AVIGNON UNIVERSITÉ

Rapport de projet CMI - Projet IOT

Groupe IOT

MOURGUES Arnaud
GUEGAN Nicolas
RADET Oliver
JAROSSAY Thomas
BLONDEAU Pierre
GRANES Nathan

24 mai 2021

Licence d'Informatique
CMI
UE Projet CMI

Responsables
M. Gozlan
M. Silanus

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

CERI

CENTRE
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE
ceri.univ-avignon.fr

Rapport de projet CMI - Projet IOT

Session 2021

Table des matières

1 Introduction

Dans le cadre de l'UE CMI Interface du réel au numérique, Nous devions choisir un projet de groupe à réaliser : Le premier est un projet qui consistait à faire danser de façon synchrone 3 robots à base roulante, et le second projet consistait à acquérir une grandeur physique avec des appareils et les restituer dans une application sur un périphérique distant.

2 Solutions

2.1 Solutions existantes

La base du projet consiste à utiliser des capteurs de grandeur physique sur une arduino, puis d'envoyer les données à un autre appareil qui devra afficher les données.

Plusieurs questions se sont posées:

- Quelles grandeurs allaient-on capter?
- Combien d'appareils allaient-on utiliser?
- Comment récupérer les données des capteurs sur un périphérique?

Pour les grandeurs, nous avions plusieurs possibilités, la température, le niveau sonore, la luminosité, l'humidité, la pression …

Un faible nombre d'appareils pourrait causer un manque de précision car certaines grandeurs physiques peuvent beaucoup varier en fonction de l'endroit où nous les captons (par exemple, la température devant une fenêtre). D'un autre côté, avoir beaucoup de capteur pourrait demander beaucoup plus de ressources car il y aurait beaucoup de données à traiter.

Enfin, pour récupérer les données sur un périphérique distant, on pouvait faire en sorte qu'on puisse se connecter à une interface graphique pour chaque appareil où l'on verrait les données captées par l'appareil en question.

2.2 Solution développée

La solution retenue est de placer 4 appareils dans la salle à différents endroits de la pièce afin de récupérer des données plus précises, 3 appareils envoient les données de leurs capteurs à un appareil principal qui fera office de serveur.

Pour voir les données, on peut se connecter à l'appareil principal via IOT Controller.

Les grandeurs physiques mesurées sont la température, le son et l'humidité car ce sont des indicateurs nécessaires au bon fonctionnement d'un établissement.

De plus, chaque appareil possède un écran LCD avec les valeurs captées affichées dessus.

3 Organisation

3.1 L'équipe de projet

JAROSSAY Thomas - Licence 2 - Informatique

RADET Olivier - Licence 1 - Informatique

GUEGAN Nicolas - Licence 1 - Informatique

BLONDEAU Pierre - Licence 1 - Informatique

GRANES Nathan - Licence 1 - Informatique

MOURGUES Arnaud - Licence 1 - Informatique

Associations :

1. Pierre et Nathan
2. Nicolas et Arnaud
3. Thomas et Olivier

nous avons majoritairement communiqué via discord, de plus nous avons mis en place un GitHub (https://github.com/notoverflow/IoT-CMI/tree/main)

3.2 Répartition des tâches

- Gestion de la communication entre les appareils
  - Configuration et utilisation de la carte client : **BLONDEAU Pierre**
  - Configuration et utilisation de la carte point d'accès : **GRANES Nathan**
- Gestion des capteurs et de l'affichage des données sur l'écran LCD
  - Production du code pour l'étalonnage des capteurs sonores grâce à un sonomètre : **JAROSSAY Thomas**
- modélisation des appareils et gestion de l'alimentation
  - Choix du mode d'alimentation adapté à notre consommation : **GUEGAN Nicolas**

- Modélisation des composants du système en 3D : **MOURGUES Arnaud GUEGAN Nicolas**
- Impression 3D du support : **MOURGUES Arnaud GUEGAN Nicolas**

3.3 Environnement matériel/logiciel

Matériel :

Nous avons travaillé sur des Seeeduino Cloud avec:

- un capteur d'humidité et de température DTH 22 Groove
- un capteur son Grove 101020063
- écran lcd RGB Grove backlight
- powerBank

Nous avons utilisé une imprimante 3 Dimensions :

- Imprimante 3 Dimensions xyz pro Davinci

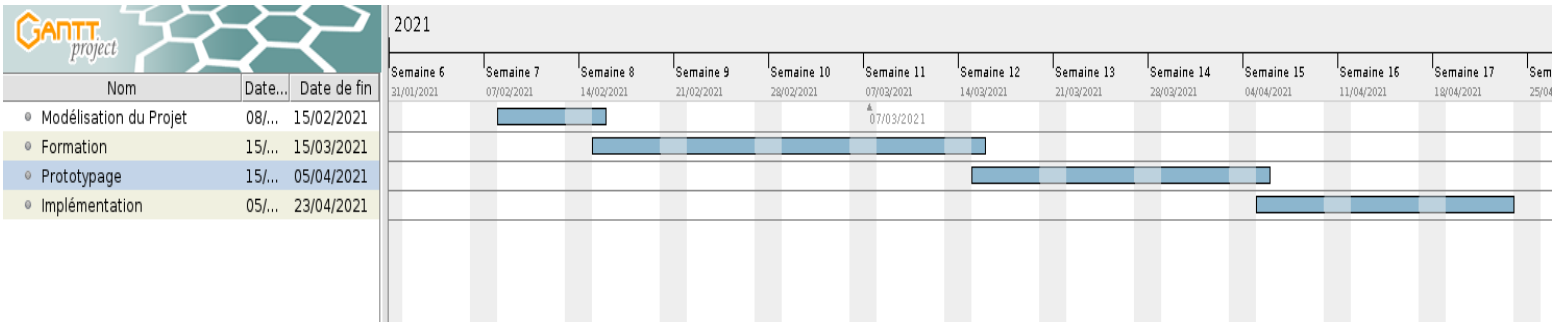
Logiciel :

Nous avons utilisé le panneau de contrôle internet des cartes seeeduino pour la configuration de ses dernières en points d'accès et en client.

Nous avons travaillé avec l'IDE arduino pour coder, l'outil le plus optimisé pour les cartes arduino.

Nous avons également utilisé Fusion 360 pour la modélisation en 3 Dimensions.

3.4 Planification



4 Réalisation

4.1 Travail d’équipe

Le travail en équipe s'est scindé en 3 groupes de 2. Avec pour objectif de regrouper tous nos travaux à la fin, pour répondre au problème posé.

Nous avons abouti au résultat suivant : une carte (client) comportant 2 capteurs, capable d'envoyer les relevés (température, son, humidité), à une autre carte (point d’Accès), par signal WiFi.

Le fonctionnement général se décrit de cette manière :

1. Le point d’Accès alimenté, il émet un signal wifi
2. Attente d’une connexion de la part d’un client
3. Après cela, on alimente le client, il se connecte automatiquement au point d'Accès
4. Le client récupérer les données des capteurs
5. Envoie des données du client au point d’Accès

METTRE SCREENS DE L'ENVOIE DE DONNÉES (CLÉ USB NATHAN)

Partie etudiant 1

Olivier Radet

1- objectif de la partie personnelle.

J’ai été chargé de récupérer à l’aide de capteurs de mon choix, la température, l’humidité et le volume sonore, dans le but d’analyser la conformité des paramètres d’ambiance avec des normes prédéfinies.

Mon travail a été divisé en plusieurs phases:

- phase 1: s’informer au niveaux des normes
- phase 2: choix des capteurs appropriés
- phase 3 :test avec des capteurs et matériel déjà disponibles dans le labo

- phase 4:création d’un programme de surveillance des paramètres d’ambiance

- phase 5: adaptation du programme avec les capteurs reçus

2- Développement du projet:

Phase 1: après recherche sur internet, les différentes normes sont très nombreuses et dépendent du lieu et de l'utilisation de cet espace. Par exemple, pour une salle de cours la température doit rester entre 20 et 23°C, le taux d’humidité entre 40 et 60 % et la pression acoustique entre 35 et 65 dB. J’ai donc choisi cet exemple pour calibrer notre capteur de surveillance d’ambiance.

(par exemple, dans une boite de nuit, la norme sonore est d’environ, 105dB).

Phase 2: Pour cela je devais sélectionner les capteurs qui me semblaient les plus appropriés en fonction des normes et de nos besoins.

a- capteur humidité:J’avais choisi initialement sur un capteur DHT11 (pour l’humidité et la température) car il était déjà disponible dans le laboratoire. Ce capteur manquant de précision, j’ai donc opté pour une meilleure précision avec le DHT22 (il peut mesurer l’humidité de 0 à 100% avec une précision de 2% tandis que le dht11 mesurait de 20 à 80% avec une précision de 5%)- taille identique, prix similaires (prix du DHT11:6.5€ et prix du dht22:10.9€). Une autre grande différence de ces 2 capteurs concernait la température; le DHT11 pouvant mesurer de -2°C à 60°C avec une précision de 2%, le DHT22 peut, quant à lui, mesuré de -40°C à 80°C avec une précision de 0.5%.

Ces différences de caractéristiques entre ces 2 capteurs permettront d’utiliser notre capteur de surveillance des paramètres d’ambiance dans des conditions plus extrêmes et plus précises (basse et haute température et forte humidité).

b- capteur sonore:

Je devais choisir un capteur sonore. L’équipe ayant choisi de développer le projet sur la base d’une carte seeeduino, la principale contrainte du choix du capteur, était alors, qu’il réponde à la norme grove. Cela permettait de brancher tous les capteurs, ainsi que l’écran, facilement, sur un shild grove.

J’ai choisi le capteur loudness sensor dont ses grandes spécifications sont sa fréquence de 5 à 20 000hz et sa grande sensibilité -60 à -56dBV/Pa. (coût 6.5€).

c- J'utilise un écran LCD RGB grove disponible dans le laboratoire, pour afficher mes relevés en temps réels. La fonction rgb de l’écran est trés pratique pour changer de couleur (passage du vert au rouge) lorsque l’un des paramètres d’ambiance sort de sa plage prédéfinie.

Phase 3:

Ces tests m’ont permis de valider la faisabilité du projet: j’arrivai à récupérer la température et l'humidité et à afficher ses données sur l’écran . L’absence de capteur permettant d’obtenir la pression acoustique rendra la fin du projet difficile car l'acquisition de la pression sonore se révélera plus difficile.

Cette phase de test m’a permi de faire face à plusieurs difficultés, notamment des problèmes de conflit.Par exemple, en branchant l’écran sur un des ports I2C ainsi que les capteurs de température et d'humidité sur le D2, je n’arrivais pas à récupérer les données de température et d’humidité car les ports I2C utilisent une partie du port D2.

Ainsi, j’ai branché le capteur DHT11 sur le port D6 (un des autres ports disponibles) pour résoudre mon problème.

Phase 4: J’ai utilisé les librairies du capteur et de l’écran pour pouvoir afficher les informations récupérées.

J’ai fait un programme en C++. Pour cela, j’ai utilisé le logiciel arduino qui permet de gérer les bibliothèques, faire des exemples, compiler, téléverser, ainsi qu’accéder au terminal de la carte.

Phase 5: Aprés réception des capteurs choisis, j’ai adapté mon programme aux nouveaux capteurs d’humidité et de température (changement du DHT 11 en DHT22).

Pour le capteur sonore, il se branche sur un port analogique et j’arrive à récupérer son signal analogique compris entre 0 et 1023. J’ai dû ensuite le calibrer à l’aide d’un sonomètre et de audacity pour jouer différentes tonalités. J’ai noté les décibels que relevaient le sonomètre ainsi que la valeur analogique de mon capteur et j’ai rentré toutes ses associations dans un tableur. J’ai donc obtenu, une formule de calibrage.

Cependant, je n’obtenais toujours qu’une valeur de décibel fixe (même si la valeur analogique a changé). Je n’ai donc toujours pas réussi à récupérer le volume sonore ambiant.

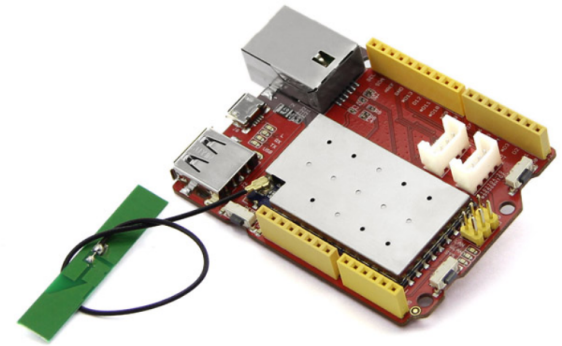
La réception tardive du capteur, et un manque de temps, ne m’ont pas permis de finaliser ce projet.

Partie de Nicolas GUEGAN

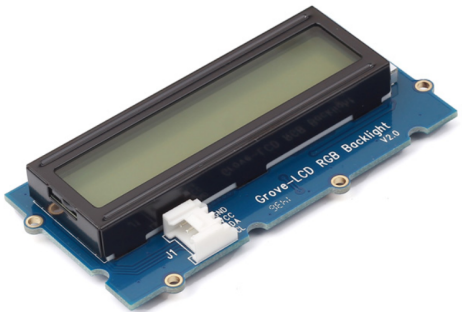
Durant la période pendant laquelle nous avons eu cours d'Interface du réel au numérique j'ai eu plusieurs tâches à remplir.

La première que nous avons décidé de m'attribuer est la gestion de l'alimentation.
Pour ce faire, il fallait d'abord lister l'intégralité des composants de notre projet.
La liste est la suivante :

- Carte seeeduino Cloud
- Ecran LCD RGB Backlight
- Capteur sonore Grove 101020063
- Capteur température/humidité DHT 22



Carte seeeduino Cloud



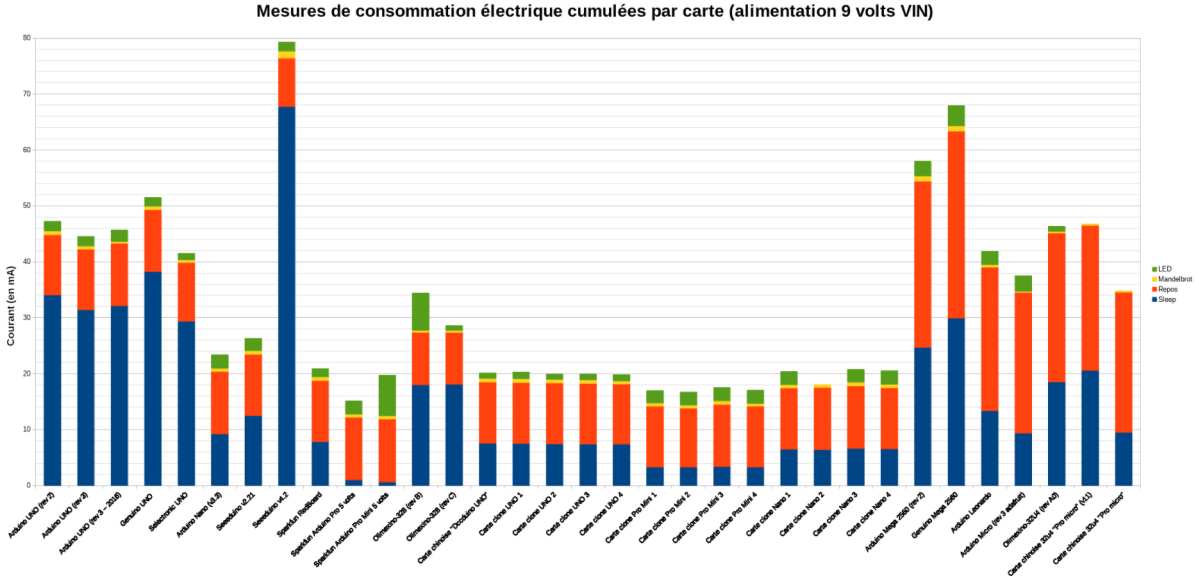
Ecran LCD RGB Backlight

Capteur sonore

Capteur température/humidité DHT22

Ensuite il fallait déterminer la consommation de chaque composant, celle-ci se mesure en milliAmpère par heure (noté : mA/h).
N'ayant pas trouvé de documentation précise sur la consommation de la carte seeeduino Cloud les données concernant cette carte sont approximatives.

Après plusieurs recherches, d'après le site " https://www.carnetdumaker.net/images/graphique-de-consommation-electrique-avec-alimentation-9-volts-externe/ ", un graphique sur la consommation de différentes cartes Arduino/Seeeduino est réalisé.



J’ai choisie la consommation de la carte v2.21 pour faire un parallèle le plus réaliste possible à la consommation de la Seeeduino Cloud.

Sachant que sa consommation est d’environ 26 mA/h et que la réelle caractéristique différente de la Cloud est d’avoir une connexion Wi-Fi disponible, nous pouvons en déduire que leurs consommations sont relativement similaires.

Ensuite concernant l'écran, le capteur sonore et le capteur température/humidité, leur consommation est de respectivement : 60 mA/h, 5 mA/h et 5.7 mA/h.

Avec ces données la consommation serait de 97.5 mA/h (environ 100 mA/h).
De plus, les broches d’entrées et de sorties sont aux nombres de 20 et peuvent délivrer 40 mA/h maximum chacune, soit une consommation de 800 mA/h.
Nous sommes donc bien en dessous de la capacité maximale de la carte.

Après plusieurs recherches, on m’apprend que des power bank sont disponibles dans la salle (fournies par l’établissement).
Elles délivrent au maximum 5200 mA, on en déduit aisément que ces powerbank suffisent largement car nous pouvons en théorie faire fonctionner notre système durant 52 heures.

Ensuite je me suis renseigné sur la législation concernant les seuil à ne pas dépasser que ce soit pour le son, la température ou l’humidité.
Concernant la température, selon le site “ https://www.picbleu.fr/page/est-interdit-chauffer-logement-plus-19#:~:text=lecture%20%3A%208%20minutes-,Il%20est%20interdit%20de%20chauffer%20un%20logement%20%C3%A0%20plus%20de,enseignement%2C%20les%20bureaux%20et%20ERP. “

qui fait appel aux articles R.241-25 à R.241-29 du code de l’énergie ainsi que de l’article R. 131-20 du code de la construction et de l'habitation, la température idéale serait de 19° celsius.
Pour l’humidité, selon le site “ https://www.editions-legislatives.fr/actualite/qualite-de-l-air-interieur-enjeux-et-perspectives “ le taux d’humidité serait optimal entre 20% et 40% à une température de 20° celsius.

Enfin le niveau sonore, n’ayant pas trouvé de réels textes sur lesquels m’appuyer, je répondrais qu’il serait d’environ 65 dB soit l’équivalent d’une salle de classe.

Ayant terminé ma tâche concernant l’alimentation j’ai décidé de m’orienter vers le codage. Celui-ci permet de récupérer les valeurs recueillies par les capteurs puis de les afficher sur l’écran lcd.
J’ai donc commencé à apprendre avec l’IDE Arduino, le même que celui qu’Olivier Radet utilisait, car c’est à lui que cette tâche a été assignée.
Cependant je me suis vite rendu compte de mon inefficacité dans le groupe car le temps que j’apprenne à maîtriser ce code et que j’apporte mon aide à Olivier, cela aurait été sans intérêt.
Voici des extraits du code arduino d’Olivier Radet pour montrer en détail le fonctionnement de l’IDE Arduino.

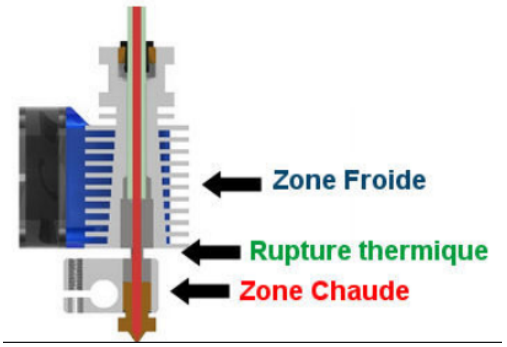
```
lcd.setCursor(0, 1);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));

lcd.print(t);
lcd.print("C / ");
lcd.print(h);
lcd.print("%");

if (t < 20 || t > 22 || h > 60 || h < 40) {

    colorR = 255;
    colorG = 0;
    colorB = 0;
      Serial.println("hors zone");

} else {
    colorR = 0;
    colorG = 255;
    colorB = 0;
      Serial.println("bonne zone");

}
```

Ici ce bout de code a pour but de changer la couleur de l'affichage selon les données recueillies par les capteurs.
Si elles dépassent des valeurs limites alors l'affichage est rouge par exemple.

J'ai donc décidé de rejoindre Arnaud Mourgues dans sa tâche afin de me rendre plus utile pour le groupe.
La tâche d'Arnaud était de réaliser un modèle 3 Dimensions d'un support capable d'accueillir le système avec la carte, les capteurs et la powerbank et de l'imprimer en 3 Dimensions.
L'imprimante 3 Dimensions utilisée est la xyz PRO da vinci 1.0 :

Lorsque je l'ai rejoint, il avait déjà modélisé une bonne partie du système.

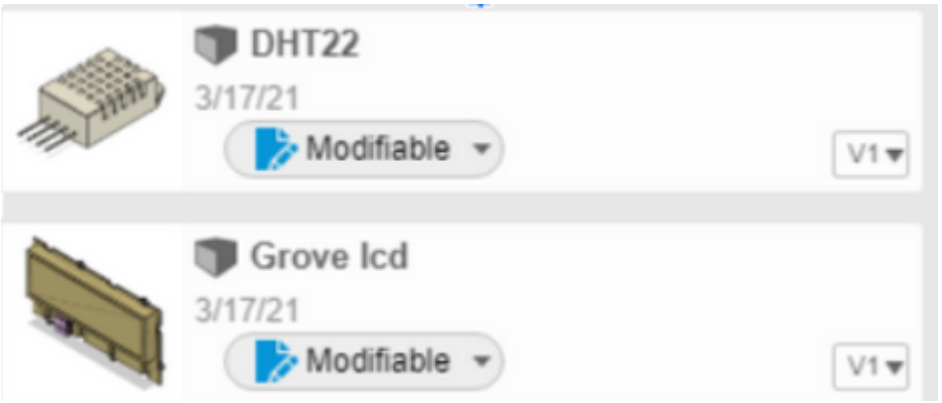
Nous avons également tenté de faire marcher l'imprimante 3 Dimensions qui était mise à notre disposition.
Pour ce faire nous avons essayé d'imprimer des modèles simples ainsi que des modèles pré-enregistrés dans l'imprimante, afin de voir comment cela fonctionnait, malheureusement le fil une fois coulé, n'adhérait pas au socle, ce qui ne permettait pas de réussir une bonne impression.
Aussi un message d'erreur s'affichait disant que le matériel n'était pas pris en charge.
Nous avons donc cherché sur des forums et sur youtube les différentes raisons possibles.
La première chose que nous ayons faite a été de nettoyer la buse.



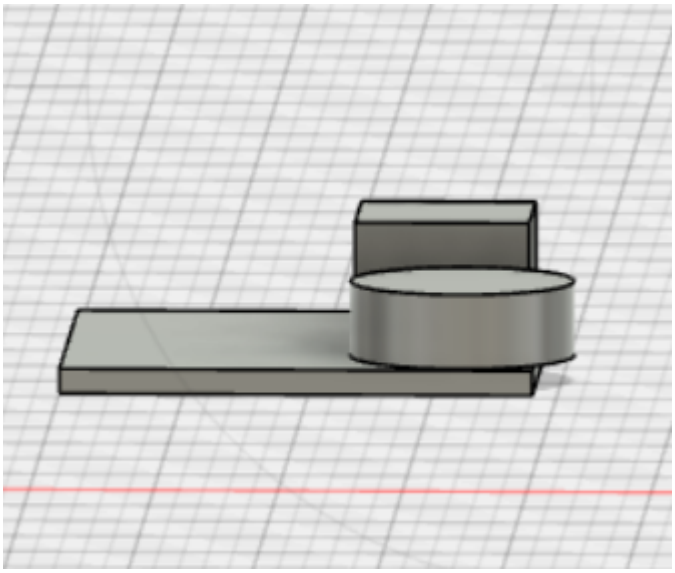
La buse est l'élément le plus bas sur la photo, c'est par là que sort le fil.

Cette action n'ayant pas eu l'effet escompté, nous avons essayé de modifier la température du socle ainsi que la hauteur d'impression suite aux conseils de notre Professeur Mr.Gozlan.
Ces modifications n'ont pas changé le résultat final.
Finalement nous avons changé la bobine de fil utilisée par l'imprimante, mais cela n'a à nouveau rien changé.

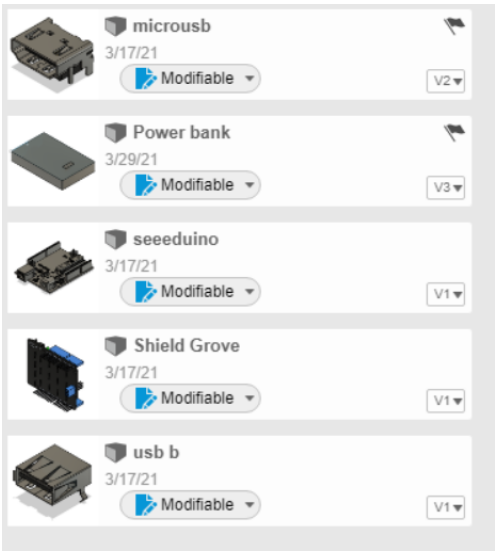
De mon côté j'ai récupéré les modèles 3 Dimensions du capteur DHT22 ainsi que de l'écran LCD sur internet.

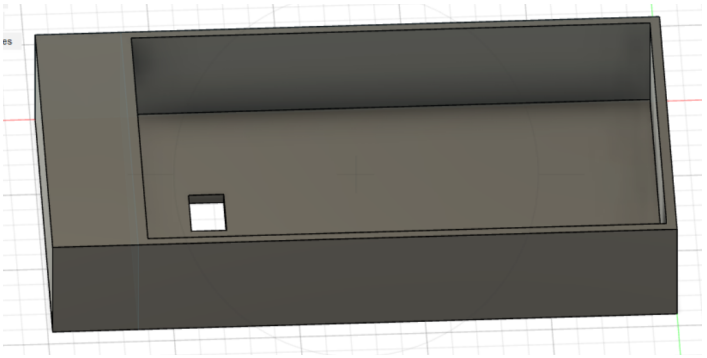


J’ai également commencé à modéliser la capteur sonore mais par manque de temps je n’ai pu le terminer :



Les travaux cumulés d’Arnaud et moi-même sont visibles ci-dessous :





Partie MOURGUES Arnaud

La partie qui m'a été attribuée est celle de modéliser le support pour le système IOT sur un logiciel de modélisation 3D.

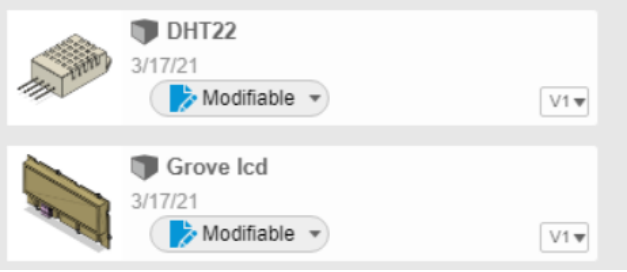
J'ai commencé à modéliser mes 2 premières pièces sur Tinkercad, mais l'outil ne m'était pas familier. J'ai alors essayé Fusion 360 qui propose un nombre de fonctionnalités plus élargis et qui m'a paru plus ergonomique. Mon choix s'est alors naturellement porté sur ce second logiciel.

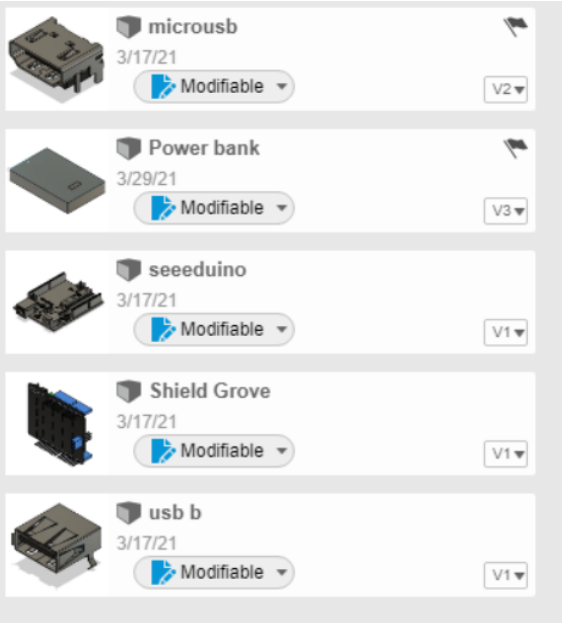
AUTODESK®
FUSION 360™

Ce premier petit travail m'a permis de me familiariser avec les fonctionnalités du logiciel. Après quoi, je me suis lancé dans la confection ou la récupération des composants du projet.

J'ai récupéré une partie des composants en fichier 3D sur internet. J'ai également modélisé la batterie portable, car inexistante. Voici une liste exhaustive des composants récupérés :

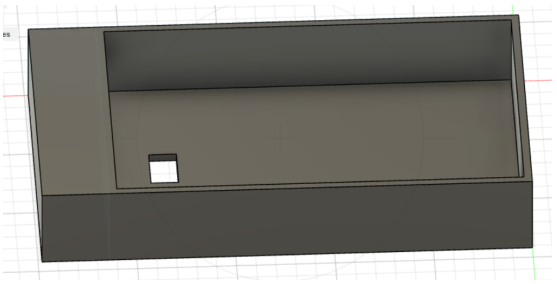
Partie de Nicolas :



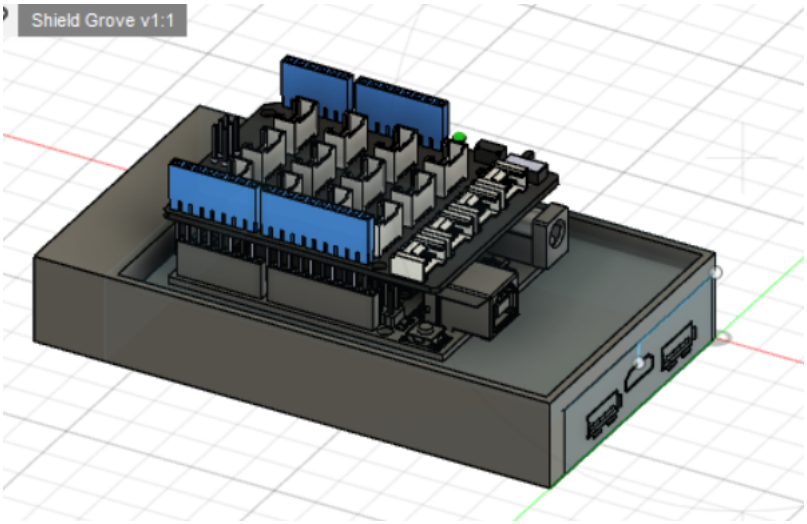


Il nous manque seulement le capteur sonore grove que nous n'avons pas trouvé sur Internet, et que Nicolas a commencé à modéliser. Aussi, il nous manque les fils de connexion entre les composants et la seeeduino, et les fils USB pour l'alimentation de la seeeduino. Je n'ai pas réussi à modéliser les fils, trop complexe pour le niveau que j'ai acquis depuis la première séance sur Fusion.

J'ai également commencé à dessiner le support ci-dessous :



Cependant, j'ai eu quelques problèmes à différents niveaux. D'abord, l'intégration des pièces au support. Après avoir récupéré les pièces dont j'ai parlé peu avant, il a fallu les mettre dans le support, les superposer ou les assembler pour obtenir un rendu réel, et aussi savoir exactement où placer les trous pour les capteurs dans le support. Mais il est trop difficile de placer exactement les pièces car je n'ai pas d'idée précise de la forme finale que doit avoir le support. Ce-dernier intègre pour le moment seulement la batterie externe qui se place dans le support, et le petit compartiment sur la gauche devait servir à stocker les fils dans un endroit sans qu'ils ne soient visibles. Nous avions imaginé que la seeeduino pourrait se trouver au-dessus ainsi que le shield, et que les composants se trouvent sur un dôme que nous n'avons pas pu modéliser.



Nous avons passé également du temps à faire marcher l'imprimante 3D que Mr Gozlan nous a confié. Nous avons eu de nombreux problèmes que Nicolas a évoqué précédemment, et nous n'avons pas trouvé de solution sur des forums ou sur le site du constructeur, du fait de la faible communauté autour de ce type d'imprimante.

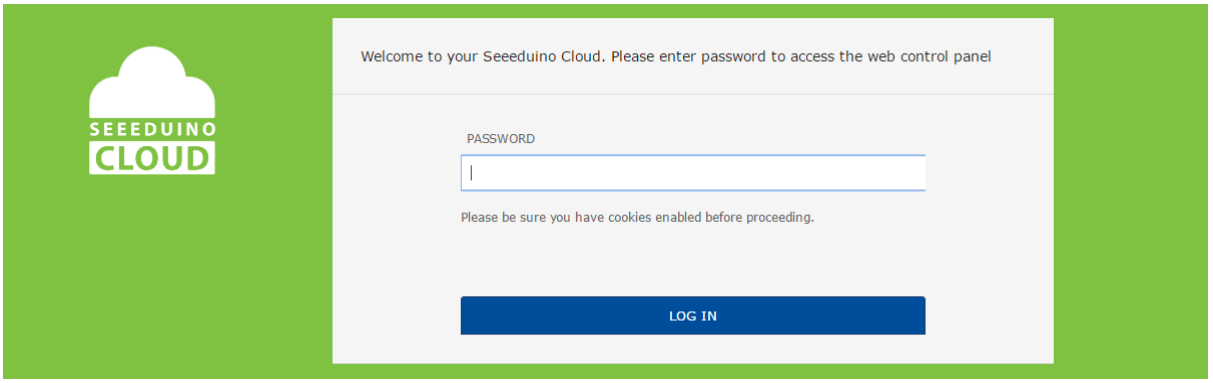
Partie BLONDEAU Pierre

Le travail que j'ai mené consistait à configurer la carte client afin qu'elle puisse envoyer des informations par wifi au point d'Accès.

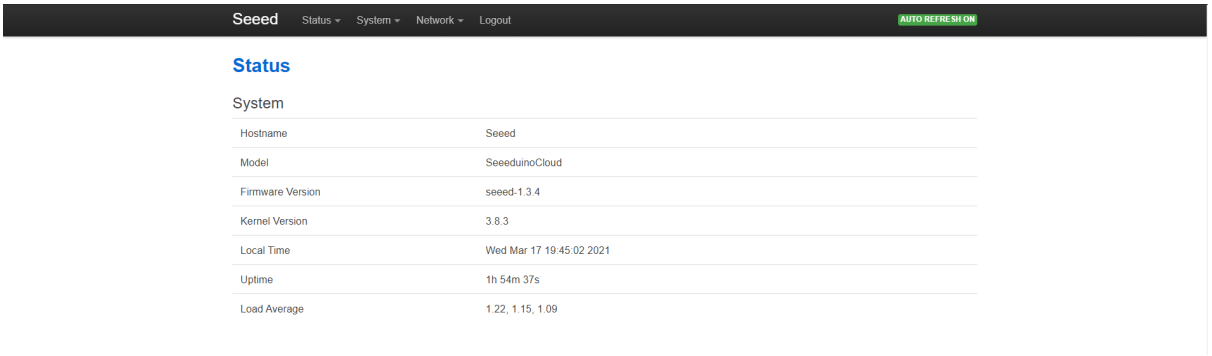
Pour commencer, il faut savoir que chaque carte est au départ configurée en tant que point d'Accès. C'est pour cela qu'elle émet du wifi lorsqu'on l'alimente. Notre 1er objectif est donc de la mettre en mode client et de la connecter au point d'Accès. Pour cela il faut au préalable que le point d'Accès soit alimenté et configuré.

Pour se faire :

1. Se connecter au réseau wifi de la carte, nommé de base SeeeduinoCloud-AXXXX, avec un téléphone ou un ordinateur portable par exemple.
2. Une fois connecté, dans la barre de recherche de votre navigateur internet, entrez une des adresses ip suivantes : **192.168.240.1** ou **172.31.255.254**, ce sont les 2 adresses ip par défaut des cartes seeeduino cloud.
3. On arrive sur la page suivante, le mot de passe par défaut est **"seeeduino"**, appuyer ensuite sur log in.



4. Une fois connecté, allez dans "SYSTEM", puis en haut, cliquer sur **see the "advanced configuration panel(luci)"**, pour arriver sur la page suivante :



5. Dans les menus situés en haut de la page, il faut faire dérouler **"Network"**, puis sélectionner **"Wifi"**. On se retrouve dans le **"Wireless Overview"**.

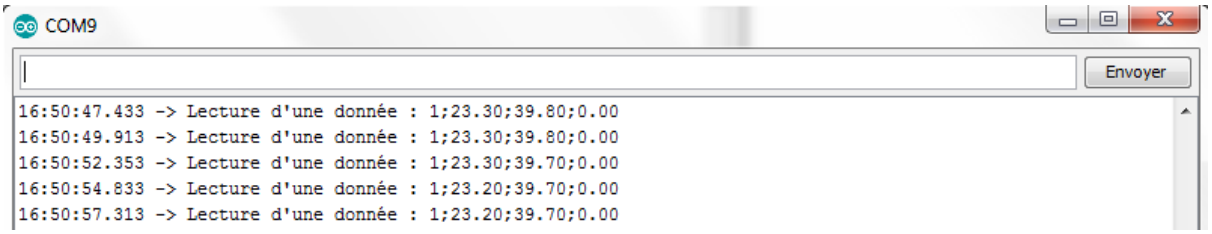
6. Appuyer sur **“Scan”**
7. Chercher le nom de l’Accès point, puis faire **“join network”**
8. Ne rien changer, cliquer sur **submit**, puis sur la page suivante, le Mode afficher devrait être“Client”, faire **Save & Apply**.

Après avoir effectué cette manipulation, notre carte est passée de point d’accès, à client. Elle est connectée en wifi au point d’accès. Mais étant donné que le client n’émet plus de signal wifi, nous avons perdu la connexion à ce dernier, et donc au Web Control Panel.

Pour s’y reconnecter, il faut se connecter au réseau wifi du point d’Accès, aller dans le dhcp (voir partie sur comment configurer de point d’Accès, le chemin y est détaillé), où nous retrouverons l’adresse IP de la carte client. En rentrant cette adresse dans le moteur de recherche, tout en restant connecté au point d’accès, pour rester sur le même réseau que notre client. Nous pouvons accéder de nouveau à son Web Control Panel.

Un des problème survenu, est que nous devons faire cette manipulation à chaque fois que l’on débranche une des deux cartes. En effet, elles ne se reconnecterons pas automatiquement. Nous allons y remédier dans le code que nous allons injecter dans la client, grâce à l’IDE Arduino.

Avant cela, il faut pouvoir envoyer des données au point d’Accès, pour ce faire, nous allons aussi passer par l’IDE Arduino. Voici ce que le point d’Accès recevra :



Le code injecté dans le client est le suivant :

```
#include <BridgeClient.h> //inclusion des librairies
#include <Process.h>
BridgeClient client;
IPAddress server(192, 168, 240, 13); //Adresse IP du point d'accès
Process p;
void setup() {
  Bridge.begin();

  p.runShellCommand("uci set arduino.@arduino[0].access_point_wifi_name='SeeeduinoCloud-AP'"); // Connexion automatique au point d'Accès
}

void loop() {

    float temperature = 25.3; // Déclaration de variables et données test
    int humidite = 56;
    int db = 15;
    int id = 1;


 if (client.connect(server, 5555)) { //Si le client est connecté au point d'Accès : on lui envoie les données
    client.print(id);
    client.print(";");
    client.print(temperature);
    client.print(";");
    client.print(humidite);
    client.print(";");
    client.println(db);

    Serial.println("envoie donnée");

  }
  client.stop();
  delay(2000);
}
```

La ligne : **“p.runShellCommand("uci set arduino.@arduino[0].access_point_wifi_name='SeeeduinoCloud-AP'");”** permet la connexion automatique au point d’Accès.

C’est ce qui résout le problème cité plus haut. Cela exécute une commande directement dans le noyau linux, connectant le client au point d’Accès.

Maintenant que l’on a tout cela, il faut juste faire attention que le point d'accès soit alimenté et prêt à l’emploi avant d’injecter le code client pour la 1ere fois. Une fois le client connecté, même si l’on débranche et rebranche le point d’Accès, il s’y connectera automatiquement.

Après cela, il a fallu mettre en commun la récupération des données par les capteurs, et le programme d’envoie. Nous avons donc pu envoyer les données récupérées par wifi :

Partie Nathan GRANES

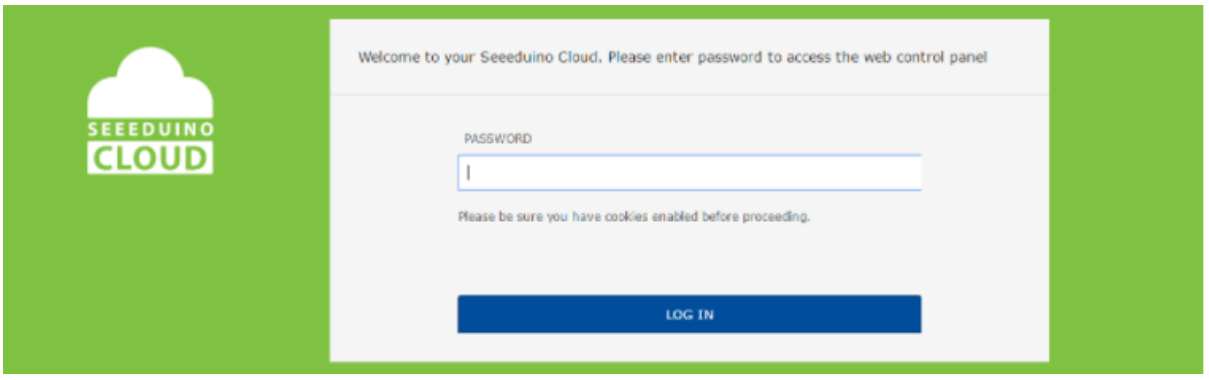
La tâche que l'on m'avait initialement assigné était "*Gérer l'interface utilisateur avec IOT controller et Proteus*". Cependant afin de gérer cette interface, la carte désignée comme point d'accès doit être configurée afin de pouvoir envoyer les données à l'interface. C'est pour cette raison que j'ai décidé de m'occuper de la configuration de la carte point d'accès qui s'occupe de recevoir les données.

Le travail que j'ai mené consistait donc à configurer la carte Access Point afin qu'elle puisse recevoir des informations envoyées par le carte Client en WIFI. Je vais donc expliquer étape par étape comment j'ai réussi à configurer la carte.

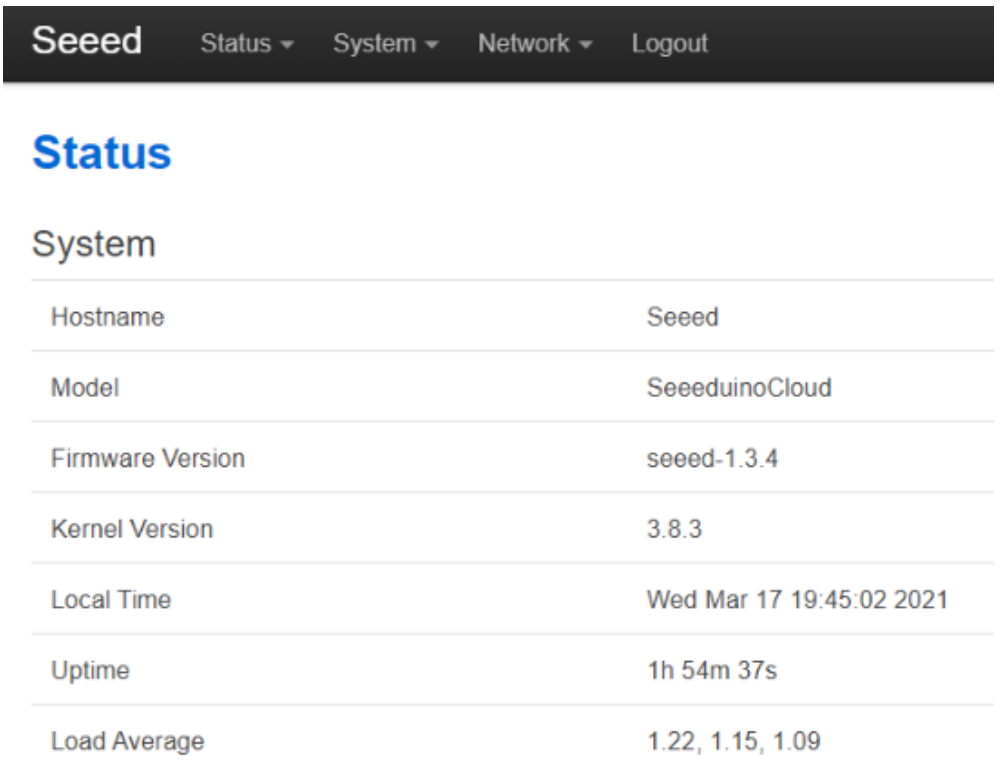
1. Se connecter au réseau wifi de la carte, nommé de base SeeeduinoCloud-AXXXX, avec un appareil pouvant naviguer sur internet (ordinateur / téléphone portable).

2. Une fois connecté au réseau, taper dans la barre d'URL d'un navigateur l'adresse IP local par défaut de la carte Seeduino Cloud :**192.168.240.1** ou **172.31.255.254** .

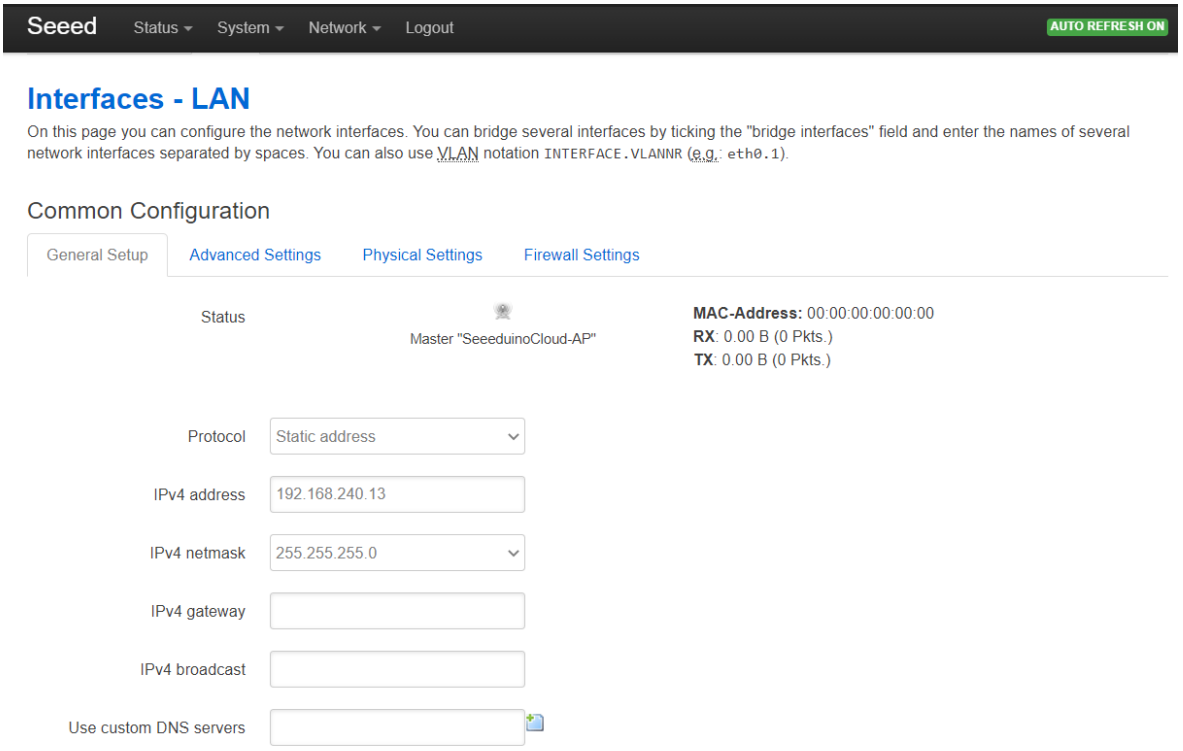
3. Si nous sommes bien connecté au réseau wifi de la carte Seeeduino Cloud, nous arrivons normalement sur la page de connexion où un mot de passe est demandé, le mot de passe par défaut est **"seeeduino"**, appuyer ensuite sur log in.



4. Un fois connecté il faut se diriger dans le menu "SYSTEM" puis cliquer sur ***"see the advanced configuration panel(luci)".*** Ceci nous permettra d'accéder au panneau de configuration avancé de la carte. (voir ci-dessous)



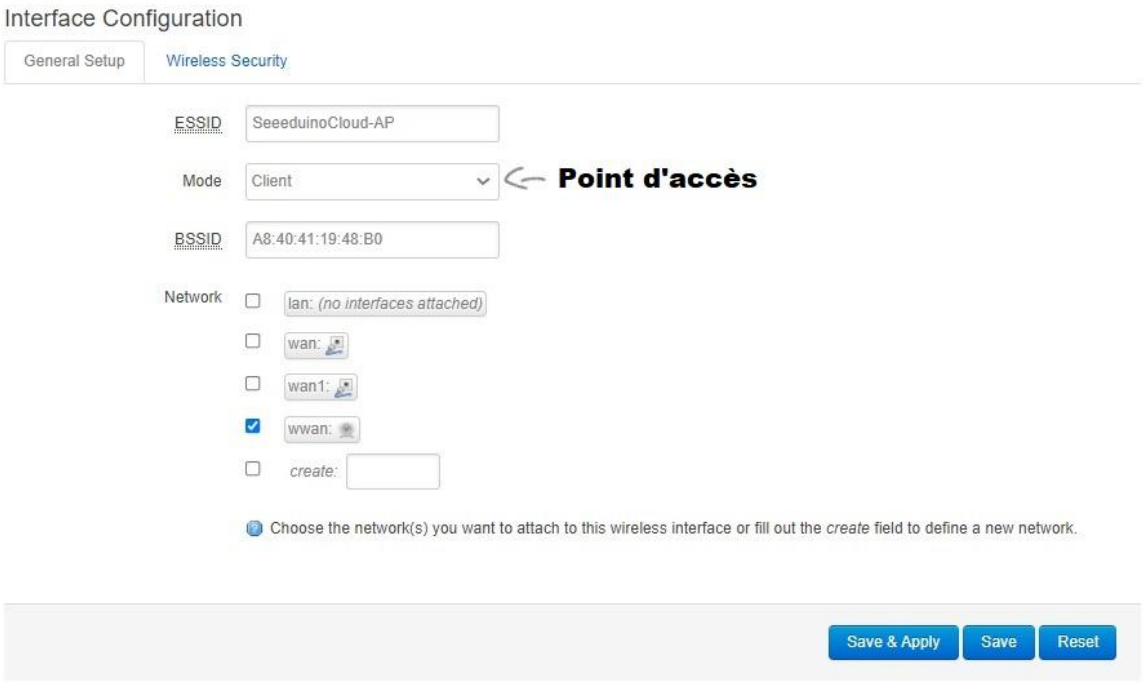
5. D'ici, la première chose à faire est de changer l'adresse IP de la carte en adresse IP fixe. Pour cela, je me suis rendu dans ***Network*** → ***Interfaces*** puis, cliquer sur ***Edit*** du réseau LAN. Nous voici sur cette page:



Choisir le protocoles ***Static address*** puis changer le dernier chiffre de l'adresse IPv4 (pour ma part j'ai choisi 13). Ainsi en validant et redémarrant la carte, son adresse

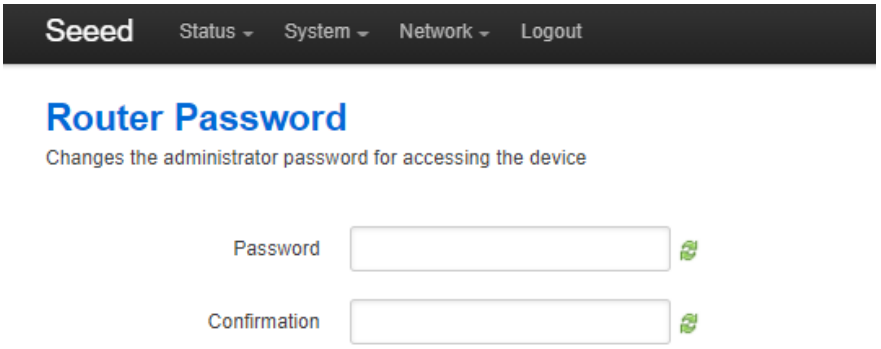
sera **192.168.240.13**. *Cette étape est très importante pour la connexion du client au point d'accès !*

6. Ensuite j'ai décidé de changer l'ESSID (Nom du réseau de la carte) afin qu'on identifie plus simplement quel est le réseau du point d'accès. Pour cela se rendre dans ***System→ Interface configuration :***



Changer l'ESSID avec un nom identifiable qui vous arrange et mettre en mode ***Access Point*** si ce n'est pas déjà fait.

7. Cette étape est purement optionnelle, elle permet de changer le mot de passe pour se connecter au panneau de configuration de la carte. Pour cela je me suis rendu dans ***System →Administration→Router→Password :***



Vous n'avez plus qu'à choisir un nouveau mot de passe.

8. Désormais la carte Point d'accès est configurée. (voir la partie de Pierre BLONDEAU pour configurer une carte client et la connecter au Point d'accès)

Une fois un Client connecté

4.8 Intégration

Nous avons combiné le code permettant la connexion entre deux Seeeduino Cloud et le code permettant de récupérer les données captées. Il nous a suffi d'incorporer le code de la carte client à celui de la carte recevant les données.

Le code de l'envoie de données à été placé à la fin de celui récoltant les informations, en créant une nouvelle fonction. Il nous a suffit de l'appeler en fin de récupération de données des capteurs, avec en paramètre les 3 valeurs que l'on souhaite envoyer.

```
//Code récupération données ...........
envoieWifi(1,temperature,humidite,db);


void envoieWifi(int id, float temperature, float humidite, int db){

  if (client.connect(server, 5555)) {
    client.print(id);
    client.print(";");
    client.print(temperature);
    client.print(";");
    client.print(humidite);
    client.print(";");
    client.println(db);

    Serial.println("envoie donnée");

  }
  client.stop();
  delay(2000);
}
```

Ainsi, nous pouvons récupérer les informations des capteurs et les envoyer par wifi au point d'Accès.

5 Conclusion

La conclusion doit dresser le bilan et ouvrir vers des perspectives d'évolution du projet : développements, améliorations possibles de votre travail.

Elle vous permet également d'expliquer ce que ce projet et le travail en équipe vous ont apporté (connaissances, découvertes, expériences,…).

BILAN?..... + PRESPECTIVES : faire en sorte qu'on puisse connecter plusieures cartes, ajouter donnée que l'on récupère ?, affichage intéressant avec moyenne des données etc -> nous informe de si on est bien ou pas par rapport au résultat optimal que l'on pourrait avoir.

Ce travail nous a permis de découvrir et d'en apprendre davantage sur les nombreux outils et logiciels utilisés. Notamment les cartes seeeduino, l'IDE arduino, les capteurs sonores, de température ainsi que d'humidité. Nous avons pu au cours de ces séances, élaborer un travail d'équipe, sans quoi nous n'aurions pas pu arriver jusqu'ici. DEVELOOPER PLUS JPENSE