09/05/2020

Thomas Paredes - Benoît Cavallo

Licence professionnelLE IOtia -2019/2020

Rapport Technique du Projet MBOT

[Sous-titre du document]

**Rapport technique du Projet Mbot**

Table des matières

[Introduction 2](#_Toc40025837)

[Cahier des charges du projet 2](#_Toc40025838)

[Description générale du projet 2](#_Toc40025839)

[Schéma d’architecture : schémas matériels et logiciels 2](#_Toc40025840)

[Décomposition des différentes fonctionnalités 4](#_Toc40025841)

[Fonctionnalités 4](#_Toc40025842)

[Scénario/acteur/rôles 5](#_Toc40025843)

[Répartition du travail 6](#_Toc40025844)

[Résultats de notre Projet (par cours) : 8](#_Toc40025845)

[Fonctionnement du mbot: 8](#_Toc40025846)

[Récupération des données à partir du robot : 12](#_Toc40025847)

[Mise en place de l’appli web avec visualisation des données : 16](#_Toc40025848)

[CONCLUSION 21](#_Toc40025849)

# Introduction

Nous nous retrouvons pour le rapport concernant le projet du nom de Mbot. Ce projet a été mise en œuvre dans le cadre de la licence professionnelle IOTIA par les élèves *Benoît Cavallo* et *Thomas Paredes*. Nous verrons dans une première partie le cahier des charges générale du projet avec les différentes fonctionnalités, l'archi matérielle, logicielle et les différents scénarios d’usages. Par la suite dans une seconde partie nous verrons les résultats obtenus lors de ce projet.

# Cahier des charges du projet

## Description générale du projet

Le projet mBot consiste à programmer un robot de type mBlock pour intervenir dans des zones difficiles (incendie) et prélever certaines données. Les données prélevées seront des mesures de température et la position GPS de celle-ci. Ensuite, un site web permet d’exploiter et de visualiser ces données.

## Schéma d’architecture : schémas matériels et logiciels

**Matériels :**

Robot « Mbot » de MBlock.

Capteur Ultrason.

Carte Arduino mega 2560 Me Auriga.

Carte LinkItOne.

Capteur de température.

Module GPS.

**Logiciels :**

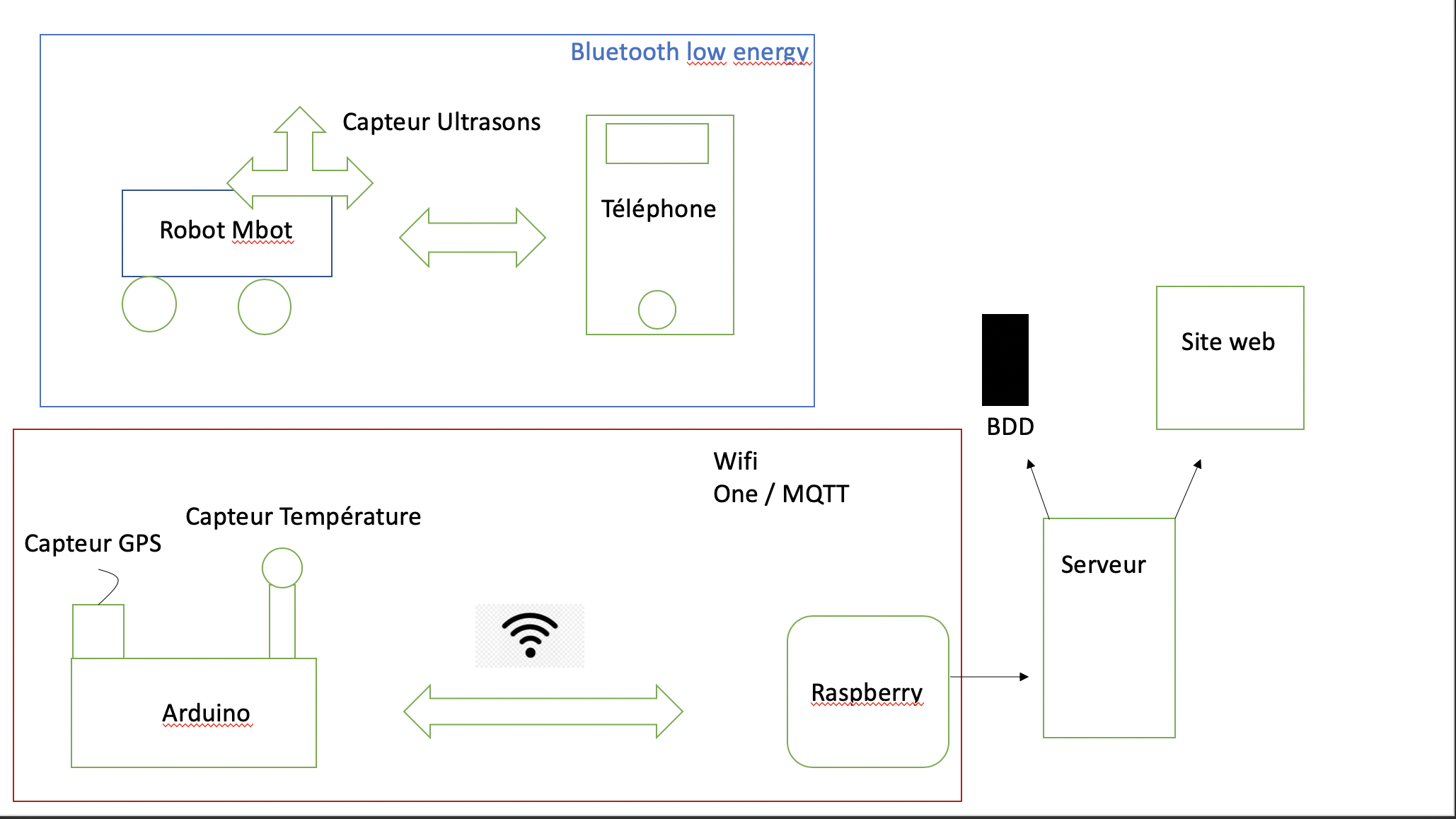
MBlock (IDE)

Arduino (IDE)

Application téléphone MBlock

Site web (Local)

BDD (MySql)



## Décomposition des différentes fonctionnalités

L’application permettra d’illustrer les 3 modules de l’UE 63 :

Module “Connexion entre Objets” de Cécile Belleudy :

Pour l’utilisation du robot nous allons programmer les déplacements du robot avec le capteur à ultrasons pour se déplacer dans l’environnement. Le capteur de température et de position (GPS) transmette les données par la LinkItOne en wifi.

Module “Développement Web Service” de Jean-Michel Bruneau :

Les données collectées seront stockées dans une BDD et visibles via le site web dont l’accès sera sécurisé en https et/ou login mot de passe.

Module “Internet des objets” de Marie-Agnès Peraldi : Mise en place d’une infrastructure MQTT.

## Fonctionnalités

Cette partie regroupe les plus grosses fonctionnalités de notre projet à développer.

*Fonction 1* : contrôler les déplacements du robot à l’aide d’une application sur l’arduino (direction avant, arrière, gauche et droite). Le code est en arduino C. Les communications se font via bluetooth low energy.

→ Test Fonction 1 : définir un motif de déplacement codé en dur sur l’arduino et contrôler le robot avec cela.

*Fonction 2* : collecter les données températures et gps et envoies en wifi au serveur qui sont stockées dans la base de données.

→ Test Fonction 2 : Envoyer des données à stocker dans la BDD, et vérifier si la sauvegarde est bien effective.

*Fonction 3* : consulter les données sur le site web et application mobile qui seront présent sur un raspberry.

→ Test Fonction 3 : accéder aux relevées des données sur le site Web.

*Fonction 4* : Accès depuis l’internet : Serveur avec port ouvert pour accessibilité depuis l'extérieur.

→ Test Fonction 4 : depuis son téléphone ou un ordinateur, accéder au site Web.

*Fonction 5* : Installation et utilisation de Mosquitto. Il est disponible sur toutes les plateformes et s’installe en quelques minutes. Comme MQTT est un standard, nous pourrons communiquer avec nos objets connectés de la même façon. Publication (envoie) d’un message MQTT depuis le terminal. Recevoir (souscrire) des messages. Et sécurisation avec un mot de passes.

→ Test Fonction 5 : Communication et affichage de la température et données gps.

*Fonction 6* : Module de connexion pour sécuriser l’accès au site Web avec un mot de passe haché.

→ Test fonction 6 : Essayer de se connecter au site sans identifiants ou mot de passe.

## 

## Scénario/acteur/rôles

Scénario 1 :

Fonctions associées : F1

Description du scénario : Un utilisateur peut faire avancer le robot.

Scénario 2 :

Fonctions associées : F3

Description du scénario : Un utilisateur collecte les données de température associée à leurs positions GPS respectives.

Scénario 3 :

Fonctions associées : F4

Description du scénario : Un utilisateur accède au site Web en local et visualise les données de température associée à leurs positions GPS respectives.

Scénario 4 :

Fonctions associées : F5

Description du scénario : Un utilisateur accède au site Web depuis n’importe où.

Scénario 5 :

Fonction associée : F6

Description du scénario : Visualisation des messages (échange) sur Mosquitto, depuis n’importe quelle plateforme.

Scénario 6 :

Fonction associée : F7

Description du scénario : une personne mal intentionnée veut se connecter au site, mais il est empêché par le module de connexion.

## Répartition du travail

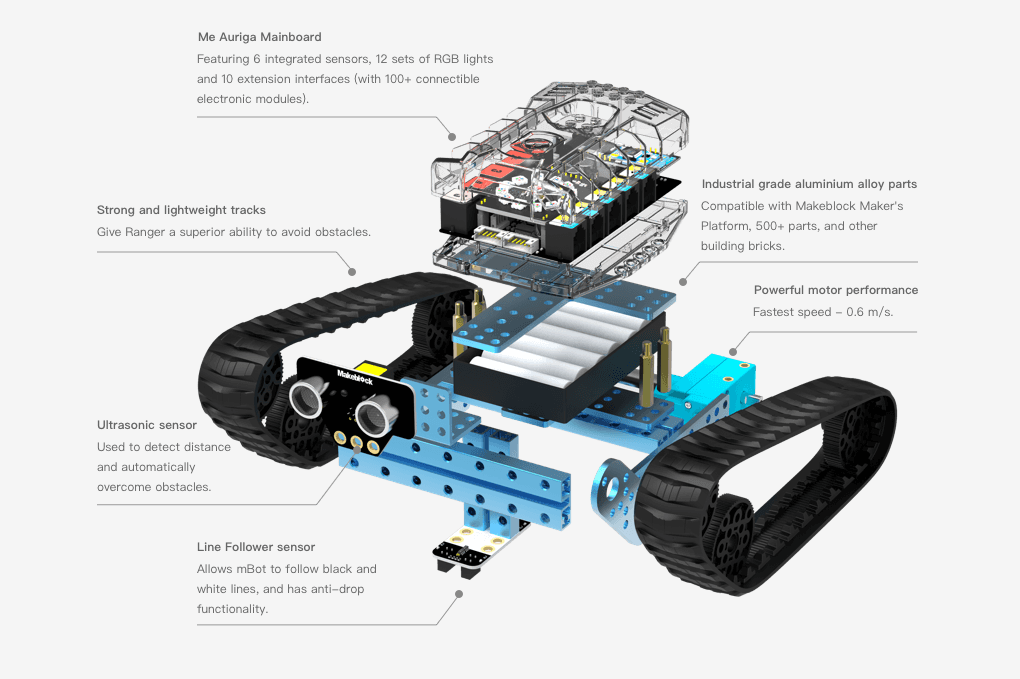
|  |  |  |  |
| --- | --- | --- | --- |
| Tâches | Etudiant | Dates | Etat d’avancement |
| Réalisation du CDC | Benoît/ Thomas | 13/03/2020 |  |
| Fonctionnement du robot Mbot, avancement, recul (utilisation du capteur ultrason) | Benoît/ Thomas | 13/03/2020 jusqu’au 21/03/2020 | En utilisant le langage dédié à l’équipement on arrive à déplacer le robot. |
| Communication entre arduino et raspberry(récupération des différentes donnés) | Benoît | Démarrage le 19/03/2020 | Travail à démarrer le 3/4 |
| Mise en place de l’appli web avec visualisation des données | Thomas/Benoit | Démarrage le 19/03/2020 | Thomas à développer un board avec les températures et une map avec des coordonnées de la position GPS du robot. Le site web et la BD sont sur le RPI. La BD sera en mysql  Pour le moment la BD ne fonctionne pas correctement. Interface à construire ‘  Revoir l’interface et les différents pages de connexion et collecte ou affichage  Mettre en place le https.avec redirection pour page d’accueil |
| Mise en place d’un module de connexion au site sécurisé hachage des données | Thomas | Démarrage le 26/03/2020 |  |
| Adaptation de la visualisation en mobile | Thomas | Démarrage le 03/04/2020 | Responsive à voir avec la nouvelle page |
| Insertion dans une plateforme MQTT mosquitto installer sur raspberry | Benoît/ Thomas | Démarrage le 03/04/2020 | Sur l’arduino (client mqtt) sur l’arduino et grâce à une librairie :  On enregistrera (publish) le capteur GPS et le capteur de température chaque fois qu’il y aura une lecture de ces deux capteurs.  La publication se fait auprès du broker installé sur le RPI (accès sécurisé ou pas).  Sur le RPI il faut un code qui subscribe au broker de ces deux données et à chaque échantillon reçu il envoie sur la BD avec la date |

# Résultats de notre Projet (par cours) :

## Fonctionnement du mbot:

Le robot mbot de son vrai nom *Mbot ranger* dans notre cas est un robot permettant de se déplacer dans un milieu hostile. Il récoltera des données de température et des données GPS. Revenons donc au mbot en lui-même. Ce dernier possède une carte intégrée Arduino mega 2560 dérivé en Me Auriga. Le code réalisé sera donc effectué sur l’IDE arduino.

Pour commencer voyons les caractéristiques principales du robot. Ce dernier possède une multitude de fonctionnalités. En effet mBot Ranger intègre six types de capteurs, notamment un capteur de lumière, capteur de température, capteur de son, capteur à ultrasons, capteur de suiveur de ligne et gyroscope. Cette large gamme de capteurs permet à Ranger d'exécuter une variété de fonctions, notamment la collecte de données, le suivi de ligne et l'évitement d'obstacles. Les enfants peuvent également utiliser Ranger pour travailler sur des projets intéressants, tels que la traversée de l'Antarctique pour aider à la recherche scientifique !

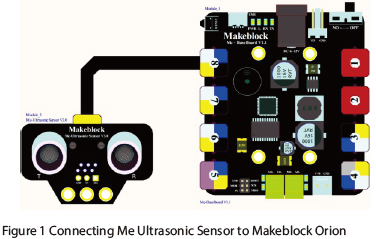


Dans notre cas nous utilisons le capteur à ultrasons pour détecter d'éventuels obstacles, les moteurs pour se déplacer et donc les piles comme batteries.

Mode de câblage capteur à ultrason

Connexion avec RJ25

Étant donné que le port du capteur à ultrasons Me a un ID jaune, on doit se connecter au port avec un ID jaune on peut donc se connecter aux ports n ° 3, 4, 5, 6, 7 et 8 comme suit :



<https://www.makeblock.com/project/me-ultrasonic-sensor>

Par la suite voilà une routine de programme permettant d’utiliser le capteur de son.

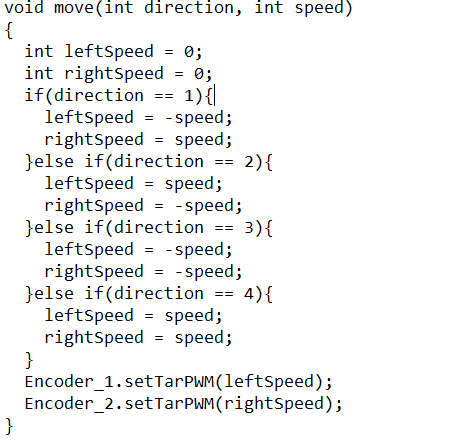


Le moteur quant à lui est obligatoirement placé sur les ports 1 et 2 si on utilise le deuxième. Pour les faire fonctionner on doit contrôler chacun d’eux. Ainsi pour changer de direction ou autres on varie la puissance de chaque moteur. Passons maintenant à l’explication simple de morceaux de codes Arduino permettant d’effectuer des mouvements simples.

Il faut donc créer des variables correspondantes, ces dernières se nomme Encoder\_x avec souvent x comme numéro du slot présent sur la carte Me Auriga.

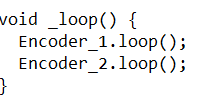


Par la suite il faut implémenter une fonction move qui prendra en paramètre la direction qui est-elle même définie à l'intérieur de la fonction et également la puissance en pourcentage compris entre 0 et 100. Voici la fonction en question :



Il suffit par la suite de mettre en place les fonctions dans le setup et dans le loop pour que cela fonctionne.





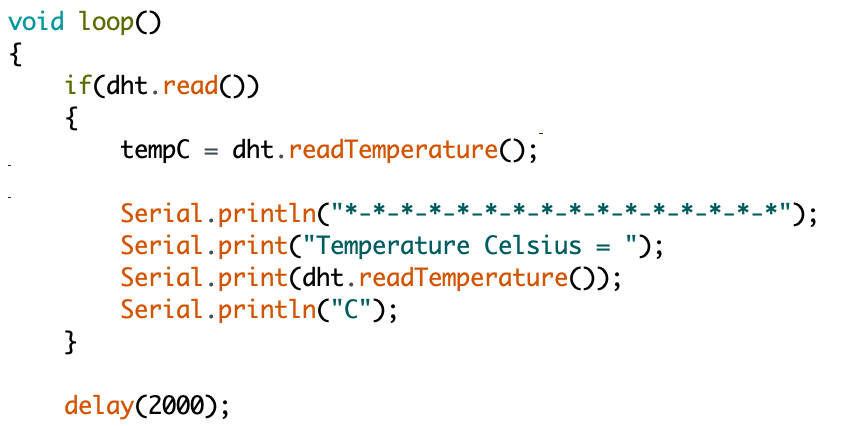
Pour finir avec notre mbot nous avons décidé de ne pas envoyer les données collectées de gps et température avec la carte me Auriga. En effet cette dernière ne possède pas de wifi. Ainsi nous avons pris la décision de placer une autre carte possédant du wifi. Nous avons choisi la Linkit One que nous avons déjà utilisé en cours.

## Récupération des données à partir du robot :

Dans un premier temps, on va récupérer les données suivantes : la température et la position GPS. Ainsi, avec nos capteurs montés sur le robot Mbot on pourra connaître en temps réel la position de celui-ci et la température de l’environnement l’entourant.

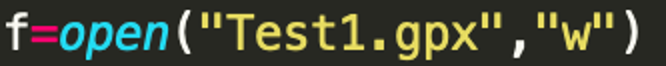
Pour la partie collecte de température, on utilise le capteur DHT11 (*ici, en illustration*).

Ensuite dans le code Arduino, on vient lire sur le pin où est connecté le capteur toutes les 2 secondes la température que celui-ci relève.

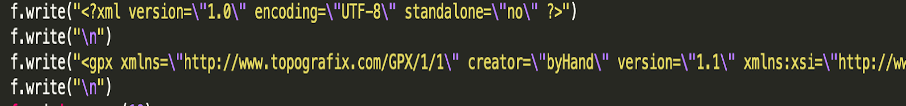


Pour la partie GPS, nous voulons récupérer la position de notre robot au moment où est capturée la température. Nous avons dû principalement récupérer les données envoyées sous forme de trame par l’arduino. En effet après l’installation de différents drivers et la mise en place du GPS sur la carte nous avons pris un exemple sur la récupération de données déjà présentes dans l’IDE arduino. A la suite de cela nous avions les trames présentes sur le port ‘COM’ que nous avions attribué.

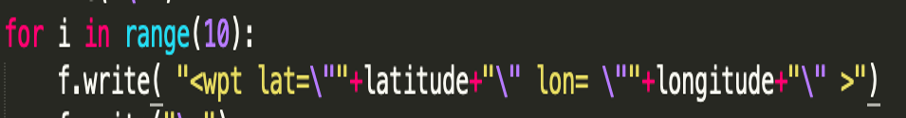
Par la suite nous avons réalisé un code python pouvant donc traiter les données de la trame. Avec les données récupérées nous devions écrire à l'intérieur d’un fichier .gpx à l’aide de la fonction open ci-dessous :



Pour écrire à l'intérieur du fichier il fallait utiliser une syntaxe assez précise avec un entête défini au préalable.



Enfin, on récupère les positions GPS (latitude, longitude etc..) dans notre fichier .gpx. Ici, on limite à 10 avec un timer toutes les 30 secondes pour récupérer la position.



Voilà à quoi correspond principalement le fichier à la fin de ces différentes opérations :



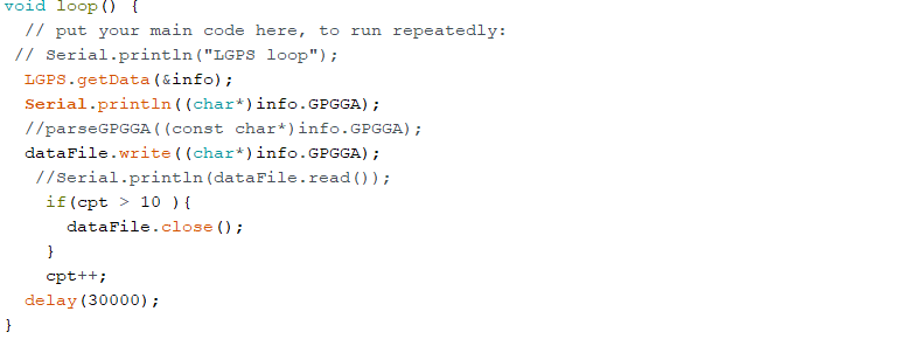
On peut voir dans ce fichier la latitude, la longitude mais également l’heure et l'élévation.

C’est pour cela que le fichier .gpx est destiné à être stocké sur la carte SD de notre LinkItone.

En dessous, le setup de notre arduino pour initialiser notre fichier gpx.



On peut remarquer l’initialisation et l'écriture de la carte SD avec les fonctions begin() et open(). Maintenant dans le loop de notre fichier nous allons écrire à partir des différentes données récupérées.



On initialise un délai de 30 secondes à la fin.

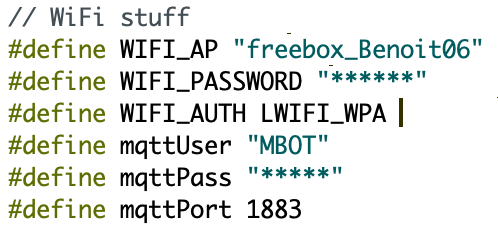
Dans un second temps, il a fallu automatiser l’envoi des ces données vers notre serveur et notre bdd hébergé sur la raspberry pi3. Nous avons donc mis en place un protocole MQTT avec le Broker Mosquitto.

Mosquitto est un serveur MQTT Open Source (Broker) que l’on peut installer sur un Raspberry Pi mais aussi sur presque tous les systèmes d’exploitation (macOS, Windows, Linux…). MQTT est un protocole de communication très rapide et légère particulièrement bien adaptée à la domotique et aux objets connectés. Il facilite la communication entre objets connectés (M2M) tout en économisant la batterie.

L’installation du Broker Mosquitto sur la raspberry est très simple : sudo apt-get install mosquitto.

Côté LinkIt One, on installe la bibliothèque PubSubClient qu’on inclut dans notre IDE Arduino.

Ensuite on définit les informations du réseau Wifi et de notre profil MQTT comme ceci :

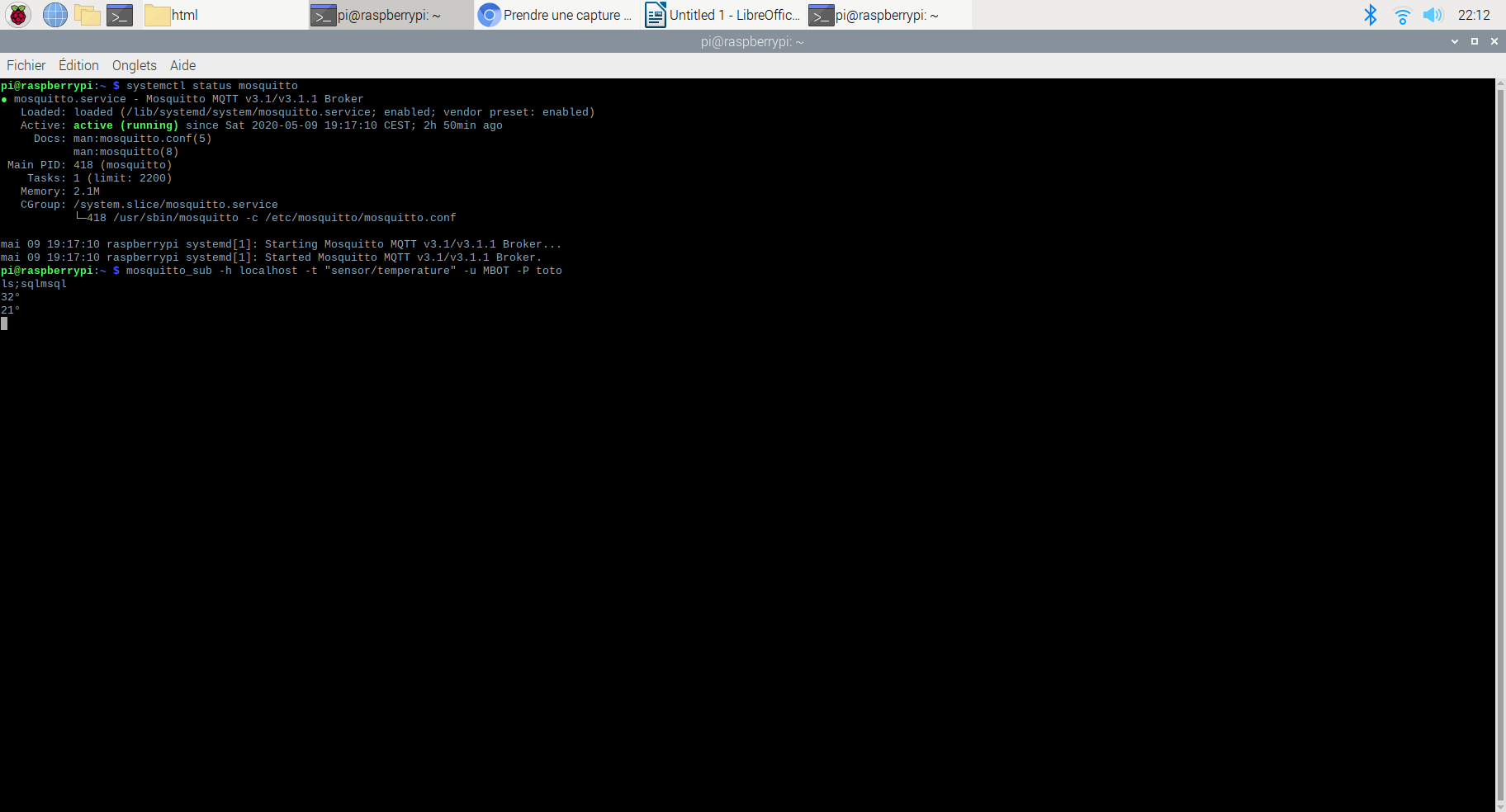




On peut commencer à publier les données sur le broker, et ensuite souscrire sur le raspberry pour les recevoir. Il n’y a plus qu’à les insérer en bdd par la suite.

Avec l’outil MQTTLens, on peut visualiser nos trames de données envoyées et reçues, je mets le screen de ce que reçoit la raspberry en mode susbcribe sur le topic “sensor/temperature”.



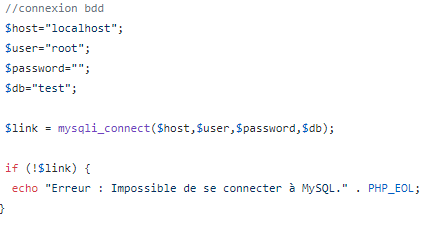


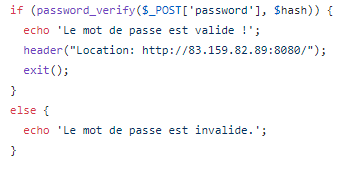
## Mise en place de l’appli web avec visualisation des données :

Dans cette partie nous allons traiter des différentes parties présentes sur l’application web à l’adresse <http://83.159.82.89:8080/>. Cette partie a été codé en local par Thomas Paredes et la mise en place sur le web et serveur web avec la base de données a été réalisé par Benoît Cavallo.Son but est l’affichage des données collecté.

Premièrement le login ou page de connexion au site est une page permettant de filtrer les personnes autorisées à rentrer dans l’application web. Le principe étant simple ce n’est pas correct de placer en clair les mots de passe sur la base de données phpmyadmin, pour cela nous avons placé un mot de passe haché dans notre base et nous effectuons les vérifications à l’aide de fonction php.

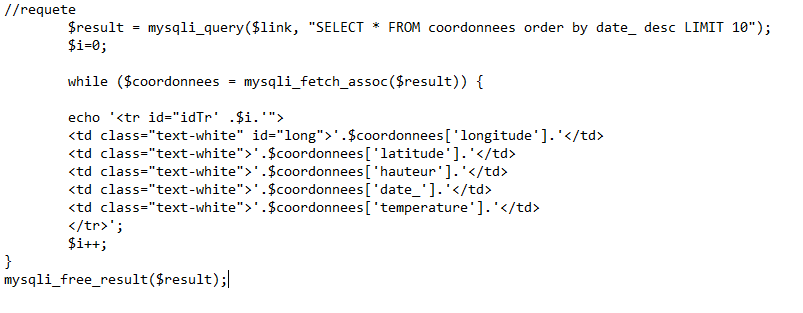
Voici dans un premier temps le code permettant de se connecter à la base de données :



Comme nous pouvons le voir nous utilisons la fonction mysqli dans sa forme procédurale. Dans cet exemple les données de connexion ne sont pas les bons. Ils correspondent aux données en local mais le principe reste le même. Pour la vérification nous utilisons un hash déjà créé que nous plaçons en paramètre pour la vérification du mot de passe placé grâce à la méthode POST, puis nous faisons une redirection vers la page de board.

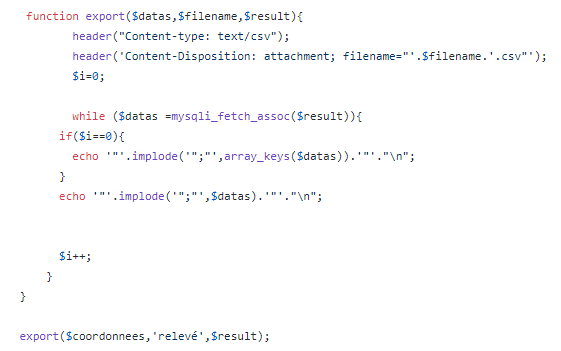
Pour terminer avec cette page niveau html il y a uniquement un formulaire possédant la méthode POST, je vous épargne la lecture de ce dernier mais le code de ce fichier et de tous les suivants seront présents dans le dossier **web** de notre git un fichier **login.php** pour la connexion, un **board.php** pour la page principale mais aussi un fichier **export.php** pour la génération d’un csv.

Parlons maintenant de la page principale ou du board comme nous l’appelons. Cette page contient principalement un menu permettant de passer entre les deux éléments de la page. La première partie est constituée d’un tableau qui comme tout autre élément du site est responsiv qui récupère les données correspondant à la longitude, la latitude, l’id du relevé, la date et la température d’un relevé effectué par le robot. Et une seconde partie correspondant à une map.

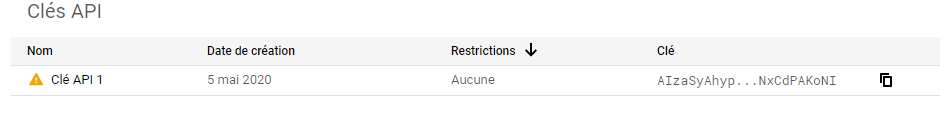


Nous pouvons donc voir une requête que nous limitons aux 10 derniers résultats rentrer dans la base à l’aide du *DESC* et *Limit 10*. Nous affichons par la suite à l’aide d’une boucle les données de la base.

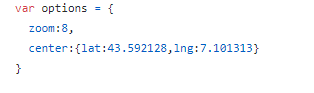
Dans cette même partie nous avons également la possibilité de télécharger le tableau au format csv. Pour cela un bouton pointe vers le lien correspondant au fichier export.php contenant le code permettant de générer un csv. Vous verrez donc ci-dessous de ce paragraphe la fonction correspondante. Elle contient des headers permettant la redirection vers l’enregistrement caché des fichiers en php, mais aussi le nom de génération du fichier. Ensuite nous bouclons en utilisant la fonction implode pour la création ligne par ligne. En paramètre cette fonction prend en compte un tableau de données (dans notre cas le résultat d’une requête sql) le début du nom du fichier mais aussi la requête en elle-même.



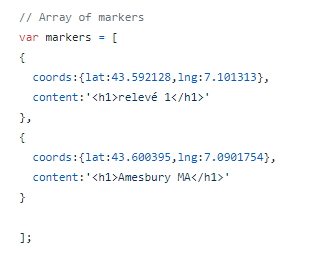
Passons à la seconde partie de la page. Cette dernière est une carte correspondant à une map utilisant l’API google map. Cette dernière affiche la position des relevés fait par le robot. Tout d’abord pour mettre en place la carte il faut s’enregistrer pour obtenir une clé d’API.



Par la suite nous avons créé un fichier map.js qui se situe quant à lui dans le fichier assets et js. Ce fichier correspond à l’initialisation de la map qui est dans ce fichier perçu comme un objet. Ainsi on y initialise plusieurs options comme la taille, la position initiale de la carte comme ci-dessous :

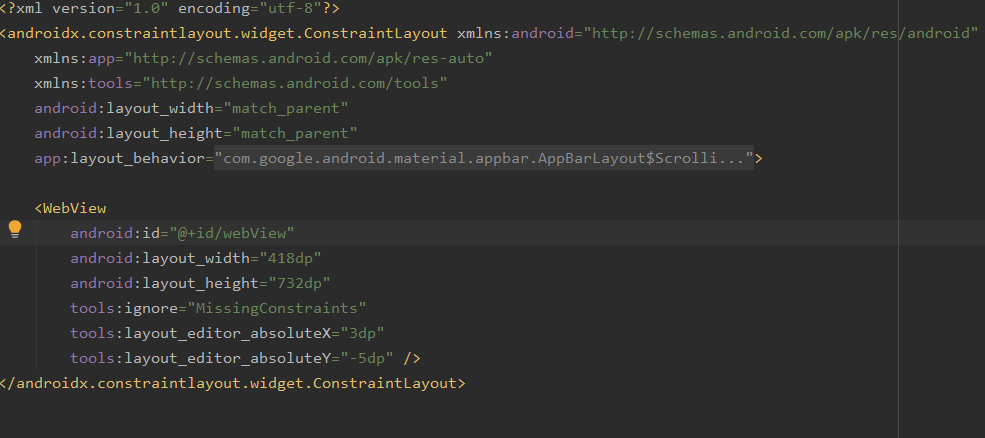


Mais le principal c’est la création de markers qui par la suite sont affiché à la carte. Ils correspondent à des coordonnées et un label.

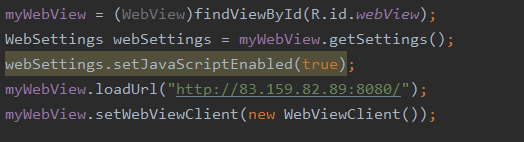


Voilà ce qui en est de cette application web, pour la génération en application mobile cette dernière étant responsiv nous mettons en place l’application sous Cordova à l’aide du plugin cordova-plugin-inappbrowser permettant la mise en place de page web. Cependant ne réussissant pas à installer cordova nous sommes passé sur android studio. Cela était également très simple mais l’application n’est donc pas portable.

Premièrement il faut ajouter une webView dans le content layout comme ceci :



Ensuite dans le mainActivity de java il faut récupérer l’id de la view pour créer l’objet et puis utiliser la fonction loadUrl qui prend en paramètre le domaine et qui permet de charger le site et il ne reste donc plus qu’à souscrire au client web.



Mais pour que tout cela fonctionne il faut par la suite autoriser dans le manifeste l’utilisation de tout ça avec la ligne suivante :



Pour la partie mise en ligne du site, on a utilisé notre Raspberry Pi3 comme serveur Web pour héberger notre site.

Nous avons donc déployé un serveur Apache, après avoir vérifié que celui-ci fonctionnait bien, on a installé PHP puis notre SGBD, MySql (avec mariadb-server et php-mysql qui servira de lien entre php et mysql). Pour l'administration de notre BDD, nous avons choisi PHPmyAdmin.

La dernière étape était de rendre accessible depuis internet notre Raspberry. Il a donc fallu rediriger les requêtes vers le serveur Apache sur Raspbian.

Il faut faire en sorte que lors d’une requête sur la box, qui est la seule à être accessible depuis l’extérieur, celle-ci soit redirigée vers notre Raspberry Pi, pour qu’elle soit traitée par le service adapté (en l’occurrence le serveur Apache2). Pour cela, il va falloir accéder à l’interface de configuration de votre box et ouvrir les ports 443 et 80 pour les protocoles HTTPS et HTTP.

# CONCLUSION

Le projet MBOT a réuni nos connaissances de plusieurs domaines tels que le Web, l’IOT ou encore la connexion entre objets pour mener de bout en bout un projet passionnant. Un site Web qui permet de visualiser l’exploitation de différentes données brutes (Température et GPS). Les données sont directement récoltées sur le terrain grâce à un robot pouvant se déplacer et contrôler à distance. Enfin nous avons pu communiquer et interagir entre différents objets et technologies pour envoyer et échanger des données.

Dans le cadre concret et aboutit de l’utilisation de notre robot, on pourrait imaginer les pompiers l’utiliser dans leurs interventions sur des zones à risques ravagés par les flammes pour récolter des informations de température.