RAPPORT DE PROJET SAÉ LORAWAN

2023/2024

Robin Augier
et
Alexandre Pichot

Table des matières

1. Présentation du Projet

Introduction

Actuellement, la moitié de la population mondiale réside en milieu urbain, et les prévisions pour 2050 est de plus de 70 %. Par conséquent, la gestion des villes pose aujourd'hui des challenges importants comme la population croissante, les embouteillages, la gestion des ressources énergétiques, réchauffement climatique, pollution, ect…

Donc on essaie de trouver des solutions pour nos problèmes environnementaux par exemple, la création de villes intelligentes qui nécessitent l'intégration de services intelligents tels que la Mobilité Intelligente, l'Environnement Intelligent, le Mode de Vie Intelligent, etc.

Donc pour notre SAE de semestre 3 en BUT2 GEII nous allons travailler sur la mobilité intelligente, et créer une location de vélos connectés qui pourra nous envoyés des informations clefs grâce à ses capteurs comme la température, l'humidité, sa localisation, sa direction, sa luminosité.

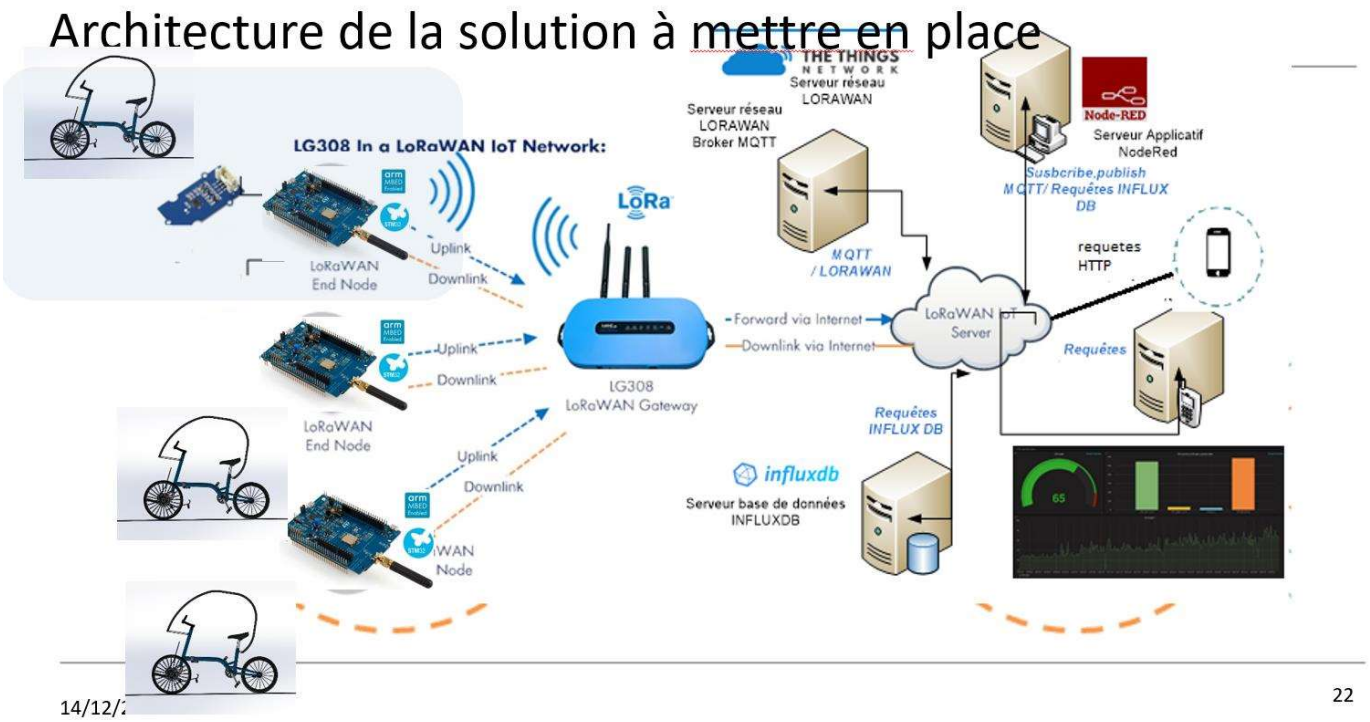
L'idée est d'embarquer sur le tricycle électrique une carte mbed, développer sur celle-ci les fonctions nécessaires pour rendre le tricycle « communicant » avec le monde extérieur et pouvoir le géolocaliser.

Nous avons donc listé nos objectifs qui sont :

- Création et réalisation d'une carte métier à l'aide de DesignSpark.
- Création et réalisation de l'antenne à l'aide de ADS
- Prise en main et réception du flux de données de The Thing Networks
- Prise en main et réalisation d'un espace node red
    1. Création d'une gauge de température et humidité
    2. Utilisation des données GPS sur une carte en vue de la localisation géographique.
    3. Création de page HTTP
    4. Utilisation de différents capteurs
    5. Ordre de démarrage
- Programmation de différents capteurs (Température, humidité, UV, gaz…) sur keii studio
- Prise en main et réalisation d'un espace InfluxDB
    1. Création de database (flux de mon vélo)
    2. Rejoindre la database centrale (flux de plusieurs vélo)

Architecture

Nous avons débuté avec une structure architecturale qui nous a été fournie lors de notre première séance, elle est la suivante :

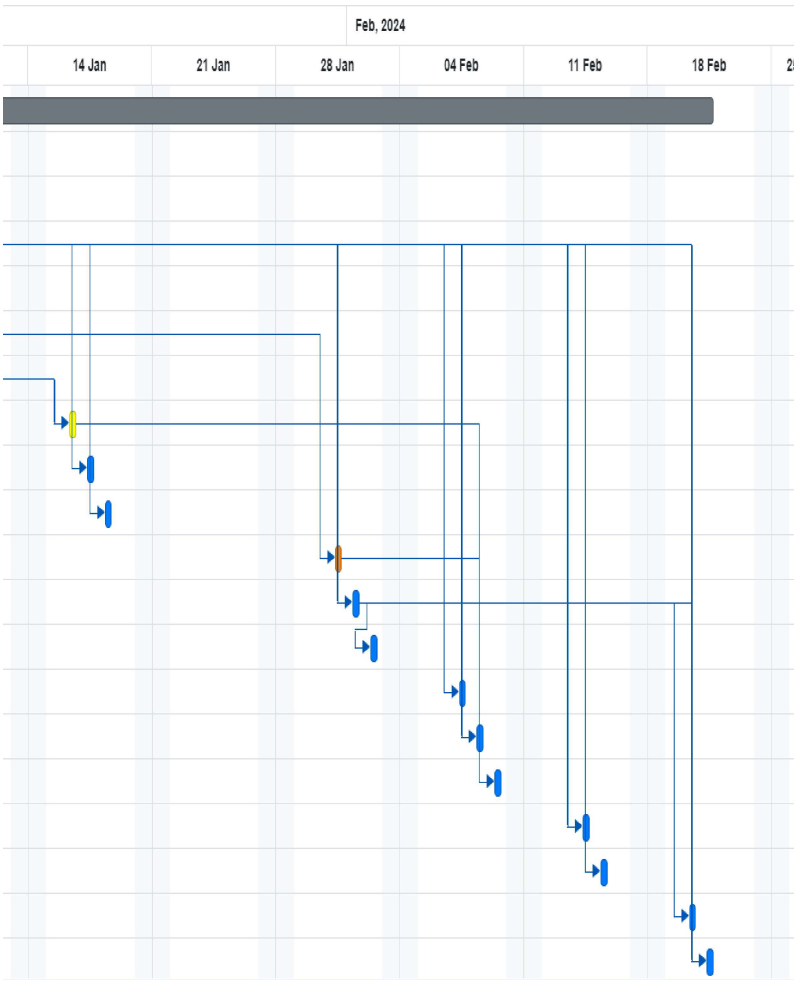


La partie de gauche étant nos vélos avec les cartes qui communiques avec une passerelle Lorawan et la partie droite étant The Things Network qui communique avec la passerelle et les différents serveurs (Node-red, InfluxDB, ect…)

2. Organisation du projet

Diagramme de Gantt

Nous avons décidé de faire un diagramme de Gantt, cet outil a été très utile pour voir l'avancée de nos tâches et avoir une vue globale sur le projet et quand on avait du retard sur nos tâches, on pouvait compenser notre retard sur nos week-ends. Pour optimiser la place sur notre rapport nous avons enlevé les vacances (annexe 1 : Gantt en entier)

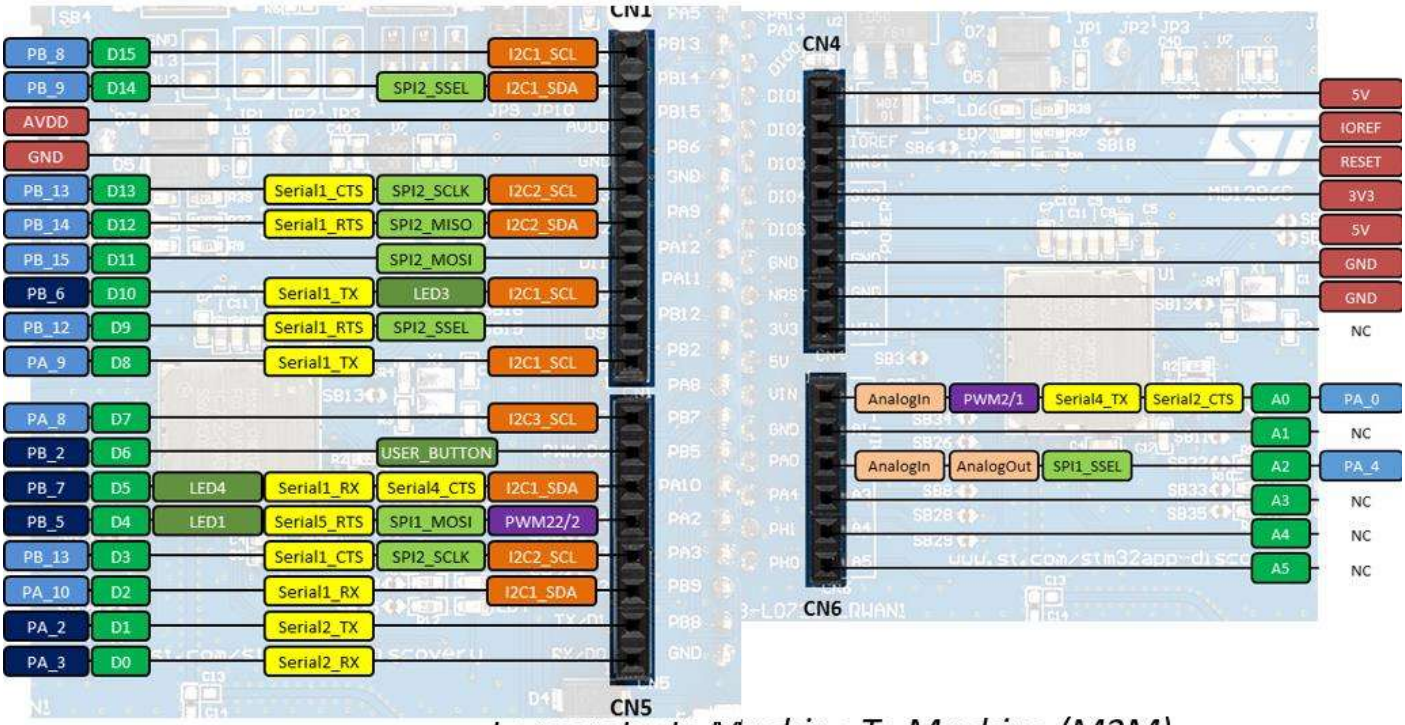


3. Création de la carte métier

Introduction

L'objectif d'implanter les différents composants de la carte dans le logiciel Design Spark PCB, de spécifier les différentes broches auxquelles je vais connecter les divers éléments de la carte

Pour commencer nous avons utilisé le cahier des charges, qui est inscrit que nous utiliserons la carte ARM mbed Disco L072 Lora, où nous utiliserons le con d'extension Arduino :



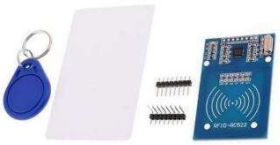
Le monde du Machine To Machine (M2M)

Sur cette carte nous utiliserons le 3.3V le 5V, le gnd, des pins digitales et analogiques pour les leds, des sorties I2C pour nos différents capteurs, les sorties SPI pour le lecteur de badge et une sortie RX-TX pour notre GPS.

Différents capteurs & actionneurs

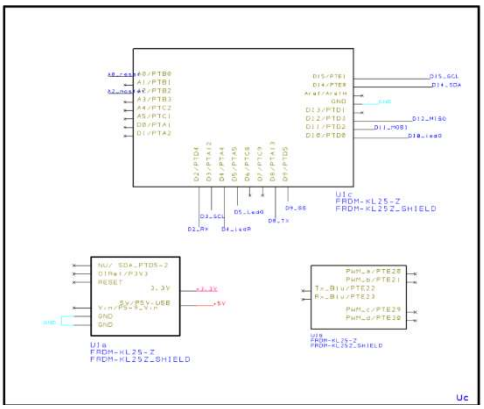
Pour ce projet, nous utiliserons différents capteurs et actionneurs donnés par le cahier des charges pour récupérer les données de notre tricycle et je vais vous les présenté :

- Un capteur i2c pour lire la température et l'humidité
- Un capteur i2c pour lire la luminosité
- Un capteur i2c pour lire le gaz (dioxyde d'azode)
- Un capteur GPS pour la localisation du véhicule (RX TX)
- Un lecteur de badge pour vérifier si l'adhérent a le droit de circuler ou pas avec notre vélo
- Trois leds V/O/R pour vérifier visuellement si l'adhérent a le droit de circuler
- Une led R pour vérifier visuellement si le véhicule démarre



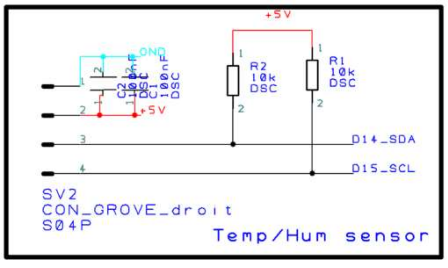
Routage avec Design Spark

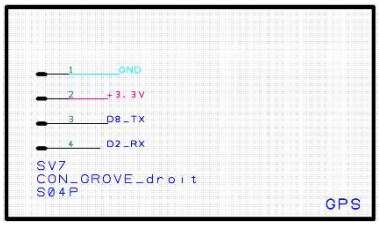
Je vais vous présenter les étapes de routage de la carte métier. On commence avec le schématique (annexe 2 : vue globale schématique).



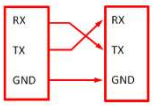
Nous avons ajouté le microcontrôleur pour reliez ses pins avec les éléments qui suivent.

Les 4 premiers groves sont là pour les différents capteurs i2c, on crée un bus i2c ou on met 2 condensateurs de filtrage (100nF) qui servent à réduire les variations de tension indésirables. Aussi deux résistances de pull Up de 10K ohmes qui permettrons de mettre le signal logique haut. Pour les pins on met la première sur GND, la deuxième sur le 5V pour alimenter le module et les pins 3 et 4 qui sont pour communiquer avec le microcontrôleur, la troisième est SDA et la dernière est SCL.

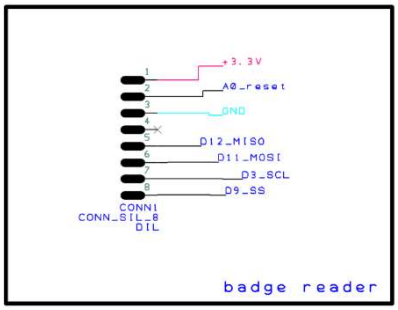


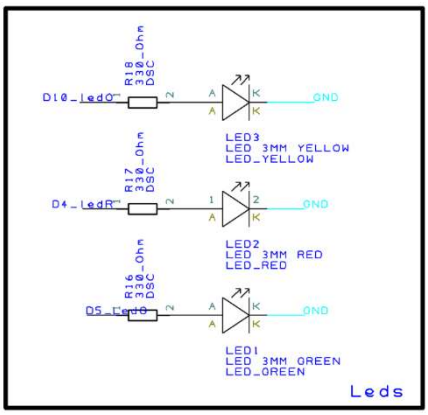


Ensuite pour continuer avec les capteurs le GPS utilise aussi un grove mais nous ne mettrons pas de résistances de pull n'y de condensateurs de lissage pour les pins du module GPS la première est la masse, la deuxième l'alimentation 3V3 la troisième TX et la dernière RX. Il ne faut surtout pas oublier d'inverser le RX TX.



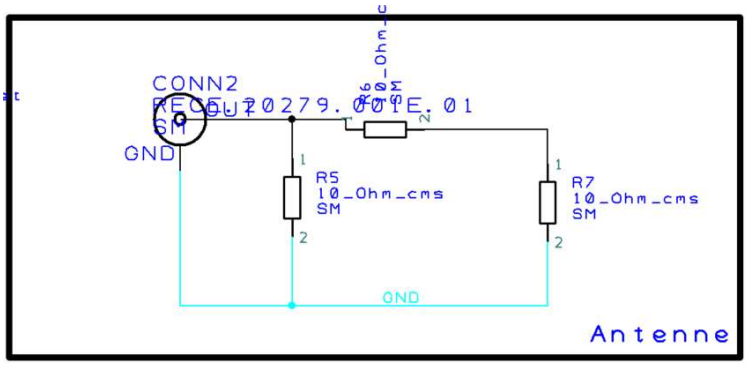
Pour finir avec les capteurs le lecteur de badge est plus compliqué il y a en tout 7 pins a connecté en tout : la première est l'alimentation 3v3 la deuxième est le reset, la troisième est la masse ensuite sur la cinquième broche se trouve miso, ensuite MOSI sur le pin 6 puis SCK sur la 7 et pour finir la pin 8 sur SS qui est le protocole SPI.

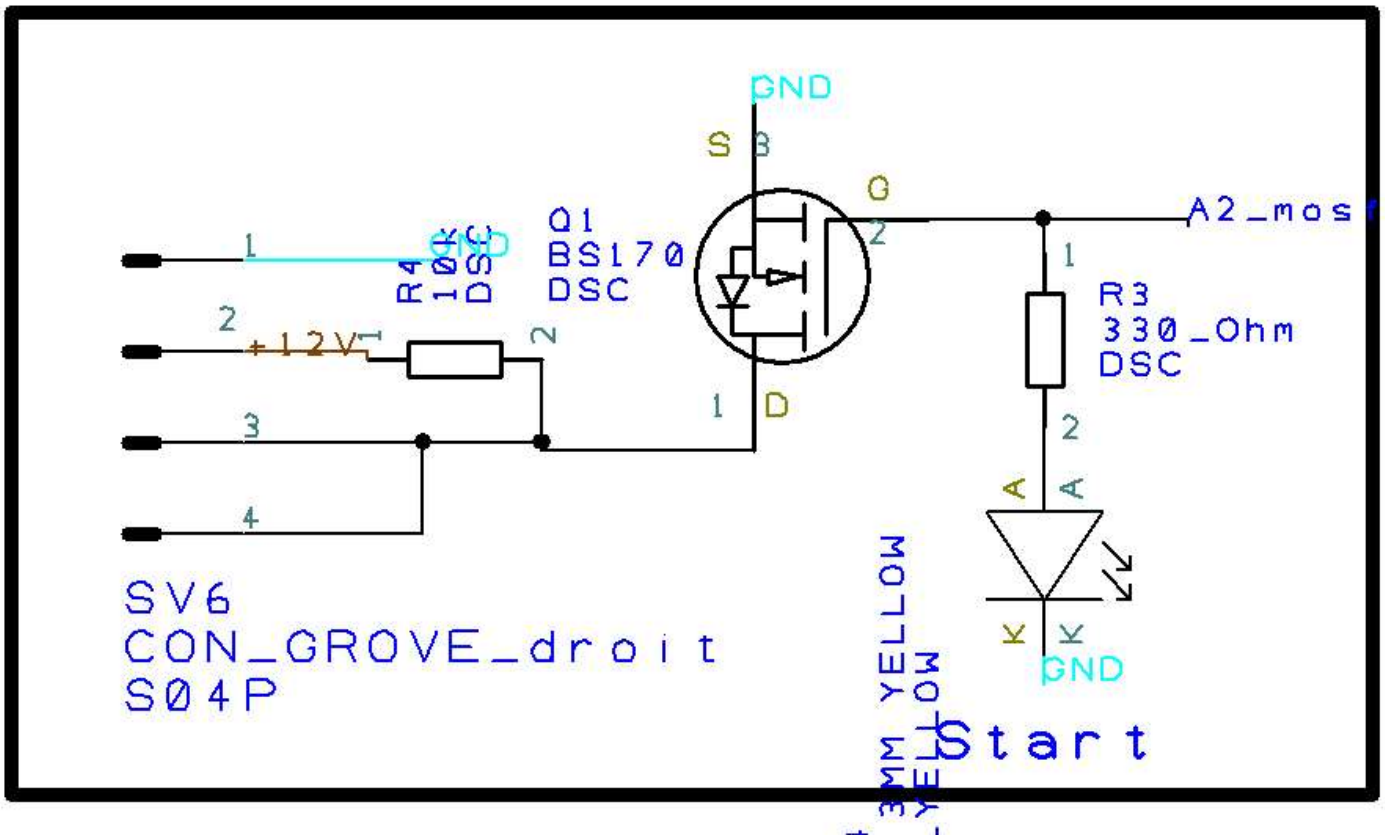




Nous avons les leds rouge verte et orange, ou nous avons calculé les résistances en fonction du courant de la led. Pour les contrôlé nous les avons mis sur des pins digitalOut du mbed en l'occurrence D10 D4 et D5.

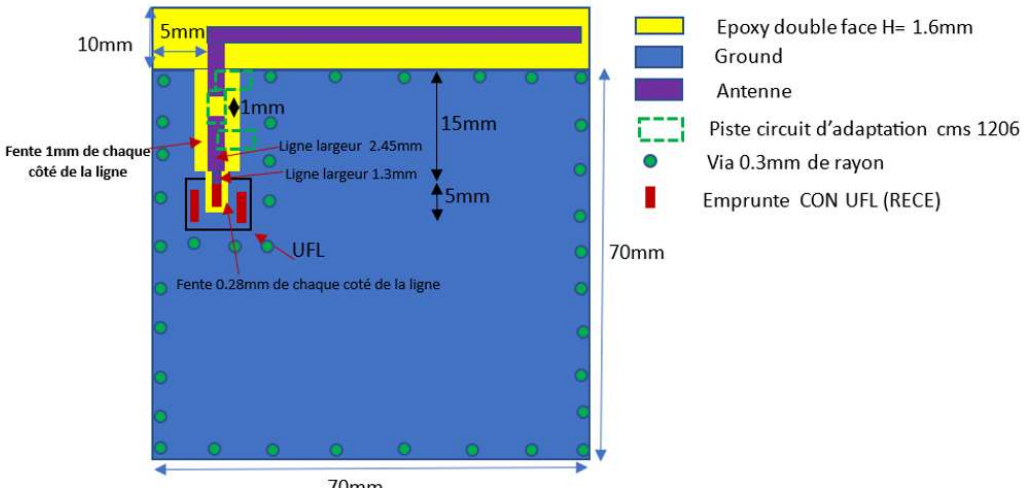
Pour l'antenne nous avons mis un composant CMS qui se nomme RECE pour plug l'entrée de l'antenne et la sortie de l'antenne ira directement sur la carte disco et on a mis 3 emplacements de résistances CMS pour par la suite adapter notre antenne et choisir les bons composants.





Puis passons au PCB, j'ai essayé de placer les composants le plus juste possible pour avoir le moins de vias et que la carte soit réduite pour respecter le cahier des charges. (Annexe 3 : PCB) (Annexe 4 : PCB sans le gnd sur toute la carte)

Nous avons eu quelques règles à respecter comme le pin 1 du lecteur du badge en bas à droite de la carte, la position de l'antenne et de ses composants, ainsi que l'emplacement spécifique de notre antenne. En plus, les dimensions de la carte ne doivent pas dépasser 70 mm x 75 mm.



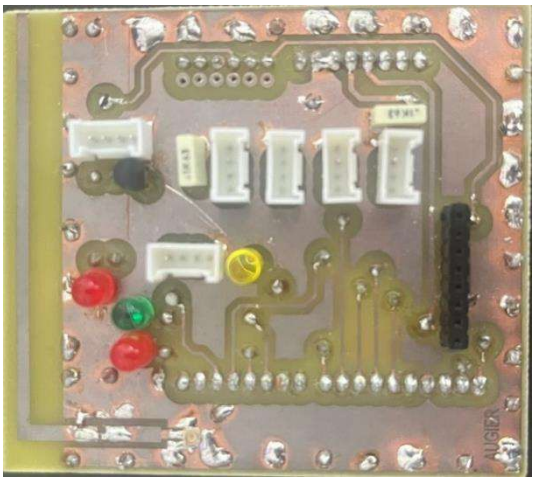
Cette carte est une double face car un côté doit accueillir la carte disco et l'autre côté pour connecter les différents capteurs et la visualisation des leds.

Quand nous avons fini le schématique et notre PCB nous les avons fait vérifier aux enseignants qui nous les ont fait passer en phase d'impression puis quand la carte était imprimée nous l'avons récupéré puis fait des trous pour faire passer nos composants, puis nous les avons soudés des moins haut au plus haut pour faciliter le soudage.

Quand tous étaient soudés, nous avons vérifié nos pistes si elles étaient bien imprimées et que nos soudures étaient bonnes avec un multimètre mis sur cette fonction : 

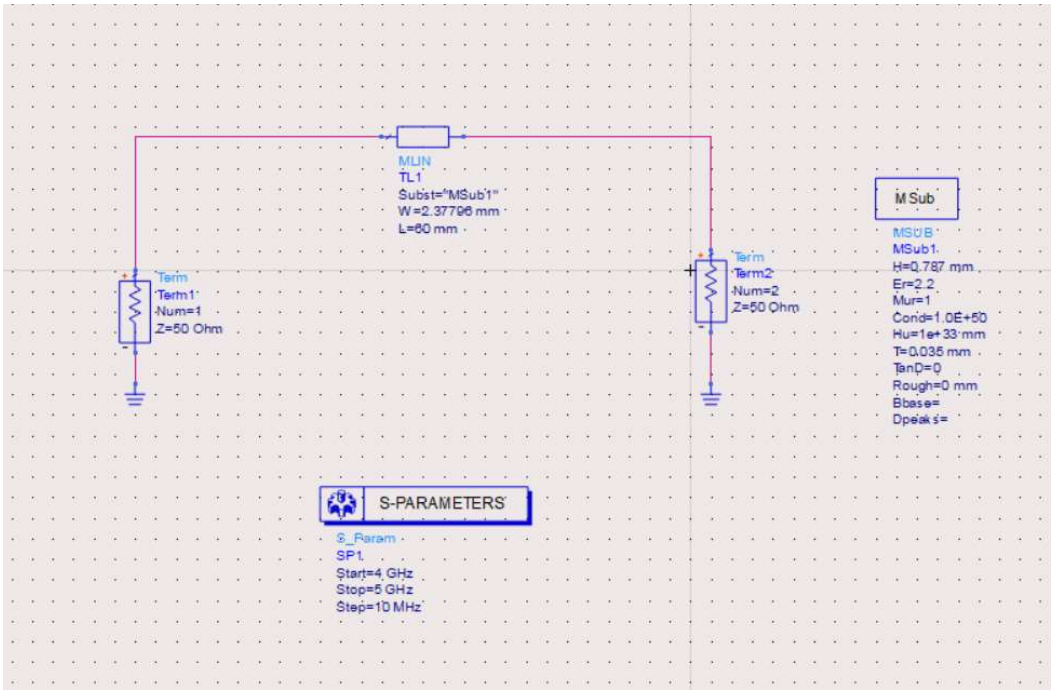
Lorsque tout était en ordre, nous avons pu effectuer des tests avec les différents éléments de la carte.

Voici la carte avec les différents éléments soudés



Création de l'antenne avec abs

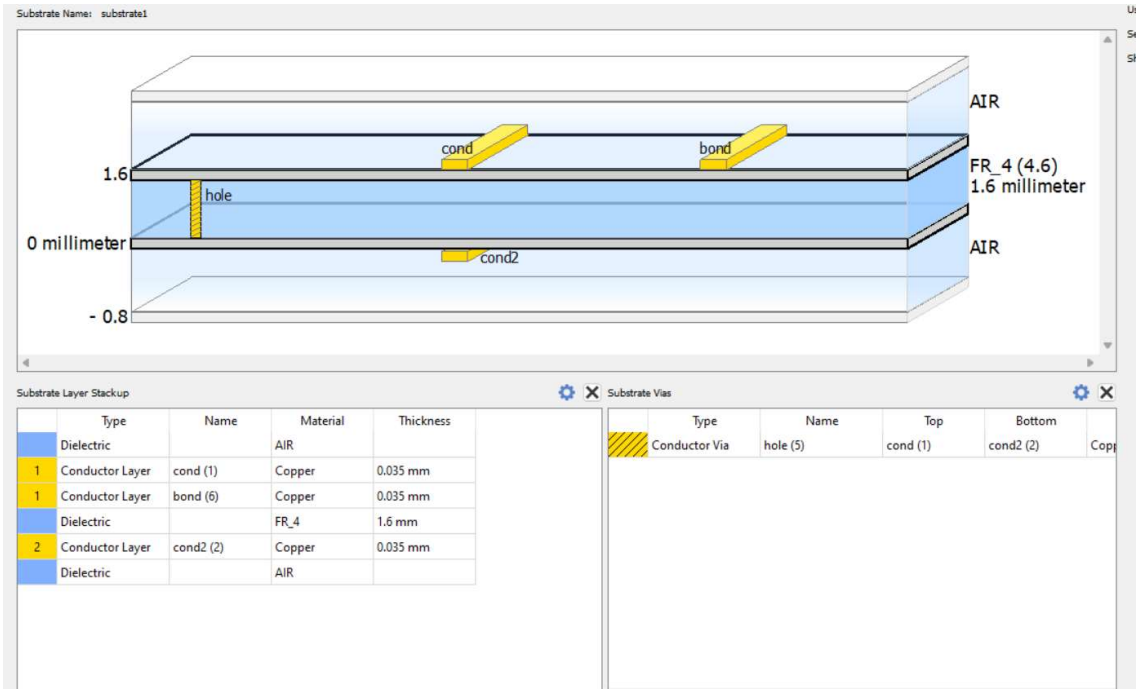
Pour faire l'antenne de la SAE Lora on utilisera le logiciel Advanced Design System 2024. Ce logiciel permet de créer son antenne de A à Z et de pouvoir l'adapter. Tout d'abord nous allons apprendre comment utiliser le logiciel avec 2 exercices qui nous permettra de faire notre propre antenne pour envoyer une trame de données. Dans ces exercices, nous apprendrons comment faire un substrat et adapter une charge.



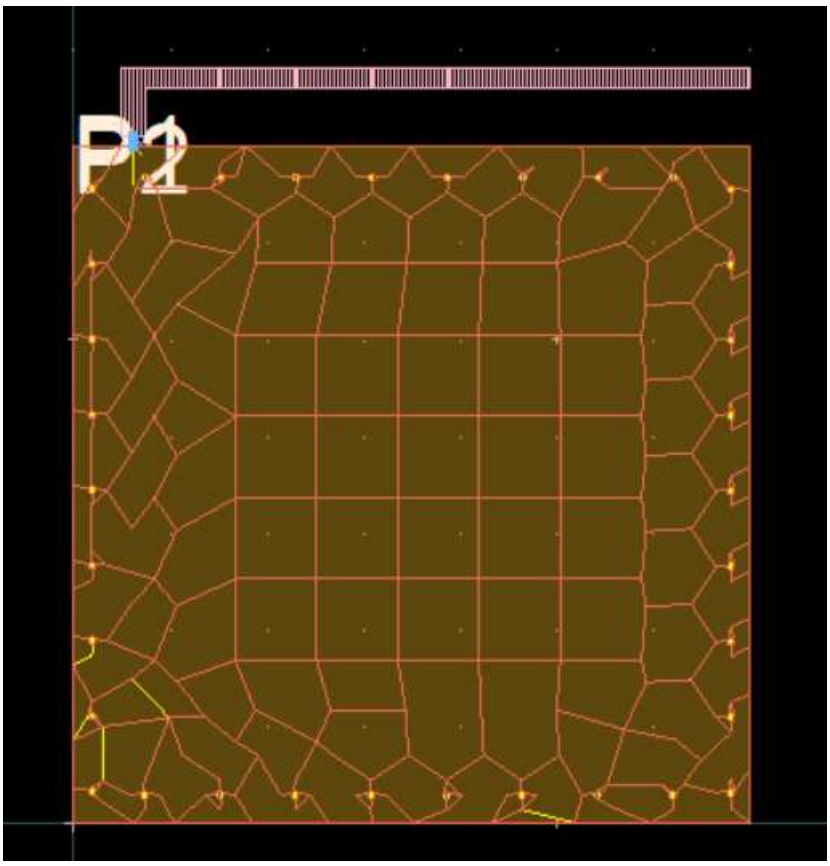
Passons à la création à l'antenne qu'on mettra sur notre carte métier.

Voici le schématique.

Voici mon substrat ci-dessous :



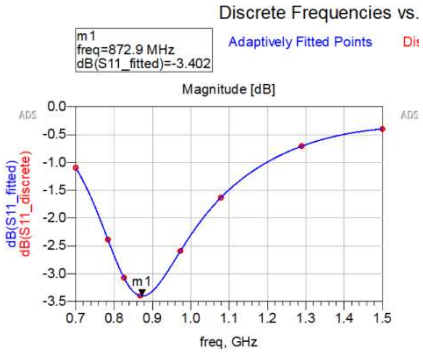
Après le substrat, on fait le PCB, on met des vias et on crée un rectangle pour représenter la carte. En plus, pour la partie de l'antenne nous le ferons aussi avec des rectangles.

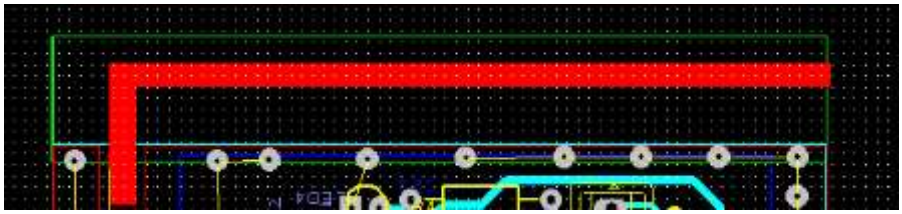


Pour suivre, nous ajustons l'antenne pour que le bas de la courbe soit à précisément 868Mhz.

Nous l'ajustons avec la longueur et la largeur de l'antenne.

(Annexe 5 : VUE EN 3D)





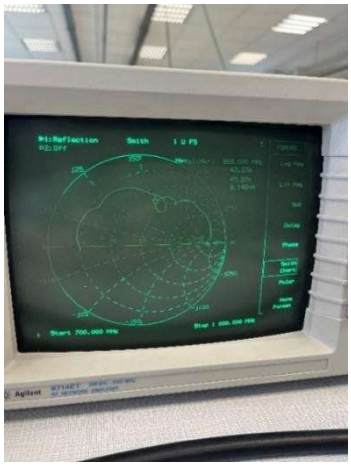
Ainsi on exporte notre antenne de ABS a design spark et on la place sur son emplacement.

Adaptation de l'antenne avec abs

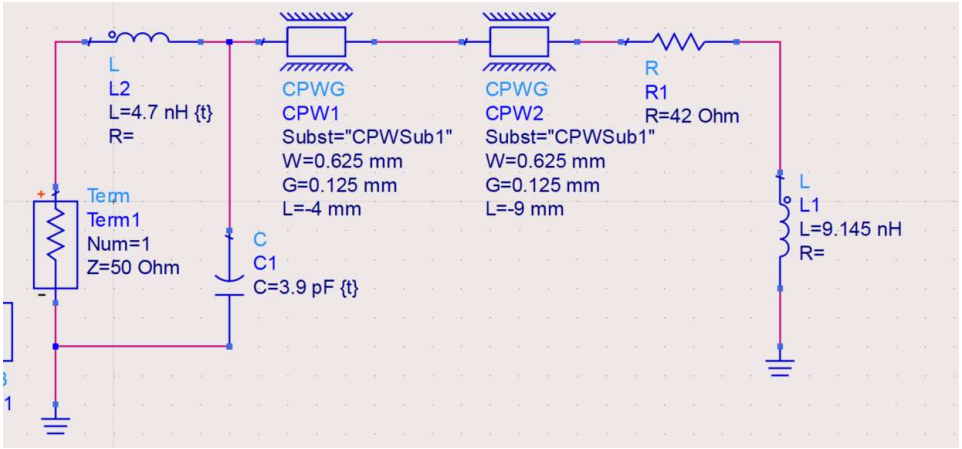
Une fois notre carte confectionnée à l'aide d'abs et de design spark, elle a été imprimer et souder.

Nous avons ensuite effectué le test à l'aide de l'analyseur de réseau ci-dessous afin de connaître notre gain en dB et de surtout pouvoir adapter notre antenne à l'aide des résultats ci-dessous.

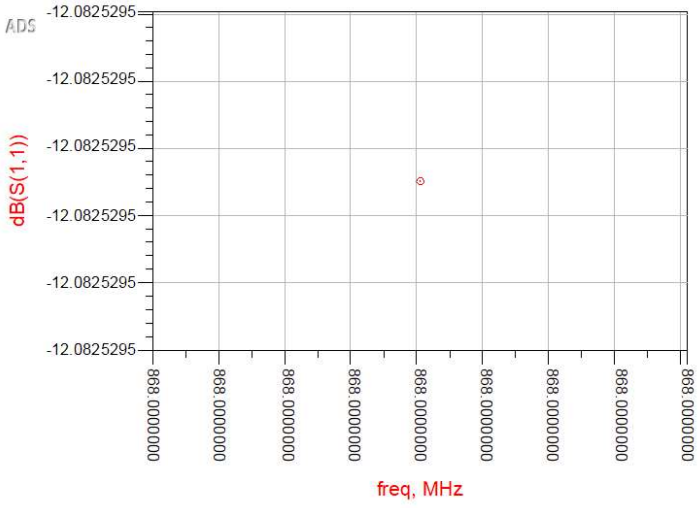
Ancienne antenne 90dB :



À la suite de ce test, nous somme retourner sur ADS afin de simuler le circuit d'adaptation et de pouvoir dimensionner les composants nécessaires pour avoir la meilleure antenne possible.

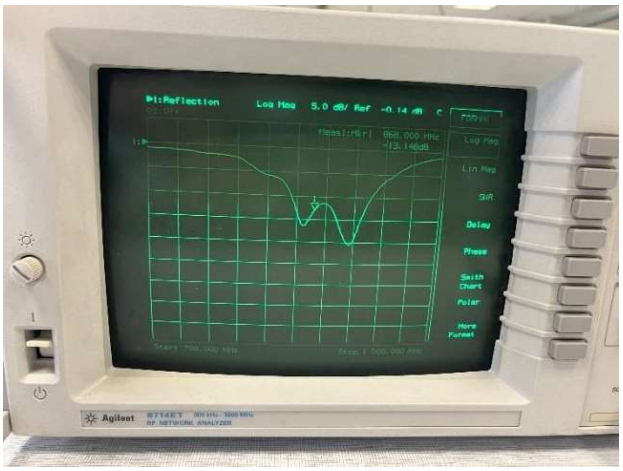


Pour que l'antenne soit bien adaptée nous avons mis un condensateur de 3.9 pF en parelle et une inductance de 4.7 nH en série. Comme le montre la representation polaire (on voit bien le point sur 868Mhz)(Annexe 6 : l'abaque de smith)

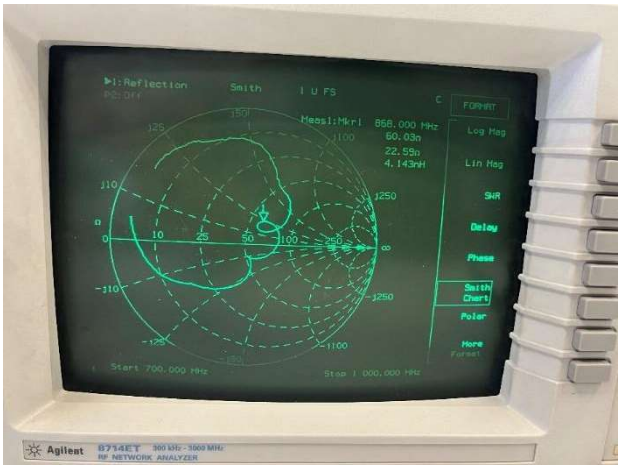


C’est ainsi que on reproduit l’opération maintenant avec l’analyseur de réseau et nous avons obtenu les résultats ci-dessous.

Nouvelle antenne 80 dB :



Représentation polaire 868Mhz : -13.146 dB



L’abaque de Smith 868Mhz : 60.03 ohmes

22.59 ohmes

4.123 nH

À la suite de ses derniers tests, nous avons pu observer que notre antenne été de très bonne qualité, car nous avons gagné 10dB sur l’envoie de flux de données ttn. (90dB → 80dB)

4. Partie TTN

The Things Networks est une plateforme cloud conçue pour récupérer les données émanant des Gateway LoraWAN. Son objectif principal est de réacheminer ces données vers d'autres applications externes, telles que Mbed Compiler ou Node-Red. Pour ce faire, nous utilisons un broker MQTT.

Ainsi un dispositif comme notre carte Mbed peut transmettre ces données. Grâce à ce broker, des échanges de données sont possibles entre Mbed et TTN, ainsi qu'entre TTN et Node-Red. De plus, les données peuvent également être transmises de Node-Red à TTN puis de TTN à Mbed.

Configuration

Avant de pouvoir observer les échanges de données entre Mbed et Node-Red, nous devons d'abord configurer The Things Networks. Pour commencer, il est nécessaire de créer un compte en utilisant une adresse e-mail et un mot de passe. Une fois le compte TTN créé, il faut suivre les exercices de la première séance du TP afin de s'entraîner et de se familiariser avec le logiciel.

La première étape consiste à créer une application en spécifiant l'ID de l'application, le nom de l'application et une description. Une fois l'application créée, nous configurons un périphérique qui sera inséré dans le code source Mbed de la carte. Ensuite, nous choisissons la méthode d'activation du périphérique, qui dans ce cas est l'Activation par Personnalisation (ABP), ainsi que le plan de fréquence, la version LoRaWAN et les paramètres régionaux. Nous générons ensuite les codes associés, tels que le DevEUI, l'adresse du périphérique, App SKey et NwkSKey, en cochant également la case "Réinitialiser les compteurs de trames". Enfin, nous copions les clés générées dans le fichier Mbed_app.json, notamment le DevEUI, l'App SKey, le NwkSKey et l'adresse du périphérique.

```
"lora.device-eui": " {0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x06, 0x41, 0x64}",
"lora.application-eui": "{ 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x02, 0x16, 0x8E }",
"lora.application-key": "{ 0x77, 0x8A, 0x26, 0xDF, 0x9D, 0x70, 0xDB, 0xEF, 0x43, 0x03, 0x0F, 0xD2, 0x5F, 0x47, 0x1D, 0x00 }",
"lora.appskey": " {0x0C, 0x3D, 0x18, 0xCB, 0x20, 0x69, 0x15, 0xAA, 0x1D, 0x04, 0xAD, 0x28, 0xFF, 0x64, 0xF6, 0x91} ",
"lora.nwkskey": "{0x4A, 0xAE, 0x7F, 0x14, 0x51, 0xBA, 0x5F, 0xAD, 0x80, 0x5C, 0x5E, 0x8C, 0x0A, 0x74, 0xF2, 0xD6}",
"lora.device-address": " 0x260BC997"
```

Ensuite dans l'espace uplink de ttn, nous avons du passé en Custom Javascript formatter et faire une fonction qui permet de mettre la trame Cayenne dans des variables.

Formatter type *

Custom Javascript formatter

Formatter code *

```
1  function decodeUplink(input) {
2    var bytes=input.bytes;
3    var temp =  bytes[2] << 24 >> 16 | bytes[3];
4    var hum = bytes[6];
5    var lon= bytes[12] << 16 | bytes[13] << 8 | bytes[14];
6    var lat = bytes[9] << 16 | bytes[10] << 8 | bytes[11];
7    var alt = bytes[15] << 16 | bytes[16] << 8 | bytes[17];
8    var RFID=((bytes[19]>>4&0x0F)).toString(16)+((bytes[19]&0x0F)).toString(16)+((bytes[20]>>4&0x0F)).toString(16)+((bytes[20]&0x0F)).toString(16)+((bytes[
9  var VEC=((bytes[28]>>4&0x0F)).toString(16)+((bytes[28]&0x0F)).toString(16)+((bytes[29]>>4&0x0F)).toString(16)+((bytes[29]&0x0F)).toString(16)+((bytes[30]
10 +((bytes[31]>>4&0x0F)).toString(16)+((bytes[31]&0x0F)).toString(16);
11 var UV=bytes[25] << 8 | bytes[26]  ;
12 var gas= bytes[34] << 8 | bytes[35]  ;
13
14 return {
15    data: {
16    temp : temp/10,
17    hum : hum/2,
18    lat : lat/10000,
19    lon : lon/10000,
20    alt : alt/100,
21    RFID : RFID,//.toString(16)
22    VEC:VEC,
23    UV:UV,
24    gas : gas/100
25    },
```

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB Longitude : 0.0001 ° Signed MSB Altitude : 0.01 meter Signed MSB

On peut prendre l'exemple de la température on voit sur le tableau à gauche que la taille du « temperature sensor » est de 2, donc il y aura 1 octet pour le numéro de port, un autre pour le type de données envoyée et donc les 2 derniers sont la valeur de la température.
Si on regarde le code en haut, ligne 3 on voit que on prend uniquement les octets 2 et 3 qui sont la leur de la température. Car l'octet 0 et 1 sont utilisé pour le numéro de port et le type de donnée

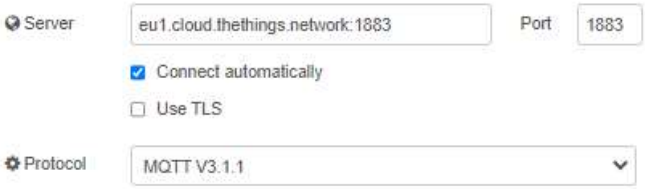
5. Programmation (Node red/Mbed/InfluxDB)

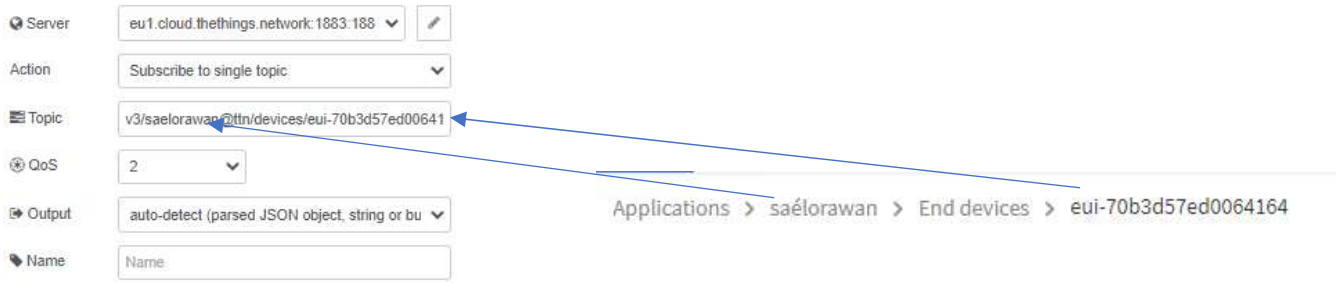
Configuration

Dans cette partie allons parlé de la configuration node red et Mbed. Dans influxDB il n'y pas de prérequis a par lancer l'exe.

Tour d'abord Node-RED est un outil de développement visuel qui permet de créer des applications IoT (Internet des objets) en reliant des nœuds préprogrammés pour automatiser des tâches et faciliter la gestion des flux de données

On a créé 2 blocs le premier pour s'abonne à notre flux ttn et l'autre pour envoyer des donné à l'aide de down Link vers ttn. La configuration est presque la même pour les deux blocs. Juste a la fin du topic pour le MQTT IN il faut maquer « up » et pour le MQTT OUT il faut écrire « down »

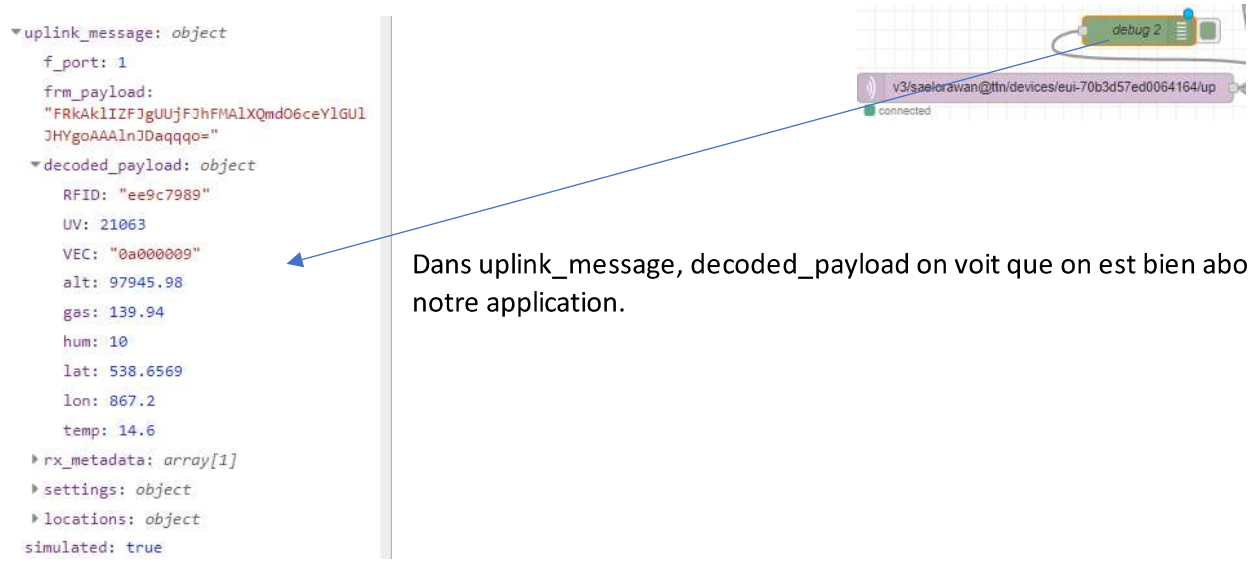




Bloc : MQTT IN

Bloc : MQTT OUT

Pour vérifier si on est bien abonné à notre application on peux le vérifier avec le bloc debug :



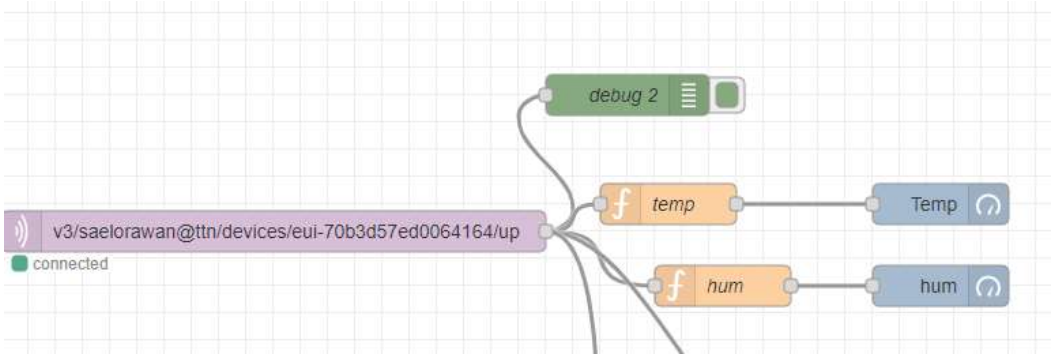
Dans uplink_message, decoded_payload on voit que on est bien abonné a notre application.

Maintenant pour keii studio, on avait déjà un projet deja fait et on devait le compléter pour les taches qu'on devait faire comme par exemple initialiser tous les pins utilisés par notre carte métier.

Gauge & Downlink

Notre premier exercice pour cette sae était faire des gauges de température et de température et d'allumé une led sur le microcontrôleur grâce à la température de la salle (on l'a mesuré avec le capteur i2c).

Tout d'abord nous allons créer cette structure sur node red

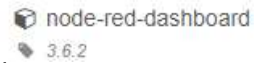


Après s'être abonné à notre application on crée deux fonctions une pour la température et l'autre pour l'humidité

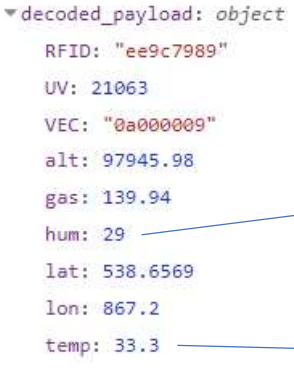
```
msg.payload = msg.payload.uplink_message.decoded_payload.temp;
return msg;
```

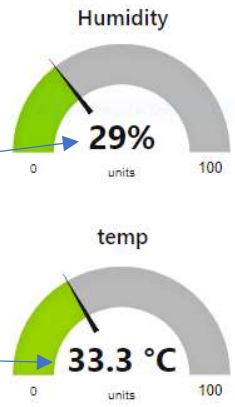
```
msg.payload = msg.payload.uplink_message.decoded_payload.hum;
return msg;
```

Pour les gauges il faut ajoutez cette bibliothèque dans les palettes



En dernier lieu, allez sur le lien suivant : http://127.0.0.1:1880/ui et vous allez avoir votre gauge de température et d'humidité.

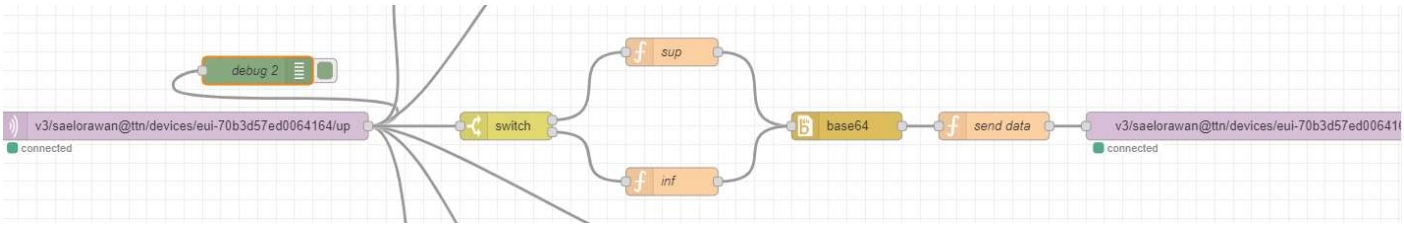




Debug : affiche les valeurs que reçoit node red

Dashboard : affichage des température et humidité avec des gauges

Dans un second temps on s'occupe de l'allumage de la led. On crée la structure suivante.



Premièrement, on crée un switch qui sort dans 2 fonctions différents, si la température est au-dessus ou égale de 28 degrés il rentre dans la fonction « sup ». S'il est en dessous de 28 il rentre dans la fonction « inf ».

On crée une variable globale msg.valuettn, et la valeur de température affecte une valeur de chaine de caractère, soit ""+0+"" soit ""+1+"" à la variable globale.

Pour finir, on convertie le résultat en base 64 on prépare le message avec une fonction. Enfin on l'envoie en downlink sur ttn quand on reçoit l'information, le code keii va allumer la led ou pas.

31 car >29.3

30 car 28>18.1

```
switch (port) {
case 3: // control led
  printf("\n led=%x", (int)rx_buffer[0]);
  if ((rx_buffer[0] - 0x30) == 0) // ascii command for led
                                  // if (rx_buffer[0]==0)
    iEtatAlarme = 0;
  else
    iEtatAlarme = 1;
  Alarme.write(iEtatAlarme);
```

On rentre dans le case 3 car on est FPort 3 et si on reçoit 0x30 on met la variable iEtatAlarme a 0 sinon on l'a met a 1. Puis on l'écrit sur la led donc 1 elle s'allume et 0 elle s'éteins.

GPS

Nous avons développé un programme permettant de récupérer les données GPS, comprenant la latitude, la longitude et l'altitude. Initialement, ces données sont affichées sur TeraTerm, puis transmises à TTN afin que Node Red puisse les récupérer et les afficher sous forme de carte.

Pour celui de le GPS il avons eu à mettre une bibliothèque

```
26  #include "GroveGPS.h"
```

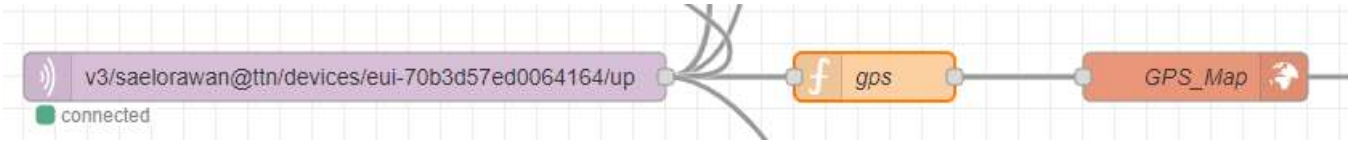
Initialisé le RX et TX

```
61  GroveGPS MyGPS(D8, D2);
```

Ensuite dans la boucle « send message » nous actualisons le capteur et nous l'envoyons sur ttn sur le channel 3

```
MyGPS.update();
Payload.addGPS(3, MyGPS.gps_gga.latitude, MyGPS.gps_gga.longitude,
               MyGPS.gps_gga.msl_altitude);
printf("GPS=%.2f %.2f % .2f\n", MyGPS.gps_gga.latitude,
       MyGPS.gps_gga.longitude, MyGPS.gps_gga.msl_altitude);
```

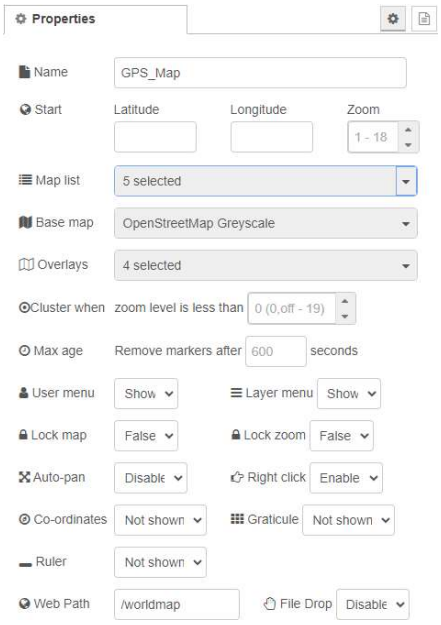
Pour la partie node red on est abonné a l'application puis on rentre dans une fonction




```
var obj= {"name" : "0A000009",
        "lat": msg.payload.uplink_message.decoded_payload.lat,
        "lon": msg.payload.uplink_message.decoded_payload.lon,
        "altitude": msg.payload.uplink_message.decoded_payload.alt,
        "Humidity": msg.payload.uplink_message.decoded_payload.hum,
        "Temperature": msg.payload.uplink_message.decoded_payload.temp,
        icon: "glass",
        iconColor : "red"
    };
msg.payload=obj;
return msg;
```

Dans la fonction on inscrit les informations nécessaires pour avoir notre localisation sur la map

Dans la palette nous utiliserons le module « node-red-contrib-web-worldmap »



Dans ce bloc on insert la latitude, la longitude et l'altitude et avec http://127.0.0.1:1880/worldmap on voit notre point sur la carte au niveau de l'iut de nice.

Capteurs i2c

Pour le capteur de température et humidité il était déjà fait

```
fTemp = myTH02.ReadTemperature();
printf("Temp=%.2f\t", fTemp);
fHumid = myTH02.ReadHumidity();
printf("Humidity=%.2f\n", fHumid);
// Payload is in Cayenne format
Payload.reset();
Payload.addTemperature(1, (float)fTemp); // Add Temp in payload
Payload.addRelativeHumidity(2, fHumid);  // Add Humidity in payload
```

Pour je vais vous expliquez, celui de l'uv nous avons eu à mettre une bibliothèque `#include "SENSOR.h"` initialiser ses pins `SENSOR UV(D14, D15); // UV sensor`

D'écrire une ligne qui permet d'actualiser les valeurs du capteur `UV.inicial(); //actualisation des data`

Puis on récupéré la valeur dans un entier et l'envoie sur ttn

```
uvread = UV.getUV(); //recup data uv
Payload.addLuminosity(5, uvread); //envoie vers ttn UV
printf("Luminosity=%.2f\n", uvread);//affichage console
```

Pour le capteur de gaz c'est exactement le même principe :

```
#include "MiCS6814_GasSensor.h"
```

```
54    MiCS6814_GasSensor GAS(D14, D15);
258       printf("NH3: %.2f ppm \r\n", GAS.getGas(NO2));
259       Payload.addAnalogInput(7, GAS.getGas(NO2));
```

Lecteur de badge

Pour le lecteur de badge nous de l'initialisé les broches du spi

```
43    MFRC522 RfChip(SPI_MOSI, SPI_MISO, SPI_SCLK, SPI_CS, MF_RESET);
```

Puis envoyé la valeur du badge sur ttn s'il arrive à lire un badge, il enverra toujours une chaine de caractère s'il recoit un badge il enverra la valeur du badge et sinon il envoie un 00 00 00 00. On fait cela pour que la réception des donnés qui suivent ne soient pas décalé.

```
235       if (RfChip.PICC_IsNewCardPresent()) { // si il y a un badge
236           // wait_ms(500);
237           if (RfChip.PICC_ReadCardSerial()) { // si on arrive a le lire
238               // wait_ms(500);
239               printf("Card UID: ");
240               for (uint8_t i = 0; i < RfChip.uid.size; i++) {
241                   printf(" %02X", RfChip.uid.uidByte[i]);
242               }
243               printf("\n\r");
244               Payload.addNFC(4, (char *)RfChip.uid.uidByte); //envoie la val du badge sur ttn
245           } else {
246               Payload.addNFC(4, "\0\0\0\0"); //sinon que des 0
247           }
248       } else {
249           Payload.addNFC(4, "\0\0\0\0"); //sinon que des 0
250       }
```

Véhicule

Pour le véhicule on envoie juste un tableau avec notre numéro de table (9) à TTN.

```
211       char VEC[] = {0xA, 00, 00, 9}; // numéro de table
```

```
256       Payload.addVEHICLE(6, VEC);
```

Apres réception à l'aide de Node Red en envoie à notre database sur Influxdb.

Database

Pour la database on utiliseras influxDB, pour commencer il faut lancer un serveur avec le raccourci « influxd.exe »

Quand le serveur est lancé on ouvrira « influx.exe » qui sera notre cmd pour écrire nos commandes

Pour commencer on va créer notre première data base qui se nomme « ALT_2024 »

Pour crée notre data base il faut faire la commande « Create database ALT_2024 »

La commande « show databases » permet de voir les databases qui sont déjà créé.

```
> show databases
name: databases
name
----
_internal
alt_2024
ALT_2024
kiffy_data
```

Puis créons notre première table « grp9_sae » pour la remplir avec le flux de donnée ttn, pour cela on utilisera node red

Dans Node Red on voit bien que on est abonné aux devices puis on rentre dans une fonction qui met les uplink dans les variables suivantes : LAT LON ALT HUM TEMP NFC VEC GAZ UV celle-ci doit avoir des noms précis pour la database centrale (annexe 3)

```
msg.data = {
    LAT : msg.payload.uplink_message.decoded_payload.lat,
    LON : msg.payload.uplink_message.decoded_payload.lon,
    ALT : msg.payload.uplink_message.decoded_payload.alt,
    HUM: msg.payload.uplink_message.decoded_payload.hum,
    TEMP: msg.payload.uplink_message.decoded_payload.temp,
    NFC:  msg.payload.uplink_message.decoded_payload.RFID,
    VEC : msg.payload.uplink_message.decoded_payload.VEC,
    GAZ:  msg.payload.uplink_message.decoded_payload.gas,
    UV:  msg.payload.uplink_message.decoded_payload.UV,

}
msg.payload=msg.data;
return msg;
```

Dans la palette nous utiliserons le module « node-red-contrib-influxdb »

Pour remplir la database avec notre flux de données.

On le configurera de la façon suivante : dans la database « ALT_2024 » avec l'host qui est l'ip du serveur influxDB avec le port 8086 et la table gr9_sae.

On voit clairement sur influxDB le flux envoyé par ttn grâce a la command « select * from gr9_sae », on peut même apercevoir quand le badge est lu ou non.

```
> select * from gr9_sae
name: gr9_sae
time                ALT  GAZ   HUM  LAT      LON     NFC       TEMP  UV  VEC
----                ---  ---   ---  ---      ---     ---       ----  --  ---
1708079287049983300 46   10.97 104  43.6866  7.2291  00000000  -50   1
1708082989997290200 45.4 5.58  104  43.6867  7.229   00000000  -50   1   0a000009
1708083004224713100 45.4 5.58  104  43.6867  7.229   00000000  -50   1   0a000009
1708083015863156500 46.2 5.7   104  43.6867  7.2291  00000000  -50   2   0a000009
1708083028905390600 46.3 5.66  104  43.6867  7.2291  ee9c7989  -50   1   0a000009
1708083046141735200 46.5 5.58  104  43.6867  7.2291  8322faa4  -50   1   0a000009
1708083047335845800 46.5 5.54  104  43.6867  7.2291  00000000  -50   1   0a000009
1708083052970694100 48.1 5.78  104  43.6867  7.2291  8322faa4  -50   1   0a000009
1708083059722653600 48.3 5.93  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083065256733700 48.3 5.89  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083072013200100 48.3 5.89  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083078674976700 48.3 5.93  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083085426752300 48.3 5.85  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083093390945100 48.3 5.89  104  43.6867  7.2291  00000000  -50   2   0a000009
1708083102473092900 49.8 5.85  104  43.6866  7.2292  00000000  -50   2   0a000009
1708083105556003600 49.7 5.81  104  43.6866  7.2292  00000000  -50   2   0a000009
1708083112254868000 49.7 5.85  104  43.6866  7.2292  00000000  -50   2   0a000009
```

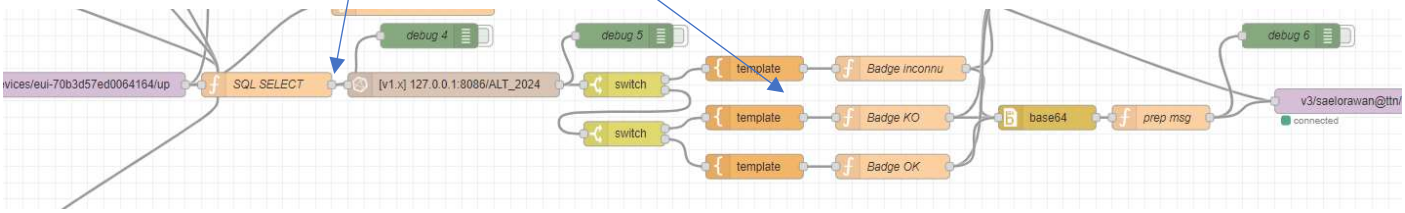
Pour terminer on avait comme mission de mettre notre database (tableau : grp9_sae) dans la database central, elle consiste de mettre toutes les databases de chaque groupe en commun cela permet de centraliser toutes les informations des vélos c'est super intéressant pour de la supervision.

Ordre de démarrage

Sur InfluxDB on va créer une nouvelle table qui se nomme « Badge » et dans cette table on va mettre 2 badges, un ou il aura la variable dem a 0 et l'autre badge la variable dem sera à 1.

```
> select * from Badges
name: Badges
time                Badge_ID DEM
----                -------- ---
1708081844078883100 ee9c7989 1
1708081874403475200 8322faa4 0
1708584414456756300 64fa6e1a 0
```

Sur node red on arrive à récupérer les valeurs du badge et on va vérifier dans la table « Badge » sur influxDB s'il connait le badge. SI oui il rentre dans un notre switch qui vérifie notre variable dem si dem est a 1 il enverras en downlink la valeur 0 (en ascii 30) si dem est a 0 il enverra la valeur 1 (en ascii 31). S'il ne connait pas le badge il enverra 2 (en ascii 32)



Après sur notre microcontrôleur on va dans la boucle « receive a message », ensuite dans la case on vérifie le port ou on reçoit dans mon cas 3 donc dans le case 3

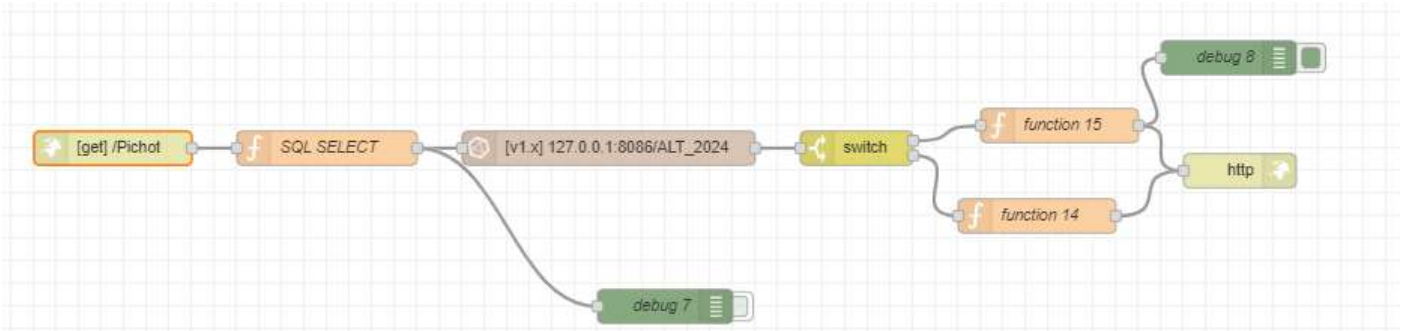
```
switch (port) {
case 3: // control led
```

Ensuite dans le switch on regardera qu'elle valeur en ascii on reçoit 30 qui signifie que l'abonnement est épuisé ,31 que la badge est connu et que l'abonnement est payé et 32 veux tout simplement dire que le badge est inconnu.

```
320         printf("\n led dem=%x", (int)rx_buffer[0]);
321     if (rx_buffer[0] == 0x30) { // badge connu mais ko
322       Led_R.write(0);
323       Led_O.write(1);
324       Led_V.write(0);
325     } else if (rx_buffer[0] == 0x31) { // badge connu + valide
326       Led_R.write(0);
327       Led_O.write(0);
328       Led_V.write(1);
329     } else if (rx_buffer[0] == 0x32) { // badge inconnu
330       Led_R.write(1);
331       Led_O.write(0);
332       Led_V.write(0);
333     }
334     break;
```

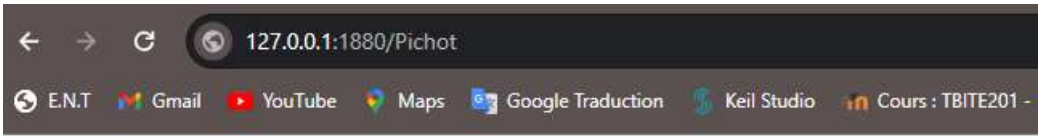
Get HTTP

Sur node red on crée cette structure



On ajoute les deux blocs « http in » et « http out ». Le premier crée un serveur client htpp et le deuxième envoie sur ce serveur une valeur.

Comme pour le démarrage on va chercher dans la database un badge. Ensuite on voit les informations recu par ttn et envoie sur notre serveur web.



{"time :"2024-02-24:29:56.57","temp" : 33.3,"RFID":"ee9c7989","gaz" : 5,"hum":20}

6. Conclusion

Pour conclure, nous avons pu utiliser ce qu'on a appris dans les cours théoriques du BUT sur un projet applicatif, on a même appris comment utiliser ttn, node red et influxDB.

On est très content de ce projet car on a compris et appris beaucoup de choses cella pourra nous permettre d'utiliser les compétences obtenues dans nos entreprises.

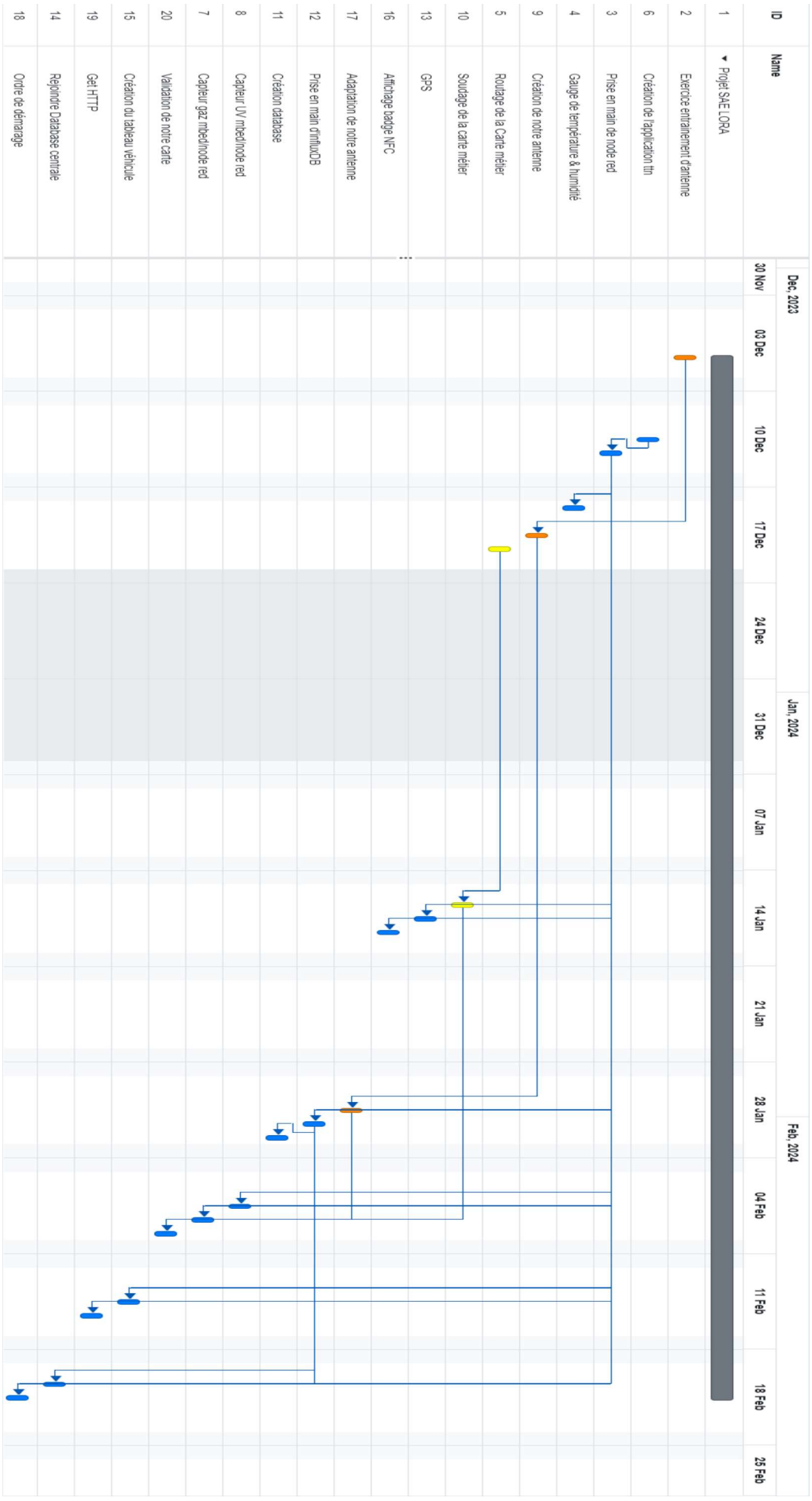
Par exemple personnellement dans mon entreprise je suis actuellement sur un chantier ou on doit faire une gestion technique centralisé et nous utilisons des Wattsense.

Qui est une passerelle Lorawan & GSM pour envoyer les valeurs lues des différents capteurs sur une page web.

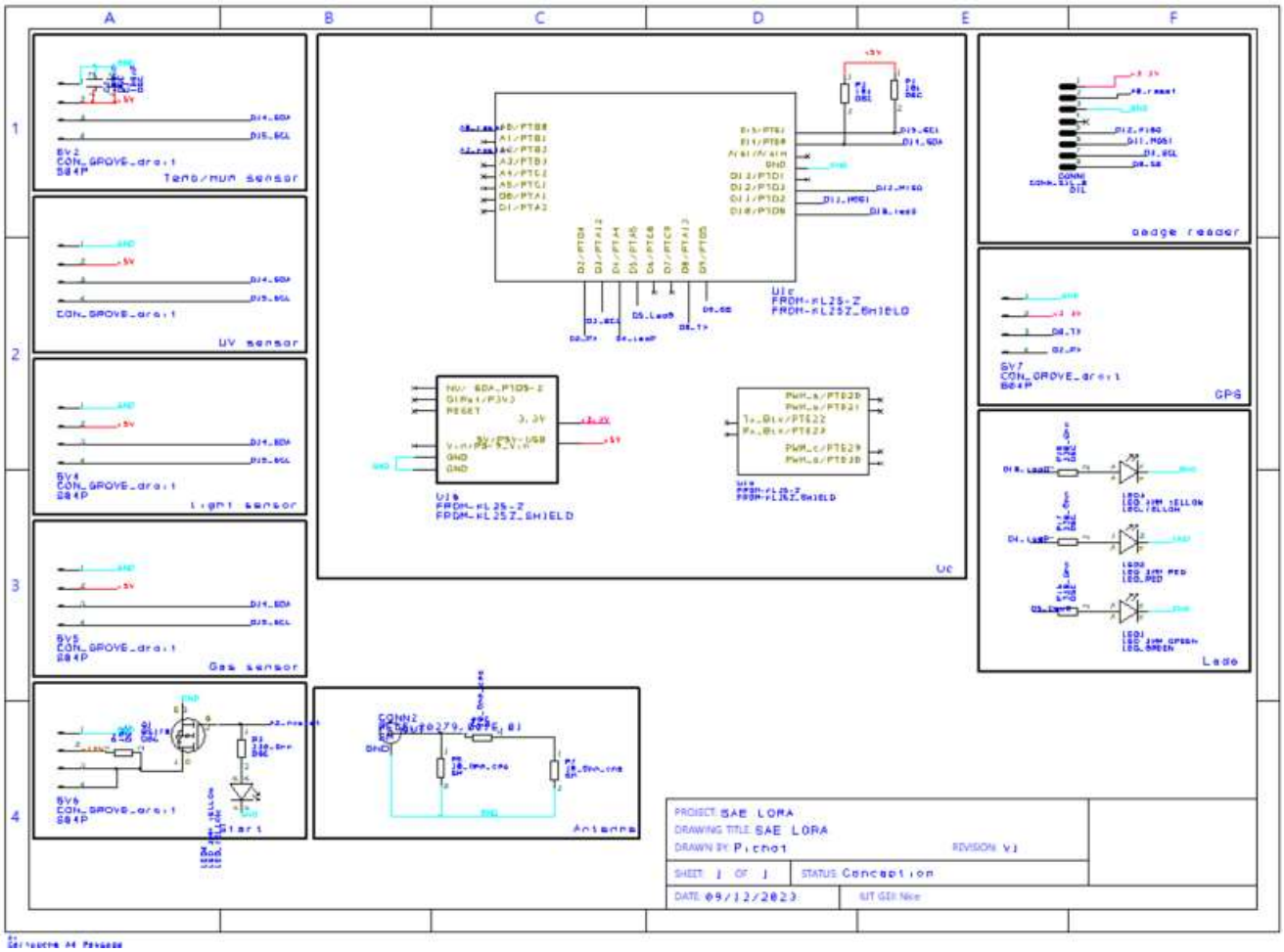
Cette sae est un super parallèle à ce chantier.

7. Annexe

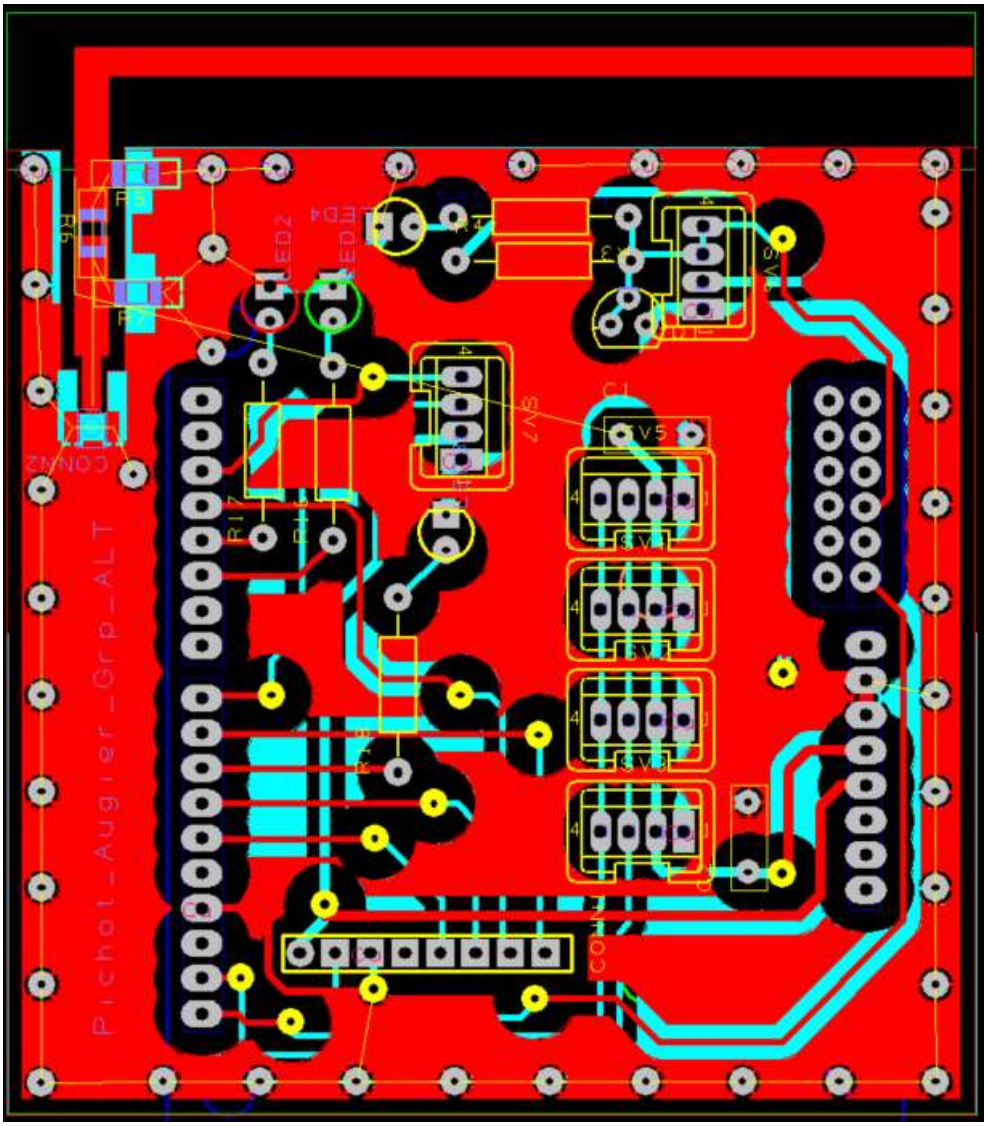
Annexe 1 :



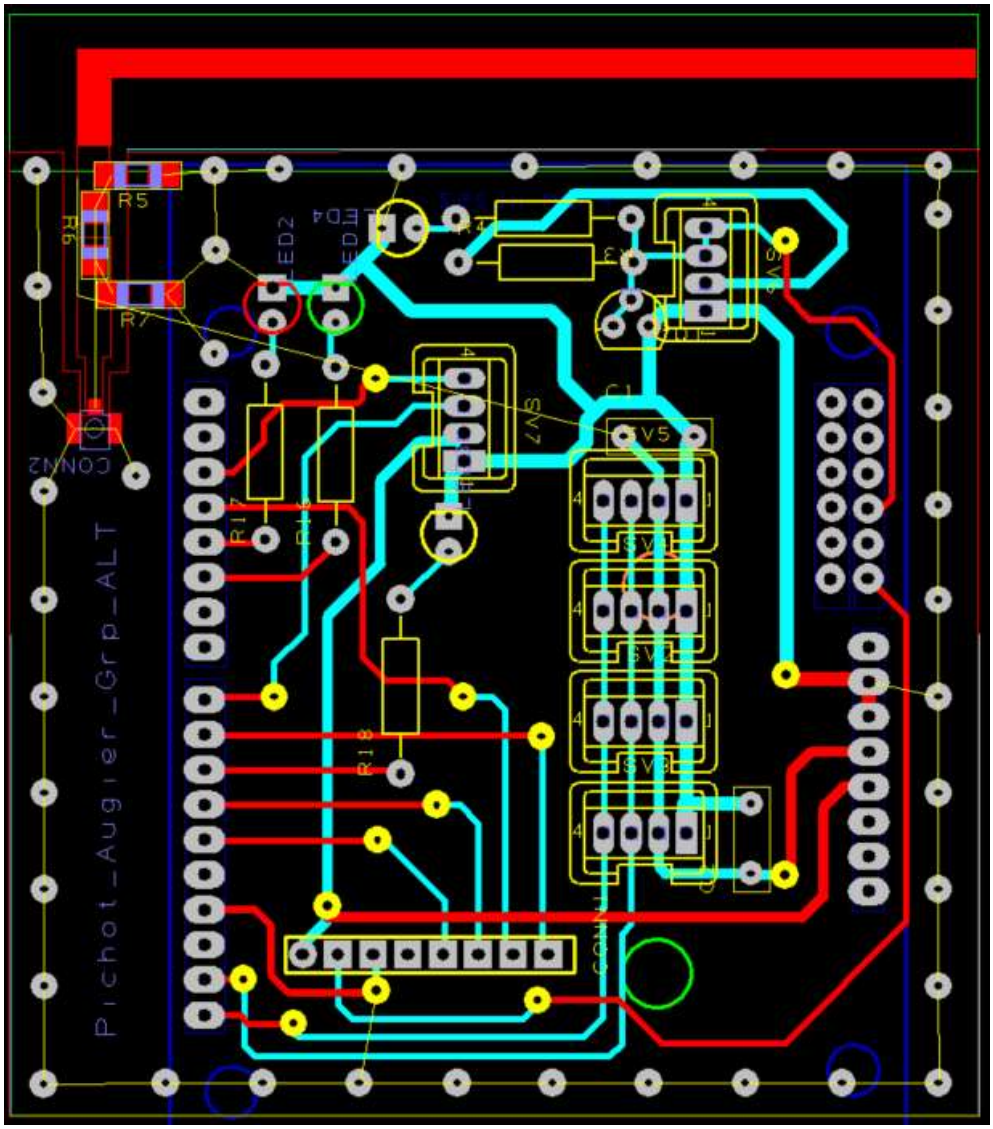
Annexe 2 :


A
B
C
D
E
F
1
2
3
4
Temp/Hum sensor
UV sensor
Light sensor
Gas sensor
Start
Antenna
Uc
GPS
Leds
gauge reader
PROJECT SAE LORA
DRAWING TITLE SAE LORA
DRAWN BY Pichot
REVISION V1
SHEET 1 OF 1
STATUS Conception
DATE 09/12/2023

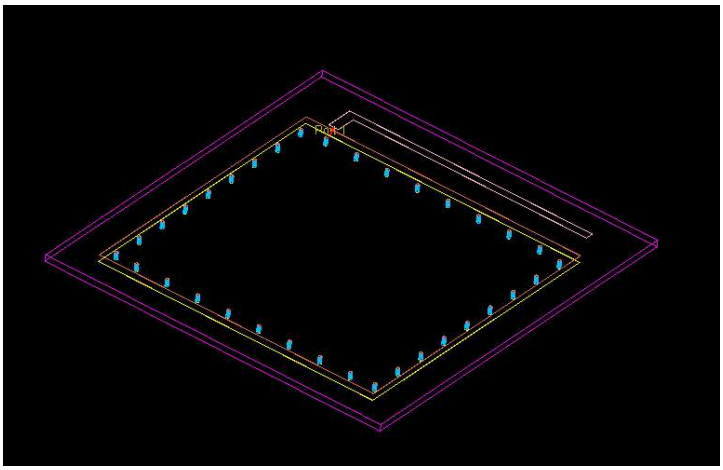
Annexe 3 :



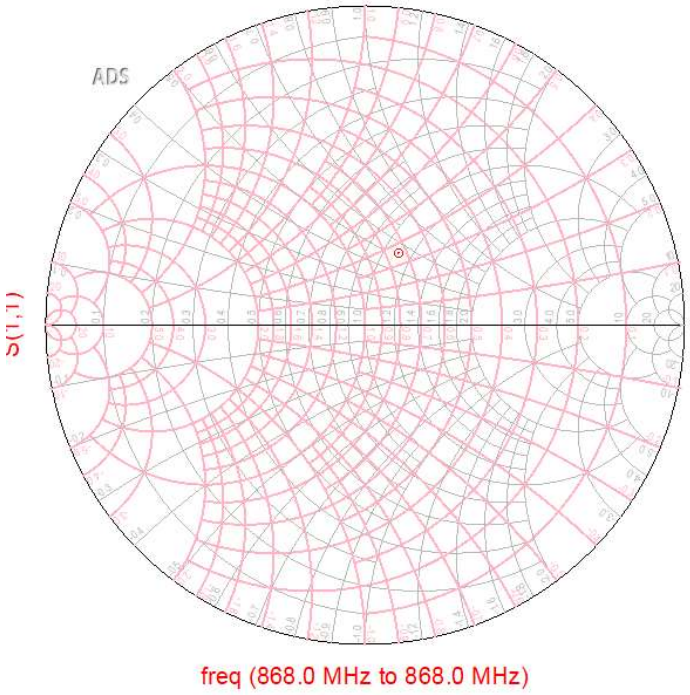
Annexe 4 :

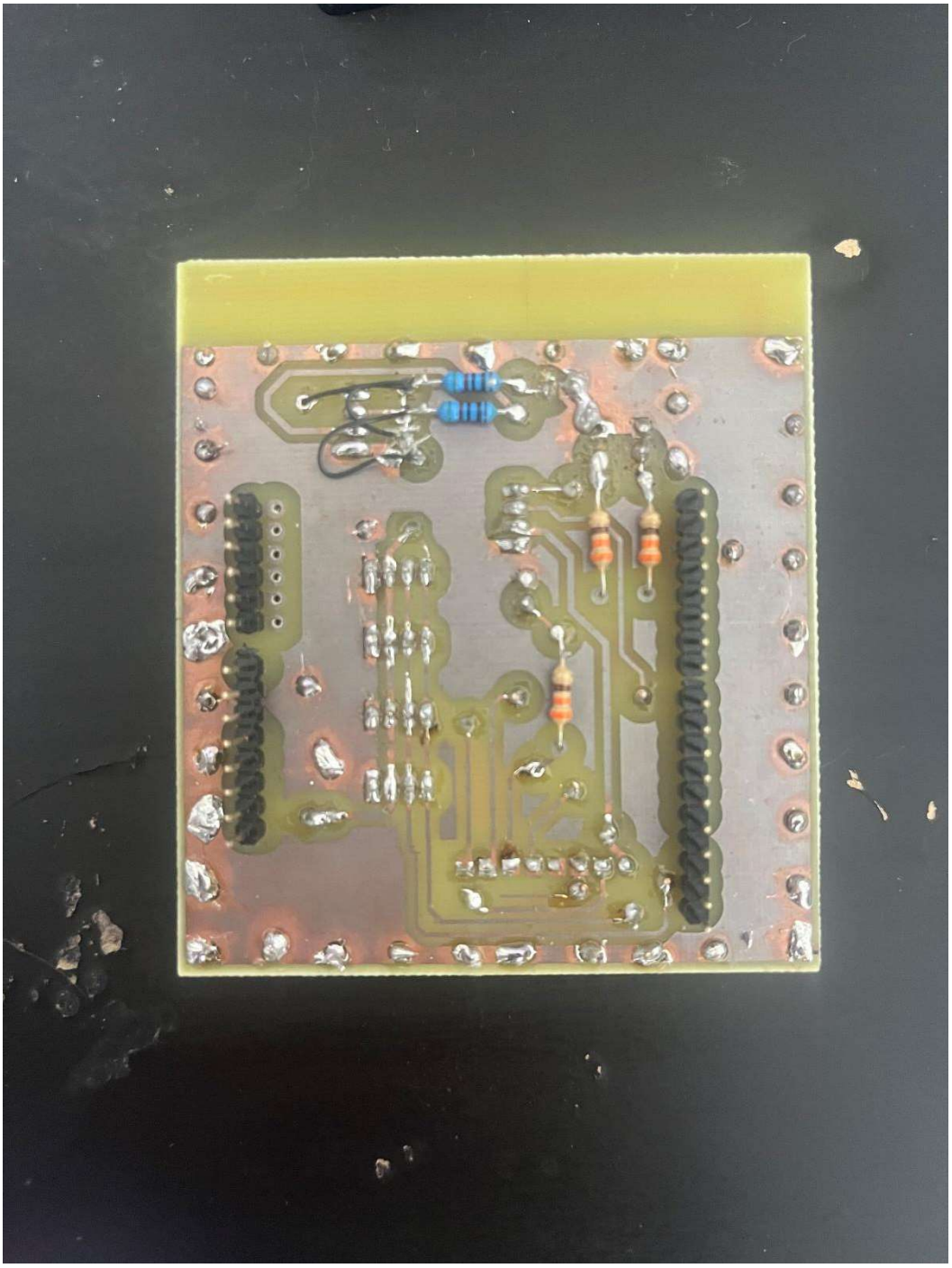

R5
R6
R7
LED2
LED4
R4
SV7
SV5
SV1
SV2
SV3
SV4
R17
R18
CONN2
Pichot_Augier_Grp_ALT

Annexe 5 :



Annexe 6 :





Downlink : 154100B5521914981452314984532195742674EE9C7989465252476 20A0000090900096724365896