longitude = str_replace(',', '.', $longitude);

                        if (is_numeric($latitude) && is_numeric($longitude)) {
                            // Vérifier si les coordonnées existent déjà
                            $checkStmt->execute([$latitude, $longitude]);
                            $count = $checkStmt->fetchColumn();

                            if ($count == 0) {
                                // Exécution de la requête SQL pour l'insertion
                                $insertStmt->execute([$latitude, $longitude]);
                            }
                        }
                    }
                }
                // Fermeture du fichier CSV
                fclose($file);
            }
        }
    }

    // Requête SQL pour récupérer les données avec état du piège
    $sql = "SELECT id, latitude, longitude, etat_piege FROM robotcat_table";
    $result = $bdd->query($sql);

    // Création d'un tableau pour stocker les coordonnées, l'état du piège et l'id
    $data = [];
    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {
        $data[] = [$row['id'], $row['latitude'], $row['longitude'],
$row['etat_piege']];
    }

    // Calculer le nombre total d'équipements
    $totalEquipment = count($data);

} catch (PDOException $e) {
    // Gestion des erreurs de connexion à la base de données ou d'exécution de
requête
    echo "Erreur : " . $e->getMessage();
    die();
}
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ROBOCAT Manager</title>
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"/>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            color: #ccc;
            background-color: #333;
        }
```

```
        #header {
            background-color: #555;
            color: white;
            padding: 10px;
            text-align: center;
        }

        #osm-map {
            height: 500px;
            width: 100%;
        }

        #footer {
            background-color: #555;
            color: white;
            padding: 20px;
            text-align: center;
            clear: both;
        }

        button {
            background-color: #28a745;
            color: white;
            padding: 5px 10px;
            margin: 10px 0;
            border: none;
            cursor: pointer;
        }

        button:hover {
            background-color: #218838;
        }

        h1, p {
            margin: 0;
        }

        p {
            margin-top: 10px;
        }

        .red-dot {
            height: 15px;
            width: 15px;
            background-color: #f00;
            border-radius: 50%;
            display: inline-block;
        }

        .green-dot {
            height: 15px;
            width: 15px;
            background-color: #0f0;
            border-radius: 50%;
            display: inline-block;
        }
    </style>
</head>
```

```
<body>

<div id="header">
  <h1>ROBOCAT Manager</h1>
  <p>Nombre d'équipement(s) connecté(s) : <?php echo $totalEquipment; ?></p>
  <button id="reset-points">Reset Points</button>
  <button id="add-equipment">Aj. équipements</button>
  <button onclick="location.reload()">Rafraîchir la page</button>
</div>

<div id="add-equipment-popup" style="display: none;">
  <form id="add-equipment-form">
    <label for="id">ID :</label><br>
    <input type="text" id="id" name="id"><br>
    <label for="latitude">Latitude :</label><br>
    <input type="text" id="latitude" name="latitude"><br>
    <label for="longitude">Longitude :</label><br>
    <input type="text" id="longitude" name="longitude"><br>
    <input type="submit" value="Ajouter">
  </form>
</div>

<script>
    // Afficher le popup lorsque vous cliquez sur le bouton "Aj. équipements"
    document.getElementById('add-equipment').addEventListener('click', function()
{
        document.getElementById('add-equipment-popup').style.display = 'block';
    });

    // Envoyer une requête POST lorsque le formulaire est soumis
    document.getElementById('add-equipment-form').addEventListener('submit',
function(e) {
        e.preventDefault();

        var id = document.getElementById('id').value;
        var latitude = document.getElementById('latitude').value;
        var longitude = document.getElementById('longitude').value;

        $.post('add_equipment.php', {id: id, latitude: latitude, longitude:
longitude, etat_piege: 1}, function(data) {
            alert('Équipement ajouté avec succès !');
            location.reload();
        });
    });
</script>

<div id="osm-map"></div>

<div id="footer">
   <p>Légende :</p>
  <p>Piège déclenchés : <span class="red-dot"></span></p>
  <p>Piège non déclenchés : <span class="green-dot"></span></p>
</div>

<script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
<script>
    var element = document.getElementById('osm-map');
    var map = L.map(element);
    var markers = [];
```

```
    L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
        attribution: '© <a href="http://osm.org/copyright">OpenStreetMap</a>
contributors'
    }).addTo(map);

    var target = L.latLng('44.9302', '2.4457');
    map.setView(target, 14);

    var redIcon = new L.Icon({
      iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-red.png',
      shadowUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    var greenIcon = new L.Icon({
      iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-green.png',
      shadowUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    // Parcourir le tableau de données et ajouter les marqueurs à la carte
    var data = <?php echo json_encode($data); ?>;
    for (var i = 0; i < data.length; i++) {
        var id = data[i][0];
        var lat = parseFloat(data[i][1]);
        var lon = parseFloat(data[i][2]);
        var etat_piege = parseInt(data[i][3], 10); // Assurez-vous que etat_piege
est un entier
        var icon = (etat_piege === 1) ? redIcon : greenIcon; // Red icon for
etat_piege 1, green icon for 0
        var marker = L.marker([lat, lon], {icon: icon}).addTo(map);
        marker.bindPopup("ID: " + id + "<br>Latitude: " + lat + "<br>Longitude: "
+ lon);
        markers.push(marker);
    }

    // Fonction pour réinitialiser les points
    document.getElementById('reset-points').addEventListener('click', function() {
        markers.forEach(function(marker) {
            map.removeLayer(marker);
        });
        markers = [];
    });
</script>

</body>
</html>
```

Annexe générale du projet : Lien vers des documentations

- **Passerelle LORA Dragino DLOS8 :**
  https://www.dragino.com/downloads/downloads/LoRa_Gateway/DLOS8/DLOS8_LoRaWAN_Gateway_User_Manual_v1.2.pdf

- **THE THINK NETWORK SANDBOX :**
  https://www.thethingsnetwork.org/docs/network/

- **PHPMyAdmin :**
  https://www.phpmyadmin.net/docs/

- **WampServer64 :**
  https://portfoliobastienblog.files.wordpress.com/2017/02/doc-wampserver.docx