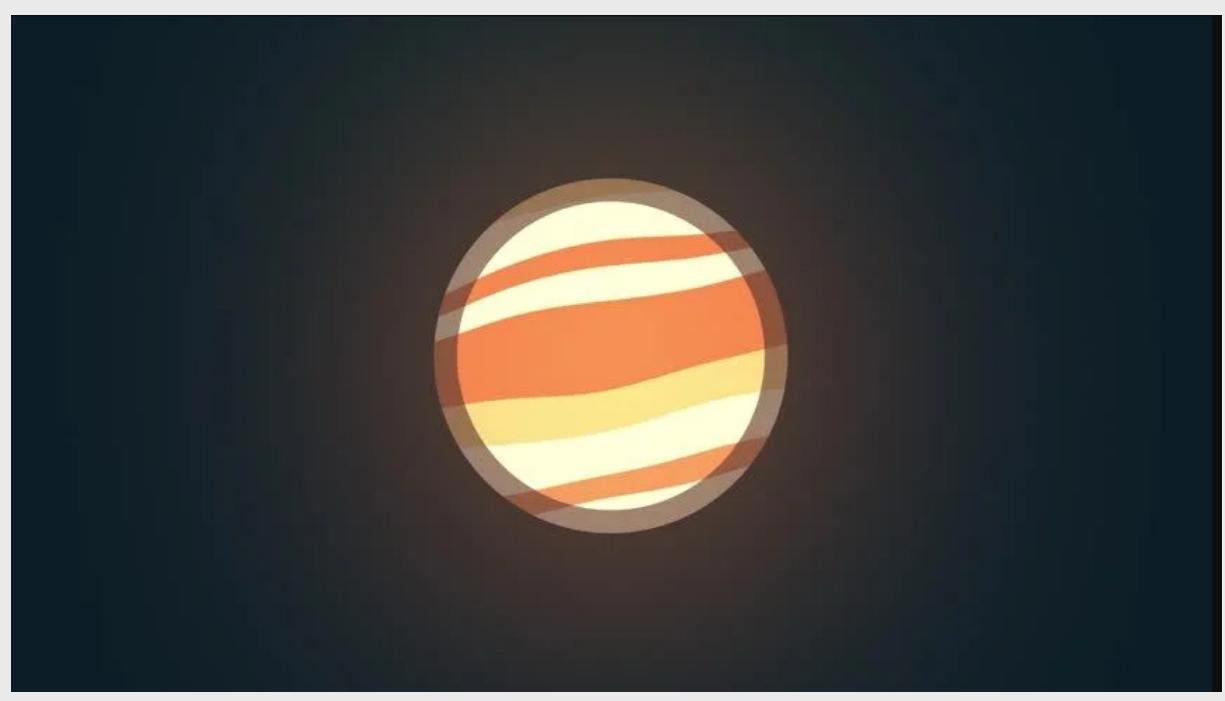


DOCUMENTATION

IOT – PROJET JUPITER

2021



SUPERVISEUR : Monsieur G. MENEZ

GROUPE D'ÉLÈVES :

CHAPOULIE Dorian

LONGIN Rémi

BOLIER Raphaël

GUILLAUMET Léo

SOMMAIRE

Partie I – Présentation du projet

Partie II – Fonctionnalités

Partie III – Installation et mise en place.

Partie IV – Lancement et fonctionnement du robot.

Partie V – Langages, framework et logiciels utilisés

Partie VI – Le matériel utilisé pour la conception du robot

Partie VII – Architecture du robot

Partie VIII – Câblage

Partie IX – Fonctionnement de la Heatmap

Partie X – Les données publiées

Partie XI – Points forts et points faibles

Partie XII – Problèmes rencontrés

Partie XIII – Améliorations possibles

Partie XIV – Conclusion

Partie I – Présentation du projet

Le projet Jupiter est une mission d'exploration des locaux de la MIAGE réalisé par :

- Dorian Chapoulié
- Rémi Longin
- Léo Guillaumet
- Raphaël Bolier

La mission consiste à déployer le robot Jupiter sur le sol afin d'étudier sa surface et de collecter des données concernant la température environnante.

L'objectif final est de transférer les données récoltées vers un utilisateur contrôlant le robot à distance et d'obtenir une heat-map des locaux.

L'utilisateur pourra alors percevoir un flux vidéo généré à partir d'un ESP32-CAM installé sur le robot Jupiter.

Partie II – Fonctionnalités

1. Récupération de la liste des robots disponibles

Un des avantages de ce projet se trouve dans la possibilité de contrôler un robot parmi une liste de robots disponibles. En effet, plusieurs robots peuvent être connectés en même temps et n'importe lequel peut être contrôlé si bien sur celui-ci est disponible.

Dans ce cas, notre robot sera un client MQTT qui sera abonné à un topic consacré au listage des robots.

Lorsque le robot recevra un message provenant du topic de listage, il répondra en indiquant s'il est disponible ou non (un robot est considéré disponible dès lors qu'il n'est pas en train d'être contrôlé).

En résumé, il faut commuter d'un robot à l'autre.

Grâce à cette méthode, vous pouvez choisir de contrôler le robot de votre choix, et être sûr d'être le seul contrôleur de celui-ci.

2. Prise de contrôle d'un robot

Lorsqu'un utilisateur souhaite contrôler le robot depuis notre interface web, un appel sur une route sera effectué à une API en Node.JS, pour que celle-ci se connecte au serveur TCP du robot. Si la connexion est réussie, alors l'interface web passe en mode « Contrôle », sinon un message d'erreur est affiché.

La partie front est reliée à notre API grâce aux websockets. Cela permet d'envoyer les commandes au robot en temps réel.

3. Les différents contrôles possibles

L'utilisateur peut contrôler le robot via notre interface web. Les contrôles sont répartis en 3 catégories :

1. Le contrôle de la direction
2. Le contrôle de la motorisation
3. Le contrôle de la caméra embarquée

4. Les données publiées

Le robot va publier périodiquement les données du capteur de température et la puissance en DB des réseaux l'entourant sur un topic MQTT.

L'API étant abonnée à ce topic, elle reçoit ces données et va les stocker en base, ce qui permet de générer des « heat-map ».

Grâce à la liste des réseaux aux alentours du robot, nous pouvons utiliser une “triangulation” de celui-ci. Nous allons donc faire une estimation de la position. De plus, comme nous avons la température à une certaine position, nous pouvons générer une heatmap en temps réel.

Notre interface web sera principalement composée de 2 modes :

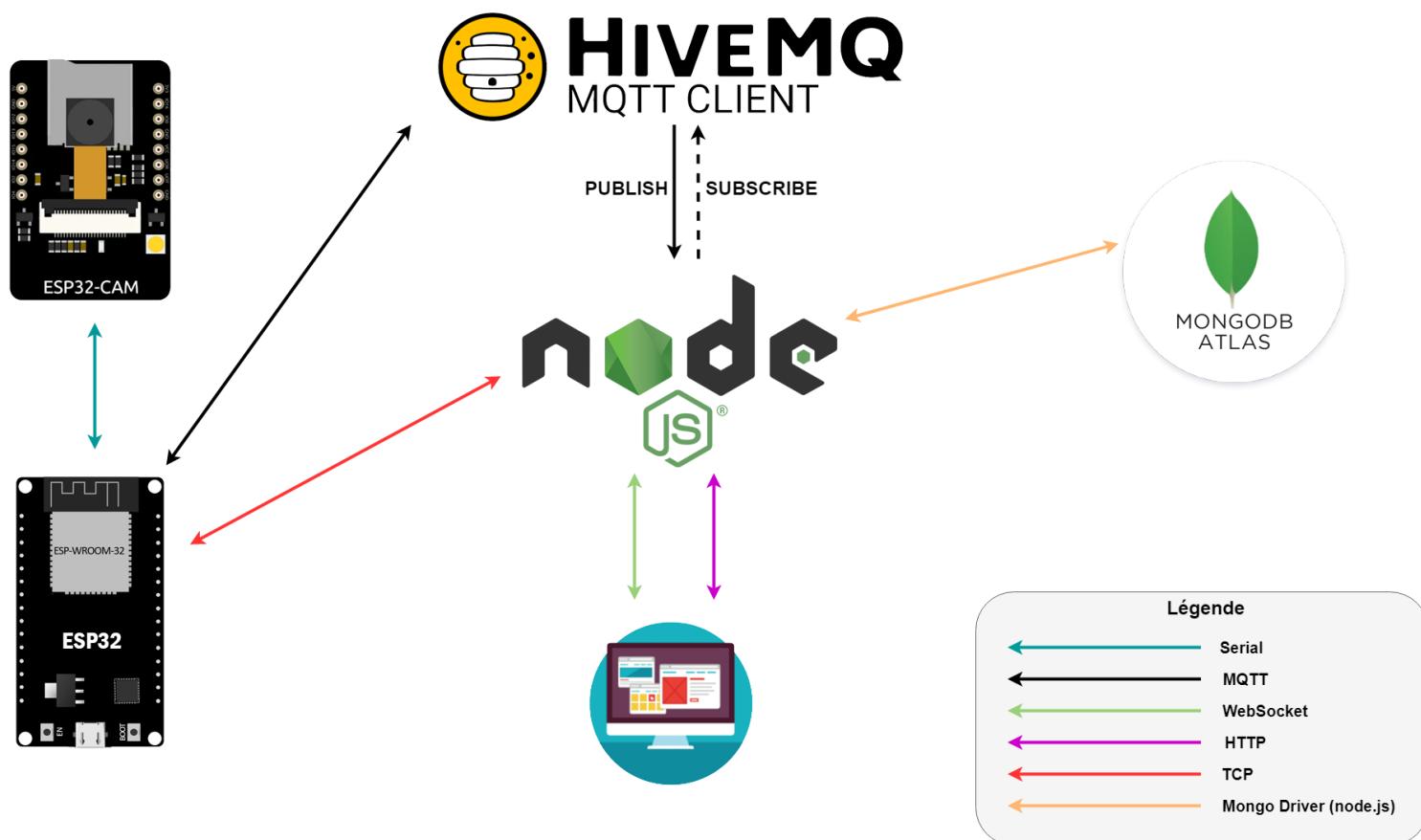
1. Le mode Contrôle
2. Le mode Spectateur

Le mode contrôle permet de contrôler un robot, et le mode spectateur permet de voir la création de la heatmap en temps réel.

5. Tableau des protocoles TCP

Protocol	Description
I-D-<valeur>	Emetteur : API Recepteur : Jupiter Description : Protocole permettant de contrôler la direction du robot Paramètres : Q/D Exemple : I-D-D => aller à droite
I-M-<valeur>	Emetteur : API Recepteur : Jupiter Description : Protocole permettant de contrôler la motorisation du robot Paramètres: A/B/Z/S Exemple: I-M-A => augmenter la vitesse
I-C-<valeur>	Emetteur : API Recepteur : Jupiter Description : Protocole permettant de contrôler la caméra du robot Paramètres : H/B/G/D Exemple : I-C-G => déplacer la caméra à gauche

6. Schéma fonctionnement global



Partie III – Installation et mise en place.

1- L'installation logicielle d'Arduino :

Au sein de notre projet IOT, il est nécessaire de mettre en place Arduino avec quelques manipulations détaillées ci-dessous.

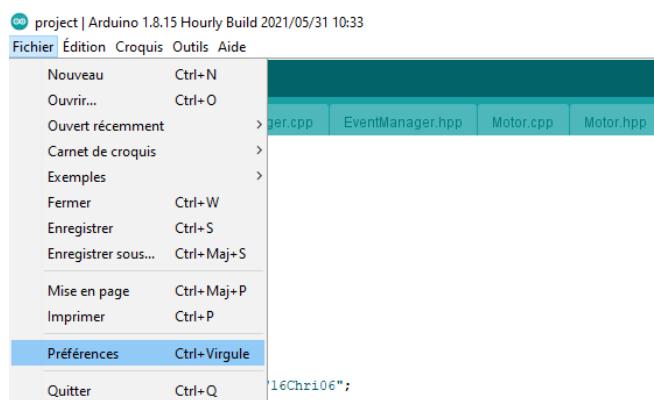
Dans un premier temps, il est nécessaire de télécharger la version officielle d'Arduino sur le site et non sur le Microsoft Store afin de ne pas entraîner de conflit avec la version Microsoft.

Lien vers la version à télécharger pour le projet : <https://www.arduino.cc/en/software>.

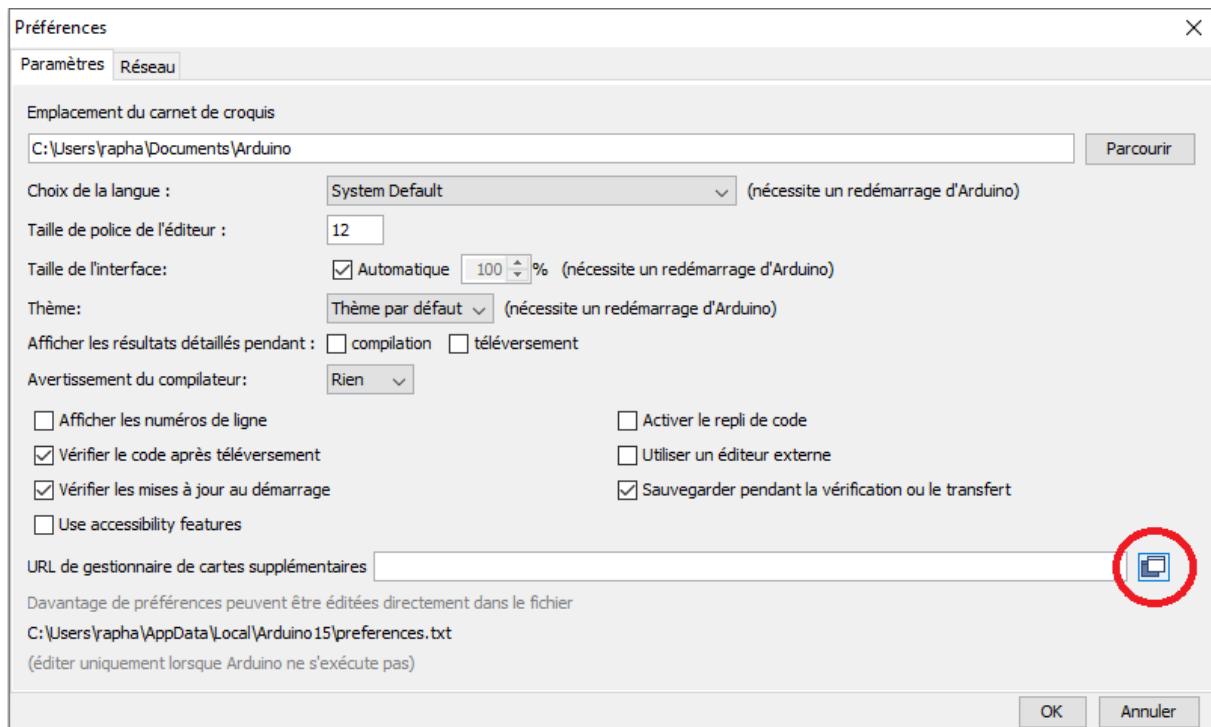
Une fois cette version téléchargée, il n'est pas directement possible d'utiliser Arduino pour notre projet. En effet, notre projet requiert un SDK particulier pour fonctionner, il va donc falloir installer ce SDK.

INSTALLATION DU SDK

Étape 1 : Ouvrir le panneau des préférences depuis le menu Arduino.



Étape 2 : Cliquer sur le bouton situé à droite de l'option « URL de gestionnaire de cartes supplémentaires »

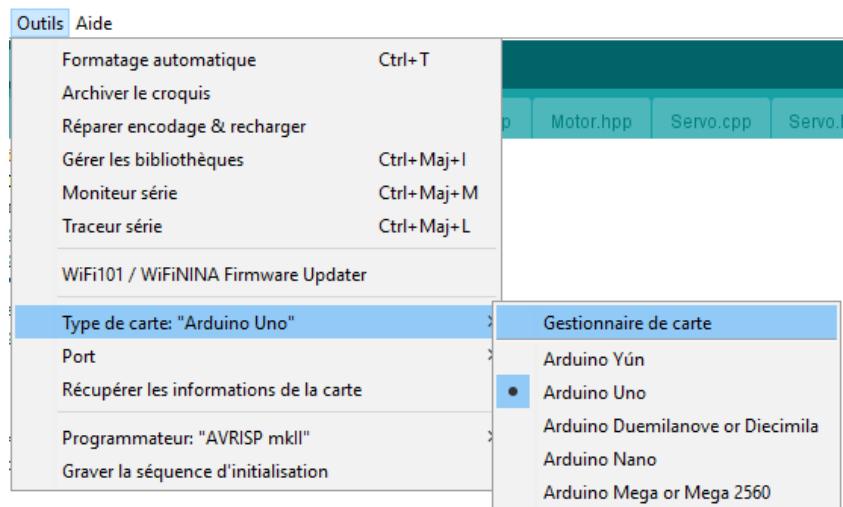


Étape 3 : Insérer le lien dans la boîte de dialogue :

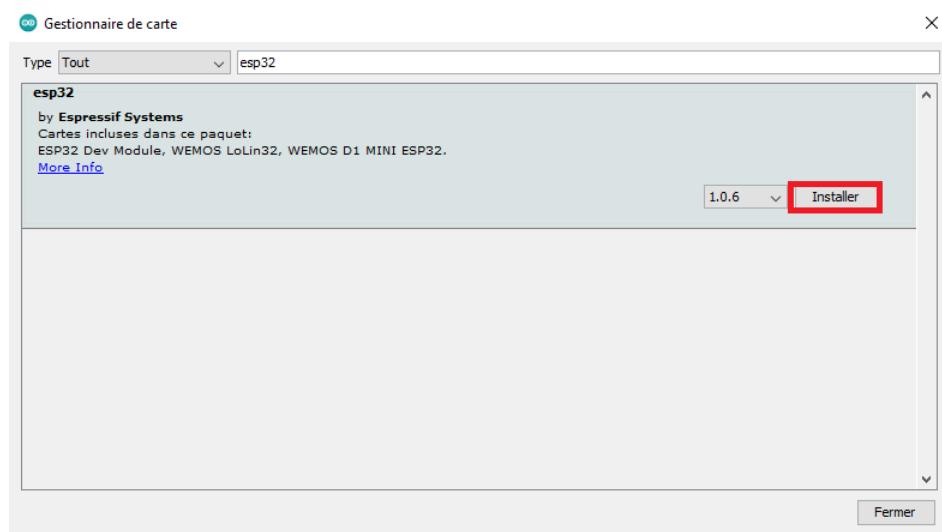
https://dl.espressif.com/dl/package_esp32_index.json

Cliquer sur OK

Étape 4 : Ouvrir le gestionnaire de carte depuis le menu « Outils » puis « Type de carte »



Étape 5 : Cherchez le SDK avec le mot clé « esp32 » et installez le.

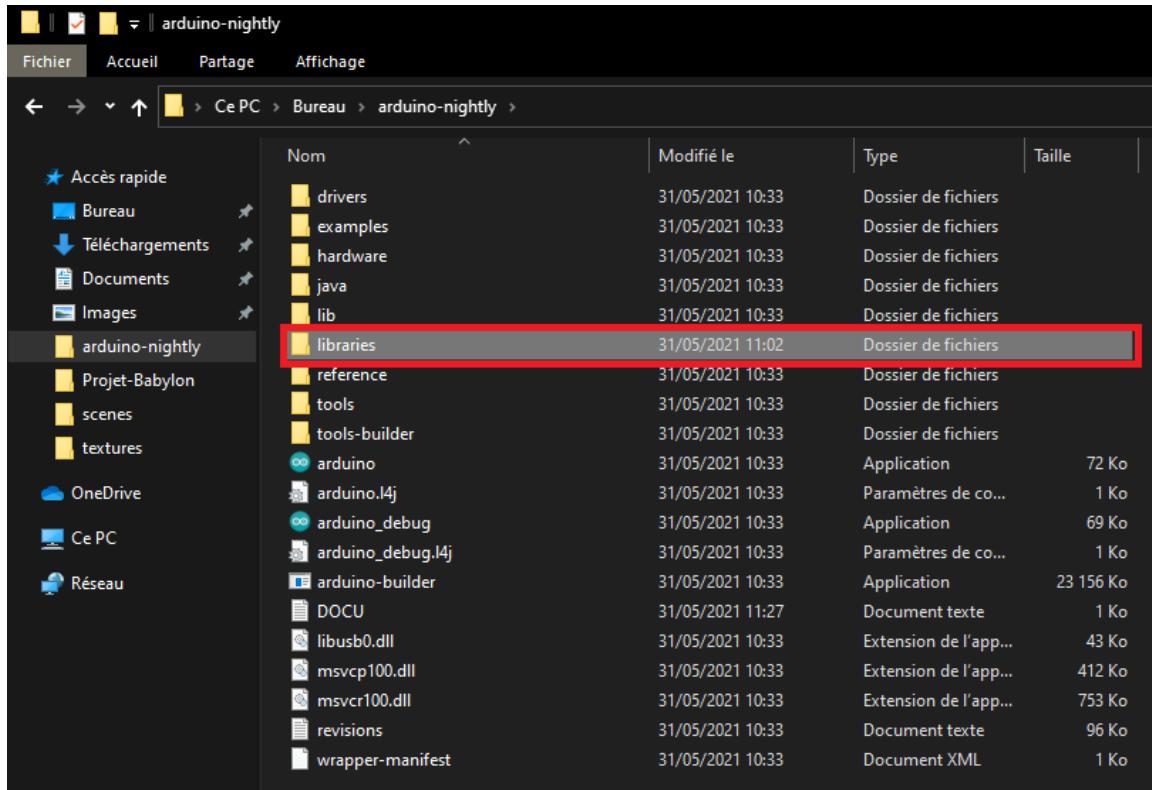


Votre SDK est désormais correctement configuré pour notre projet !

2- Ajout des librairies :

Une fois Arduino correctement configuré, il est nécessaire d'ajouter les librairies pour le fonctionnement du projet.

Pour ce faire, simplement glisser les librairies disponibles sur le GitHub du projet vers le dossier librairies d'Arduino.



Les librairies sont correctement configurées pour notre projet !

3- Configuration du réseau de l'ESP :

Afin que notre ESP puisse se connecter à internet et puisqu'il ne dispose pas de puce 4G, il est nécessaire de le connecter à votre Wifi.

Pour ce faire, ouvrez le fichier « projet.ino » contenu dans :

« Esp -> project »

Modifiez vos noms de réseau et mot de passe.

Il faut faire la même manipulation pour l'ESP32 CAM

```
#include "TCPServer.hpp"
#include "Motor.hpp"
#include "Servo.hpp"
#include "mqtt.hpp"
#include "EventManager.hpp"
#include "Event.hpp"
#define RXD2 16
#define TXD2 17

const String ssid = "Monresau";
const String password = "Monmotdepasse";

TCPServer* server = TCPServer::getInstance();
```

Sauvegardez, votre ESP est prêt à se connecter à votre WIFI.

4- Lancement de l'API et du client :

Une fois votre logiciel mis en place et votre ESP prêt à être connecté, il faut maintenant lancer l'API et le client.

Pour ce faire, assurez-vous dans un premier temps que votre ESP est bien branché à votre ordinateur.

Lancez deux terminaux.

Dans le premier, placez-vous dans la route suivante :

« Projets\projet_iot\Server\api »

Lancez un NPM -i

PATIENTEZ DURANT L'INSTALLATION

Lancez l'API avec un **NPM RUN START**

Dans le second, placez-vous dans la route suivante :

« Projets\projet_iot\Server\front\dashboard »

Lancez un NPM -i

PATIENTEZ DURANT L'INSTALLATION

Lancez le client avec un **NPM RUN START**

A la fin de cette étape, vous devriez arriver sur la page
« Liste des robots ».

Partie IV – Lancement et fonctionnement du robot.

1- Connexion au robot :

Arrivé à cette étape, vous devriez observer la liste des robots connectés à l'API comme suit :

Liste des robots

Nom
▼ 30:AE:A4:8C:1C:04
▼ 24:62:AB:F3:6F:C4
▼ 24:6F:28:22:97:D0
▼ 24:0A:C4:62:5E:A8



Il est possible à partir de cette page de dérouler la liste des robots et d'apercevoir la liste des robots connectés et qui sont disponibles.

Liste des robots

Nom	IP	Firmware	Camera IP
^ 30:AE:A4:8C:1C:04	192.168.1.92:25565	1.0	
Advanced			
▼ 24:62:AB:F3:6F:C4			
▼ 24:6F:28:22:97:D0			
▼ 24:0A:C4:62:5E:A8			



A partir de cette interface vous pouvez actualiser la liste des ESP disponible avec le bouton en bas à droite.

De plus, vous pouvez vous connecter au robot sélectionné avec le bouton suivant :



A noter que la connexion au robot peut prendre un moment.

Lors de la tentative de connexion, survient parfois l'erreur suivante :

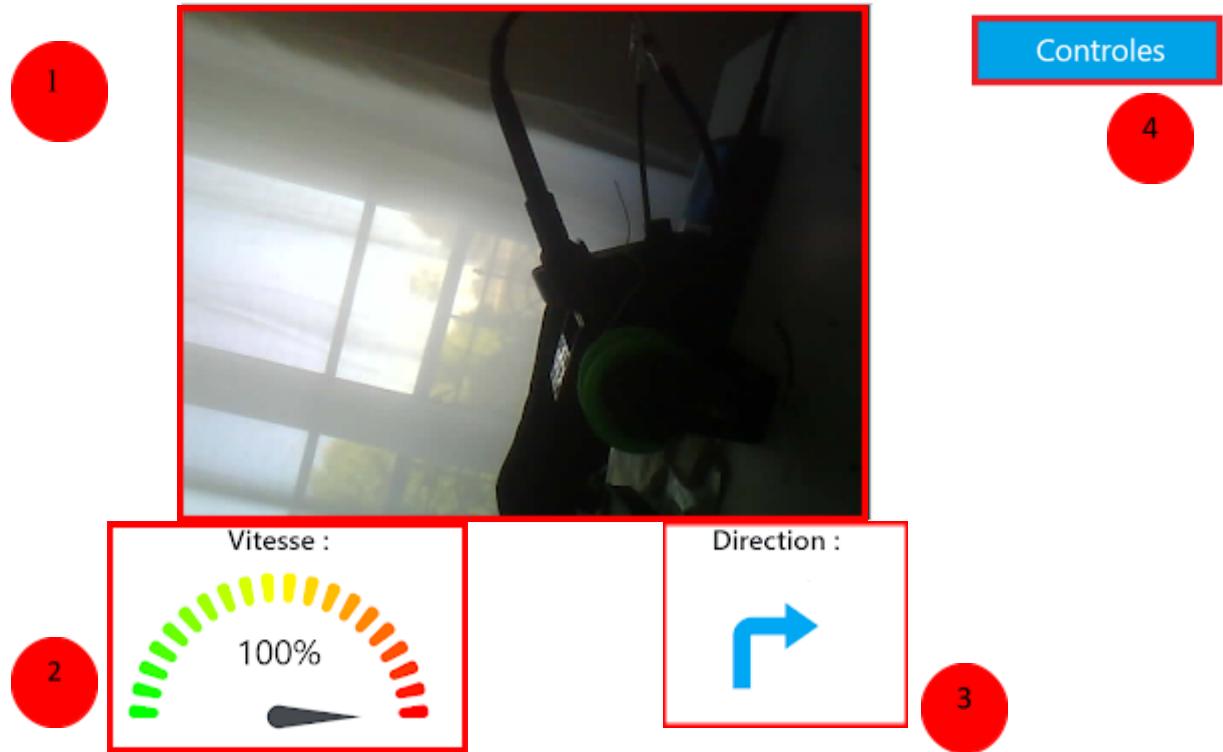
⚠ Le robot n'est plus disponible.

Dans ce cas, nous conseillons de relancer l'API et le client ainsi que de débrancher et rebrancher l'ESP à l'ordinateur.

Si un robot est entrain d'être contrôlé par un autre utilisateur, vous ne pouvez pas vous connecter à celui-ci. Vous recevrez donc le message d'erreur ci-dessus, qui vous informe que le robot n'est pas disponible pour le moment.

2- Interface du robot :

Une fois connecté au robot, vous arriverez sur son interface comme suit :



Plusieurs informations sont donc affichées à l'écran, nous allons les expliquer une par une.

Caméra

Le robot dispose d'une caméra qui retransmet les images en direct visible partie

La caméra est en 4:3 et dispose d'une résolution max de 1600x1200.

La caméra est fixée sur le robot qui peut être déplacée avec les flèches directionnelles.

1

Vitesse

Il est également possible d'appuyer sur la touche MAJ pour faire accélérer le robot d'avantage et CTRL pour ralentir/ freiner.

2

Si vous arrêtez d'appuyer sur la touche avancer (Z) ou reculer (S), le robot s'arrêtera tout seul.

Direction

La direction du robot est affichée dans la zone

3

La direction du robot peut être contrôlée à l'aide des touches de déplacement Z Q S D.

Z : Aller en avant

S : Aller en arrière

Q : Aller à gauche

D : Aller à droite.

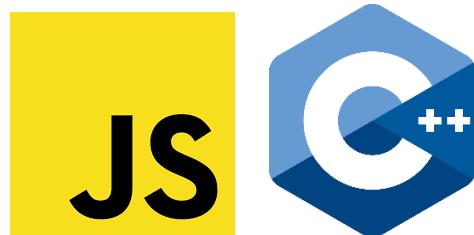
Lorsqu'une direction est donnée au robot avec une touche, celle-ci est montrée à l'utilisateur avec une image.

Enfin, les différents contrôles peuvent être retrouvés facilement en cliquant sur le bouton se trouvant en haut à droite de notre interface.

4

Partie V – Langages, Framework et logiciels utilisés

1- Les langages :



Nous avons utilisé le langage Javascript afin de réaliser les différents programmes de notre interface Web et de notre API.

Pour les programmes des Esp, nous avons utilisé le langage C++.

2- Les Framework :



Pour la partie front de notre interface web, nous avons utilisé le Framework React JS. Puis pour la partie back et la réalisation de L'API, nous avons utilisé le Framework Node JS.

3- Les logiciels :



Pour réaliser ce projet nous avons utilisé les différents logiciels suivants :

- **Visual Studio Code** : L'éditeur de code pour réaliser l'interface Web et notre API.
- **Arduino ide** : Environnement de développement intégré pour écrire les programmes des Esp 32.
- **Fusion 360** : Logiciel 3D permettant la conception de circuits imprimés et donc celle des différents composants du robot
- **Ultimaker Cura** : Logiciel de découpe pour impression 3D, pour imprimer les différents composants du robot.
- **Tinkercad** : Logiciel permettant de réaliser la modélisation de circuit d'esp.

4- Gestion de base de données :



Nous utilisons un Public mqtt broker grâce à HiveMQ qui nous permet d'utiliser un broker gratuitement afin de nous permettre de récupérer les différentes données envoyées par le robot via le protocole MQTT.

Les données sont ensuite récupérées sur le broker grâce à notre api node pour être ensuite envoyées et stockées dans notre base mongo.

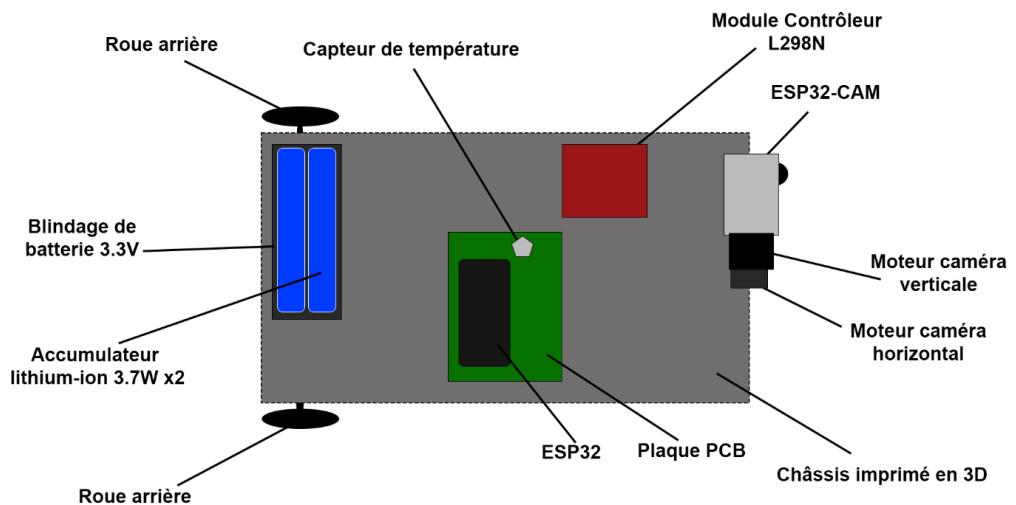
Partie VI – Le matériel utilisé pour la conception du robot

Pour concevoir le robot nous avons utilisé les différents matériaux suivants :

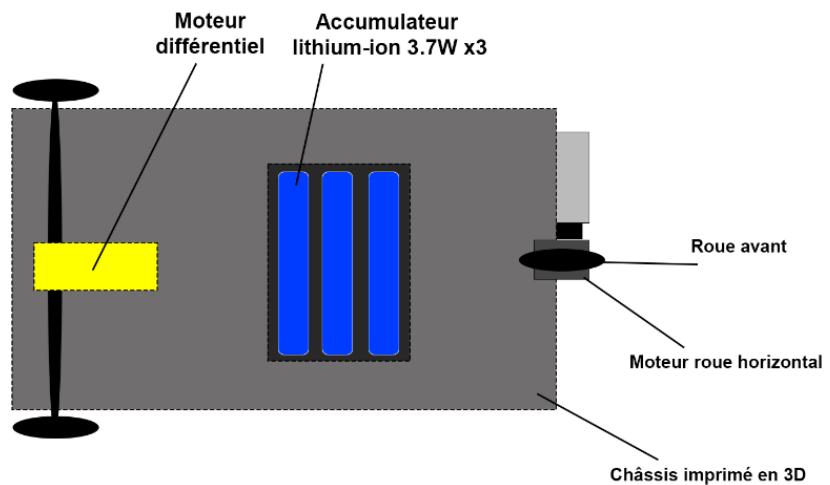
- Un Esp 32 fournit par MIAGE.
- Un Esp 32 cam permettant d'accéder au flux vidéo de la caméra via un réseau WiFi local ou internet.
- Un module contrôleur de moteur L298N permettant le pilotage de moteur à courant continu ou d'un moteur pas-à-pas et donc permet le contrôle du moteur arrière. Par ailleurs, grâce à la PWM, nous pouvons contrôler la vitesse et le sens.
- 2 Batteries : 1 pour le moteur à l'arrière du robot et les servomoteurs puis 1 pour l'Esp 32 et l'Esp 32 cam.
- 4 moteurs : 1 pour les roues arrière, 1 pour la roue avant et 2 pour l'Esp 32 cam.
- 3 roues imprimées : 2 à l'arrière et 1 à l'avant.
- 1 Chassie imprimé qui est le corps du robot.
- 1 plaque de circuit imprimé PCB pour souder les composants et permettre de les maintenir et de les relier électriquement entre eux. (Possède plusieurs avantages : pas de faux contact, plus facilement utilisable, plus petit..)
- Des pins PCB afin de brancher l'Esp 32 sur la plaque PCB
- Des PCB borniers afin de relier les différents fils à la plaque PCB.
- 1 résistance
- 1 Capteur de température

Partie VII – Architecture du robot

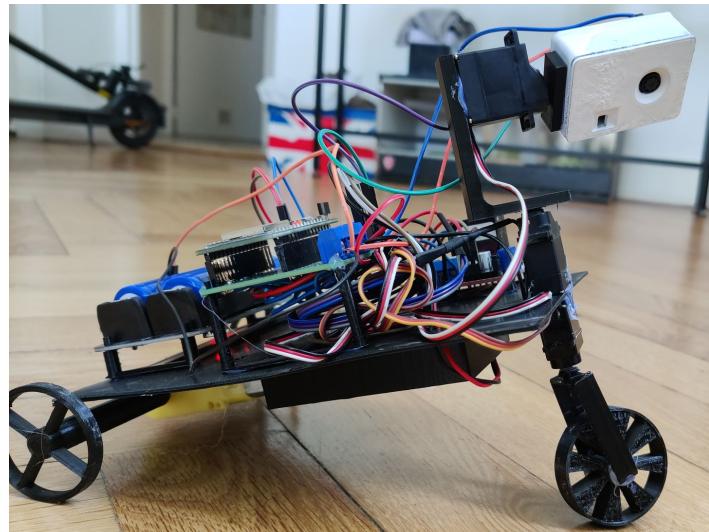
Vue du haut



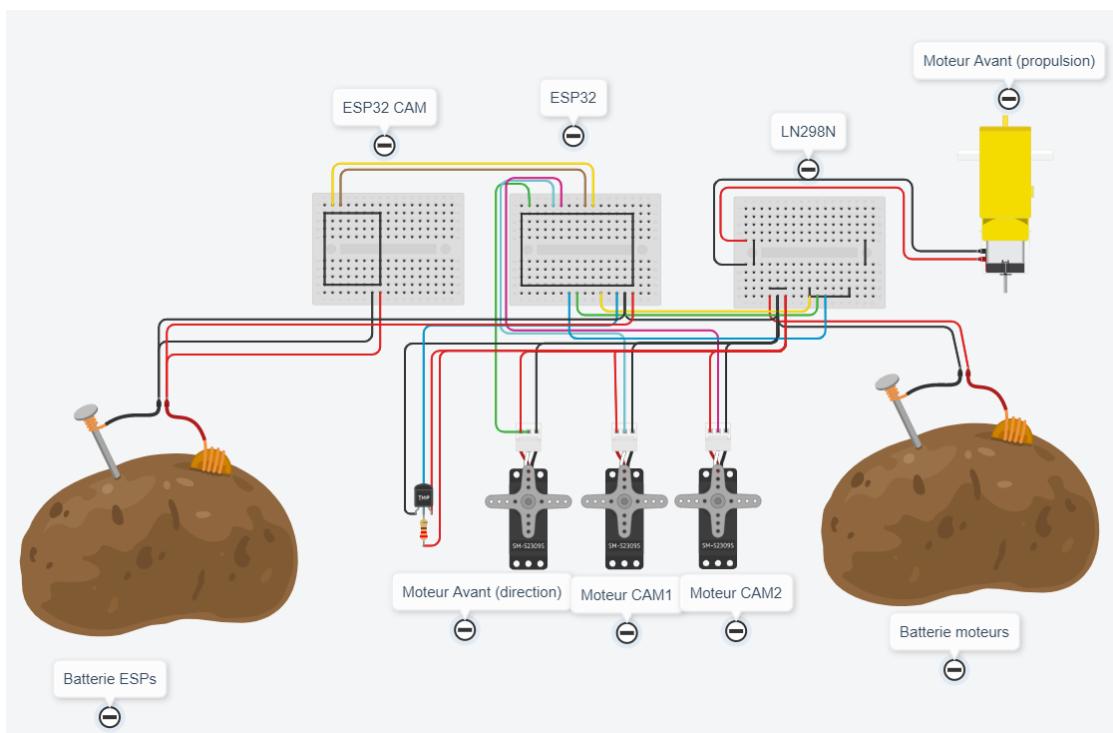
Vue du dessous



Rendu



Partie VIII – Câblage



Cette modélisation a été réalisée sur le logiciel tinkercad et montre de façon simplifiée les différents câblages entre les différents matériaux composant le robot.

Partie IX – Fonctionnement de la Heatmap

Afin de pouvoir récupérer les différentes positions du robot sur le site de la MIAGE nous avons récupéré dans un premier temps la liste de tous les réseaux présents dans MIAGE.

Comme les réseaux de la MIAGE sont composés de plusieurs routeurs permettant d'étendre le signal wifi dans les locaux, il ne fallait pas se baser sur les réseaux SSID mais plutôt sur les réseaux des adresses MAC.

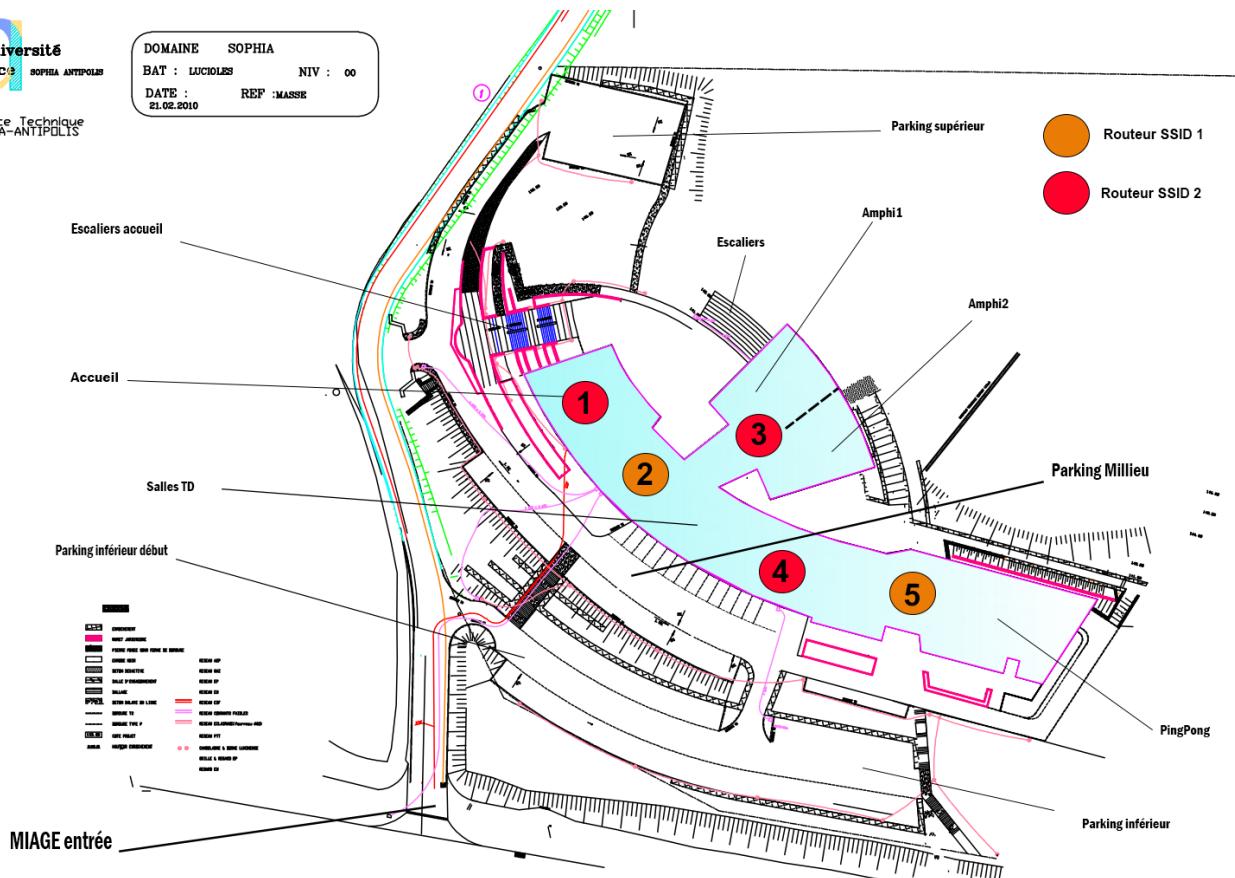
Si on s'était basé sur les adresses SSID on aurait eu plusieurs conflits

Voici un exemple des données recueillies lors de notre quadrillage :

```
{
  "locationName": "EscalierEntree",
  "wifiRSSI": {
    "F8:8F:CA:75:21:E4": "-80",
    "58:97:1E:ED:37:32": "-83",
    "58:97:1E:ED:37:33": "-83",
    "58:97:1E:ED:37:30": "-84",
    "58:97:1E:CB:17:A0": "-88"
  }
},
{
  "locationName": "ParkingSuperieur",
  "wifiRSSI": {
    "E8:65:49:22:C6:B1": "-83",
    "E8:65:49:22:C6:B0": "-84",
    "E8:65:49:22:C6:B3": "-84",
    "E8:65:49:04:D0:93": "-88",
    "E8:65:49:04:D0:90": "-88",
    "38:4B:76:03:DD:4B": "-89",
    "58:97:1E:ED:5B:43": "-91",
    "58:FB:96:27:69:D8": "-91"
  }
},
```

On aura également associé à chaque lieu des coordonnées sur la heatmap

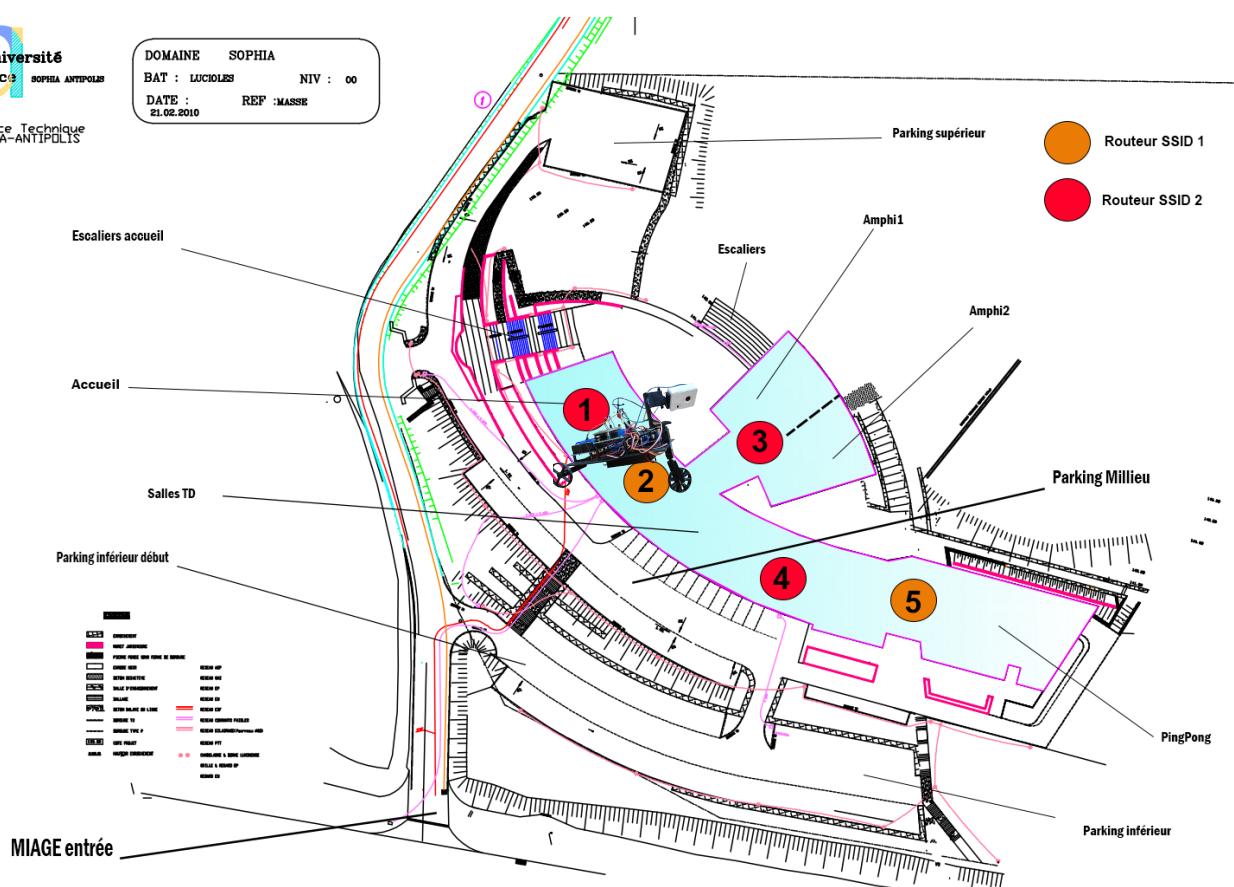
```
EntréeMIAGE: {
  x: 400, // x coordinate of the datapoint, a number
  y: 610, // y coordinate of the datapoint, a number
  value: 10, // the value at datapoint(x, y)
  ssid: ""
},
ParkingMilieu: {
  x: 520, // x coordinate of the datapoint, a number
  y: 450, // y coordinate of the datapoint, a number
  value: 10, // the value at datapoint(x, y)
  ssid: ""
},
```



Comme on peut le voir sur ce schéma ci-dessus des plans de la MIAGE, on peut retrouver plusieurs routeurs à différents emplacements

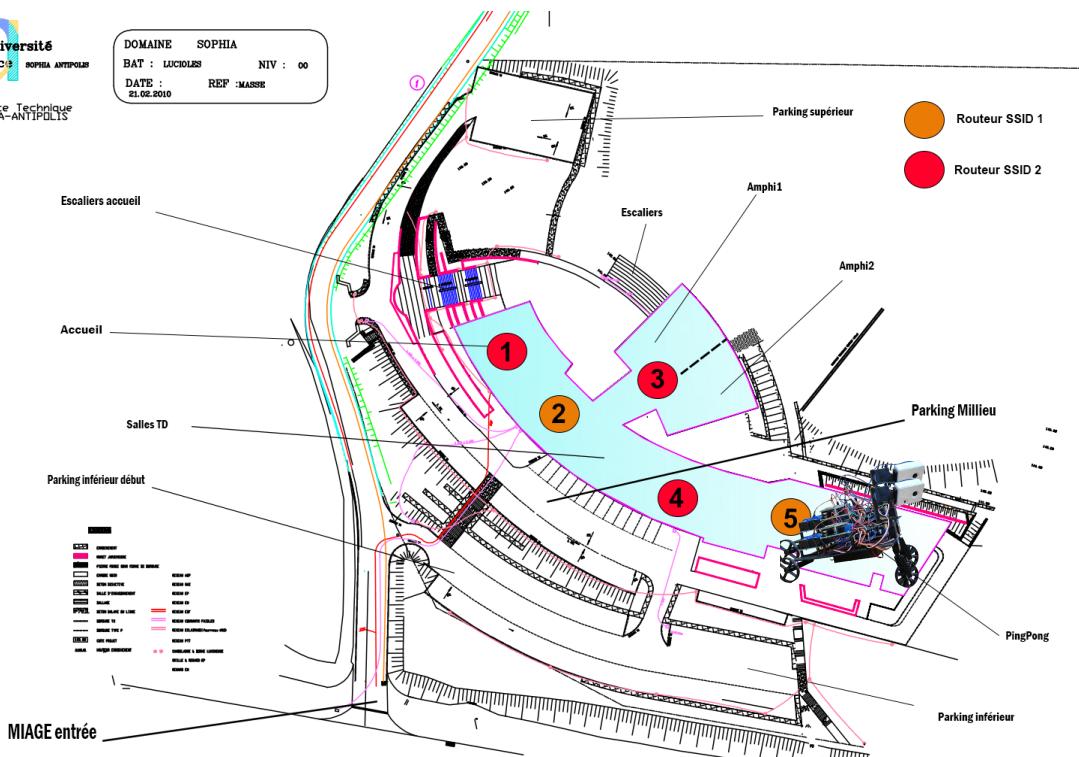
Si le robot ne détecte que le Routeur 1 et 2 on va supposer théoriquement que son emplacement est entre Routeur 1 et 2

DOMAINE SOPHIA
BAT : LUCIOLES NIV : 00
DATE : REF : MASSE
21.02.2010

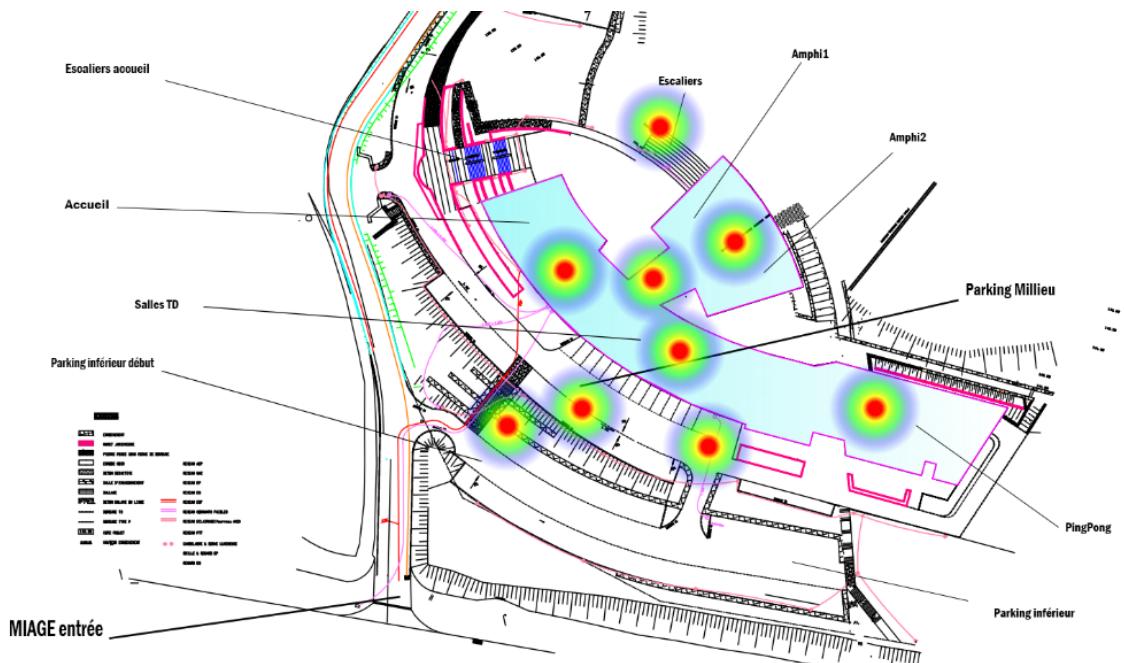


Si cependant il ne détecte qu'une adresse MAC par exemple celle du routeur 5 on va supposer que sa localisation est sur l'emplacement du routeur 5.

DOMAINE SOPHIA
BAT : LUCIOLES NIV : 00
DATE : REF : MASSE
21.02.2010



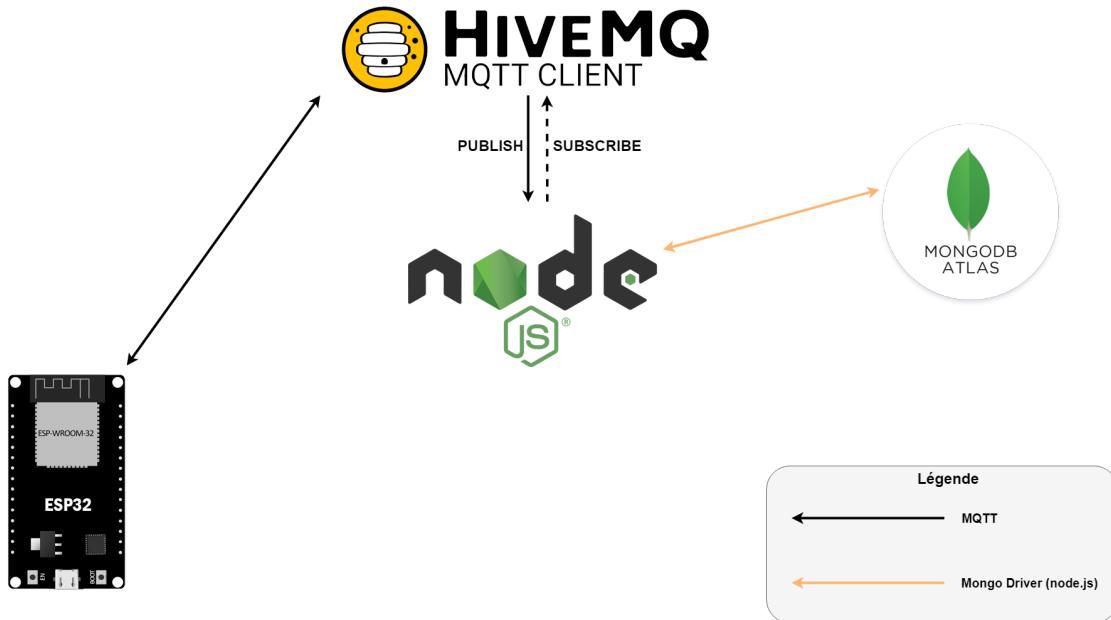
En utilisant toutes les données envoyées par le robot on peut alors afficher une heatmap comme ceci :



(Le point représente sa position théorique et la couleur représente sa température).

Nous avons essayé d'utiliser la puissance RSSI mais nous n'avons pas réussi à trouver de méthode convenante qui nous permettrait l'exploiter correctement. Une amélioration possible serait d'utiliser cette puissance afin de placer plus précisément le robot sur la heatmap.

Partie X – Les données publiées



Afin de recueillir les données que le robot récupère tout au long de son utilisation comme les adresses MAC se trouvant autour de lui ainsi que la température, nous avons utilisé le broker MQTT public "hiveMQ". L'esp-32 va donc envoyer à différents topics mqtt les données collectées. De son côté l'api va s'abonner à ces topics pour récupérer les données et les envoyer à un cluster mongodb Atlas qui va se charger de les stocker.

Voici un exemple des données stockées :

The screenshot shows the MongoDB Atlas interface for the collection **rsssi-wifi-data.rssi-wifi-data**. The collection size is 55.17KB, total documents are 203, and indexes total size is 36KB. The interface includes tabs for Find, Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes (green). A **FILTER** button with the value {"filter": "example"} is visible. The **QUERY RESULTS 1-20 OF MANY** section displays two documents:

```
_id: ObjectId("60b4e9670b54e421d087cb6d")
jupiterID: "24:62:AB:F3:6F:C4"
E8:65:49:39:0E:E0: "-21"
temp: "37.00"
date: 1622468967581
```



```
_id: ObjectId("60b50c7805554fe952d7255c")
jupiterID: "24:62:AB:F3:6F:C4"
58:97:1E:CB:3B:92: "-21"
temp: "37.00"
date: 1622468967581
```

Ainsi il suffit de récupérer les données sur base de données mongodb et de les comparer aux coordonnées des adresses macs issues de notre quadrillage pour afficher sur la heatmap la position théorique du robot, la couleur du point de la heatmap variera également en fonction de la température associée à la localisation.

Partie XI – Points forts et points faibles

Points forts

Tout d'abord, au niveau de la conception du robot, plusieurs points forts peuvent être notés tels que la présence d'un module contrôleur et pilote de moteur qui dispose d'un PWM (modulation de largeur d'impulsion) permettant de faire varier la vitesse et la rotation et donc d'augmenter ou ralentir la vitesse du robot et de permettre la marche arrière. Ensuite, la présence de batterie pour assurer une meilleure autonomie et une meilleure puissance distribuée. La présence d'une plaque pcb permet également d'augmenter la fiabilité, en effet cela limite les faux contacts, les fils qui s'enlèvent et un branchement plus efficace.

Bien évidemment, pouvoir contrôler le robot à distance et de façon exclusive fait partie des points ce projet, mais aussi la possibilité d'avoir un flux vidéo grâce à la caméra embarquée qui peut être contrôlée.

Enfin, la possibilité de contrôler d'autres robots est aussi un point fort même s'il est nécessaire qu'il y ai d'autres robots montés présents. Puis, le stockage des différentes données sur la température récoltées qui sont sauvegardées de façon régulière dans notre base de données et qui nous permet de créer une heatmap.

Points faibles

Il y a tout de même plusieurs points faibles qui peuvent être notés, principalement au niveau de la caméra, par exemple, il est impossible de la flasher depuis notre application, il est nécessaire d'intervenir manuellement (entrez les logs du réseau wifi). La qualité de celle-ci peut également laisser à désirer.

Malheureusement, il est pour l'instant impossible de déployer en ligne notre api, étant donné qu'on utilise l'adresse ip locale de chaque robot, cela pourrait être résolu avec la mise en place de endpoint et ouverture de ports mais cela implique de changer le code qui était impossible pour nous dû à un manque de temps.

Enfin, les batteries doivent être rechargées individuellement, il n'est pas possible de les recharger toutes d'un coup ce qui peut être un point faible.

Partie XII – Problèmes rencontrés

Tout au long du projet, certains éléments ont posé problème et ont dû faire l'objet d'attention particulière :

Lors de la mise en place de la HeatMap, nous nous sommes aperçus que la seule méthode pour nous de produire cette carte est de se rendre sur place afin d'obtenir manuellement la liste des réseaux pour pouvoir la cartographier par la suite.

De même, l'emplacement du robot est déterminé en fonction des adresses MAC qu'il détecte, ainsi, s'il ne détecte qu'une seule adresse IP, son adresse sera la même que celle de l'adresse, s'il en détecte 2, son adresse sera au milieu des deux adresses, ainsi de suite ... C'est pourquoi cette méthode ne fournit qu'une position approximative de l'emplacement du robot et mériterait une amélioration.

En ce qui concerne les servomoteurs, il a été nécessaire de changer le channel PWM afin de pouvoir faire varier la vitesse, autrement, le robot n'aurait pas pu avoir de vitesse variable et l'intensité délivrée au robot n'aurait pas pu être modifiée.

Enfin, en ce qui concerne l'alimentation du robot, celle-ci était constituée de piles mais présentaient une très faible autonomie ainsi qu'une puissance bien en dessous de ce que l'on avait besoin. Il a ainsi fallu passer sur des batteries, avec une autonomie bien supérieure ainsi qu'une puissance accrue.

Partie XIII – Améliorations possibles

En ce qui concerne les améliorations, nous avons pensé à de nombreuses choses qui ne sont pas actuellement implémentées dans le projet par faute de temps/coûts/matériel :

- Possibilité de prendre des photos via l'ESP cam
- Détection des collisions à l'aide d'émetteurs/récepteurs ultrason
- Design du robot (meilleure adhérence des roues, possible avec un filament pour imprimante 3D de type silicone)
- Meilleure technique de localisation
- Déploiement en ligne de l'API
- Mosaïque de toutes les caméras disponibles de l'ensemble des robots
- Mise à jour du firmware lorsqu'un robot se connecte à l'API si nécessaire (à l'aide d'une interface administrateur)
- Utilisation d'un broker privé pour améliorer la sécurité des données
- Sécuriser les transferts de données et la connexion
- Possibilité de voir le niveau des batteries

Partie XIV – Conclusion

Ce projet nous a tout d'abord permis d'en apprendre d'avantages sur la manipulation d'un ou plusieurs ESP, que ce soit au niveau du câblage, de la mise en route, de leur fonctionnement... Ensuite, ce projet nous a permis de développer nos connaissances sur l'utilisation de différents protocoles tels que le protocole TCP puis le protocole MQTT et ainsi nous permettre de mieux comprendre les principes et les différents processus de l'IOT. Par ailleurs, pour certains d'entre nous, ce projet nous a permis de renforcer nos connaissances dans certains langages et framework utilisés côté front et back, React.js et Node.js.

Enfin, pour certains d'entre nous ce fut la première fois que nous devions concevoir un robot et donc utiliser différents logiciels de conception et impression pour imprimer certains composants en 3D, puis comme nous l'avons dit, au niveau du câblage, l'utilisation également de plusieurs matériaux différents... En conclusion, ce fut un projet très enrichissant qui a pu nous apporter de l'expérience dans plusieurs domaines différents.