Compte Rendu de Projet

Table des matières

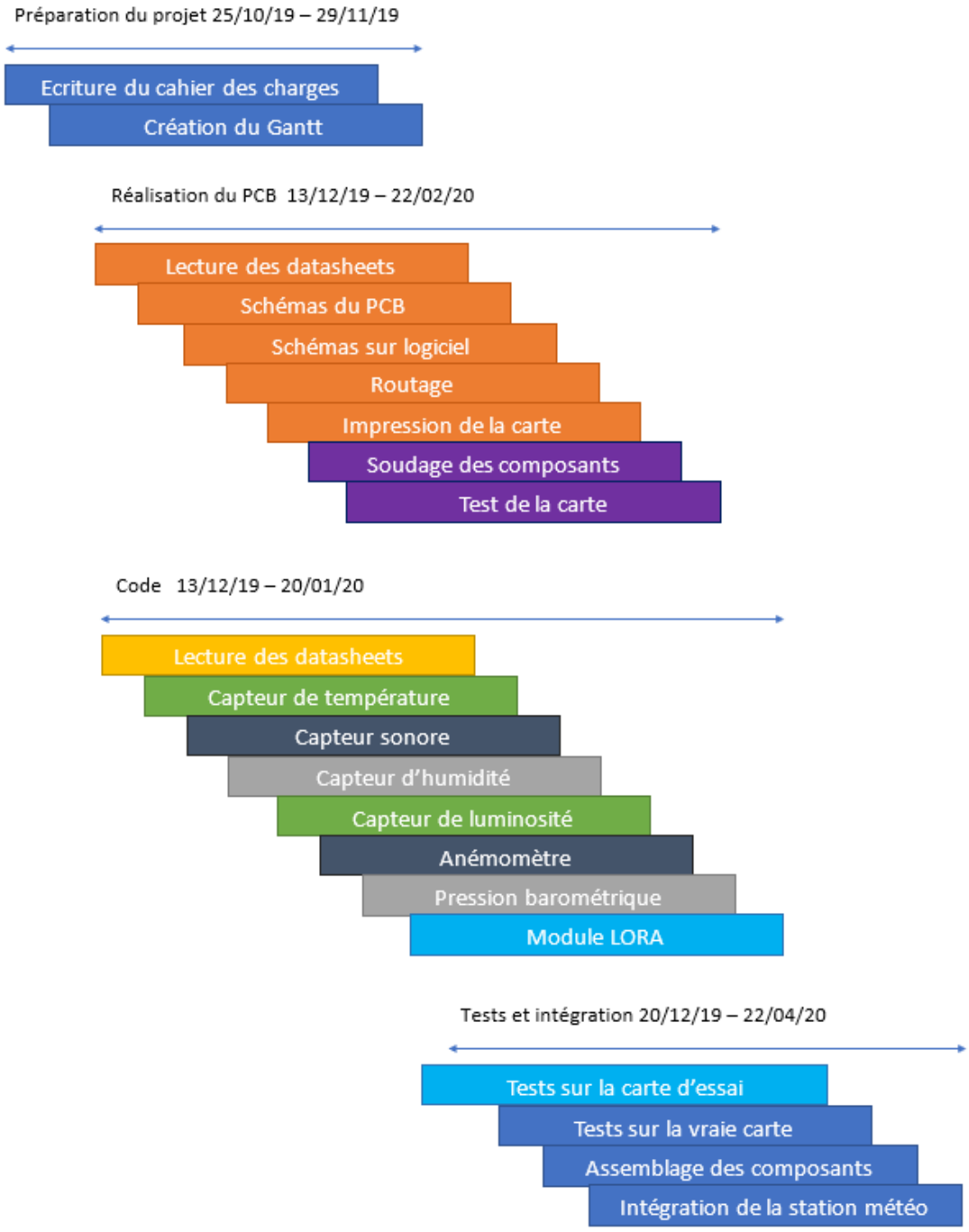
Contexte

M. Peter souhaitait optimiser la récolte de son miel et d'améliorer les conditions de vie des abeilles dans ses 12 ruches. Le but principal est d'améliorer la récolte du miel et les conditions de vie des abeilles. C'est pourquoi nous avons décidé de faire une ruche connectée. Cette ruche sera composée de 5 modules :

- Module interne
- Module externe
- Module météo
- Gateway
- Varroas

Notre groupe a choisi de faire le module météo. Ce module doit permettre le relevé automatique toutes les 15 minutes de différentes informations. Notre module doit aussi pouvoir centraliser des données et communiquer les différentes mesures relevées au module Gateway pour pouvoir les stocker dans la base de données.

Gantt


Préparation du projet 25/10/19 – 29/11/19
Ecriture du cahier des charges
Création du Gantt
Réalisation du PCB 13/12/19 – 22/02/20
Lecture des datasheets
Schémas du PCB
Schémas sur logiciel
Routage
Impression de la carte
Soudage des composants
Test de la carte
Code 13/12/19 – 20/01/20
Lecture des datasheets
Capteur de température
Capteur sonore
Capteur d'humidité
Capteur de luminosité
Anémomètre
Pression barométrique
Module LORA
Tests et intégration 20/12/19 – 22/04/20
Tests sur la carte d'essai
Tests sur la vraie carte
Assemblage des composants
Intégration de la station météo
Tout le monde
Maxime
Samuel
Manon
Louis
Maxime + Samuel
Manon + Louis
Samuel + Louis + Manon

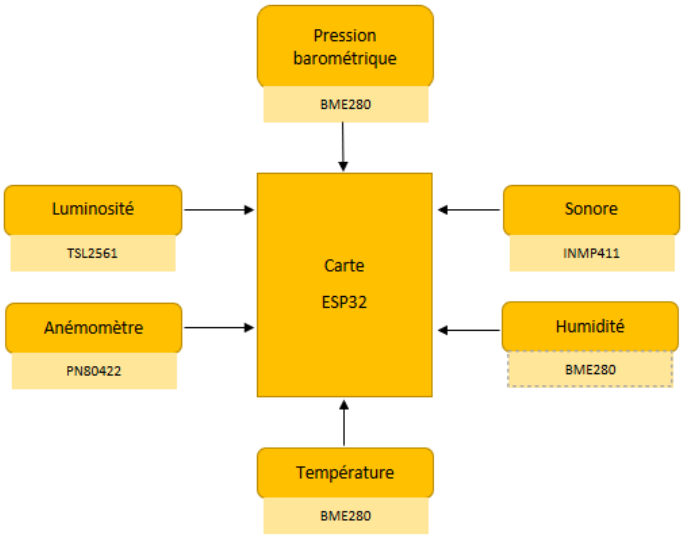
Présentation de la station

Le module météo doit pouvoir relever des informations des capteurs suivants :

- Indicateur hygrométrique
- Température
- Son (détection frelon)
- Pression barométrique
- Luminosité
- Pluie
- Direction du vent
- Vitesse du vent

Nous avons choisi les capteurs suivants :

Capteur	Référence
Luminosité	TSL2561
Anémomètre	PN80422
Température	BME280
Humidité	BME280
Sonore	INMP411
Pression barométrique	BME280



Câblage

Nous avons d'abord commencé par étudier les différents capteurs, cités dans la section précédente, que nous avons utilisé sur notre module météo. Pour cela, nous avons récupéré les datasheets des différents capteurs sur la base de données fourni par M. Peter et sur internet.

Par la suite nous avons réalisé, à l'aide du logiciel Kicad, des schémas de câblage individuel de chaque capteur.

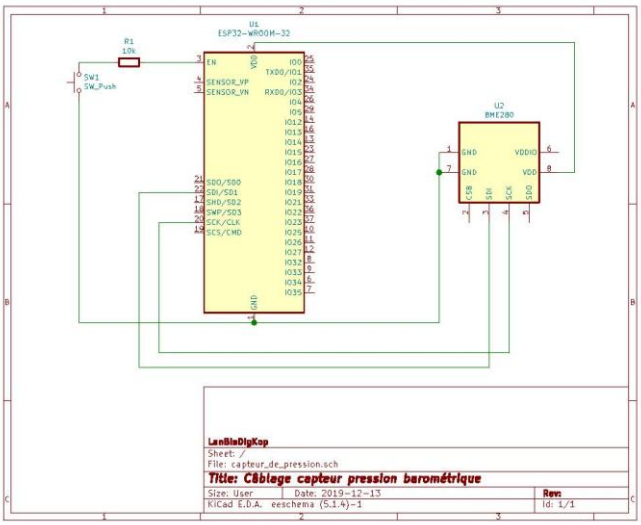


Schéma 1: Capteur de pression

Une fois les différents schémas réalisés et validés, nous avons reçu les templates des fichiers de schéma et du PCB de la carte. Ces templates ont été réaliser avec le logiciel Eagle, il nous a fallu utiliser ce logiciel pour compléter le schéma complet.

Pendant la réalisation du schéma de câblage complet, nous avons rencontré quelques difficultés qui nous ont fait perdre du temps.

L'une des principales difficultés rencontrées était l'utilisation du logiciel Eagle, en effet dans nos cursus antérieurs nous avons tous utilisé des logiciels différents, il a fallu donc apprendre à utiliser ce logiciel. De plus, certains capteurs n'était pas présent dans les librairies de composant de base du logiciel, il a fallu donc les créer.

Pour le raccordement des différents capteurs au module ESP 32, nous nous sommes concertés pour définir les différentes entrées/sorties que nous utiliserons dans les différents programmes qui piloteront notre module.

Une fois toutes ces étapes de franchies, nous avons pu avoir un schéma complet regroupant tous nos composant pour ce module.

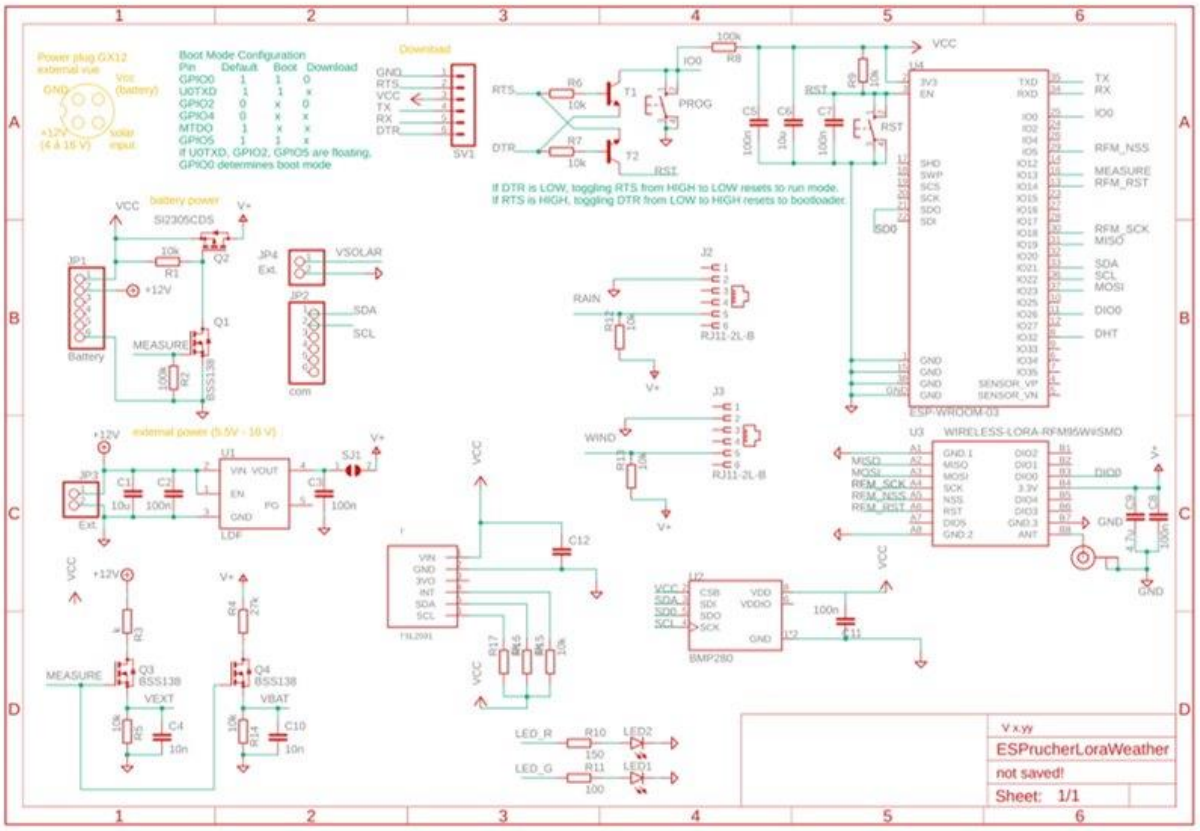


Schéma 2: Module météo

Notre carte se compose essentiellement de petits capteurs, nous avons donc décidé avec M. Peter de connecter ces capteurs sur des borniers qui eux sont reliés sur le module ESP32.

Ci-dessous, un tableau récapitulatif des capteurs utilisés ainsi que leurs alimentations et moyen de communication avec l'ESP 32.

	RFM95W	TSL2591	BMP280	RJ11	INMP411
TENSION	3.3V	3.3V	3.3V	3.3V	3.3V
COMMUNICATION	SPI	I2C	I2C	X	X

Conception de la carte

Pour la conception du PCB de notre module météo, nous avons reçu un template de la carte avec certains composants déjà positionner, notamment le module RFM Lora qui est déjà relier à l'antenne, ainsi que le module ESP 32 et certains composant. Le reste des composants était libre d'être positionner.

Cependant, comme nous avons des borniers qui nous servent à nous connecter à l'extérieur de la carte, il est d'un bon sens de les placer sur les bords de la carte. De plus nous avons deux capteurs qui communiques à l'aide de port RJ11, l'anémomètre et la girouette, il est donc logique de les placer également sur les bords de notre carte.

Enfin, nous avons deux boutons poussoirs pour le téléchargement et le reset du programme dans notre module. Ces deux boutons doivent être facilement accessible, c'est pour quoi nous les avons éloignés des capteurs qui peuvent prendre plus d'espace et nous empêcher de les atteindre facilement.

Ci-dessous une image de notre carte pour le module météo :

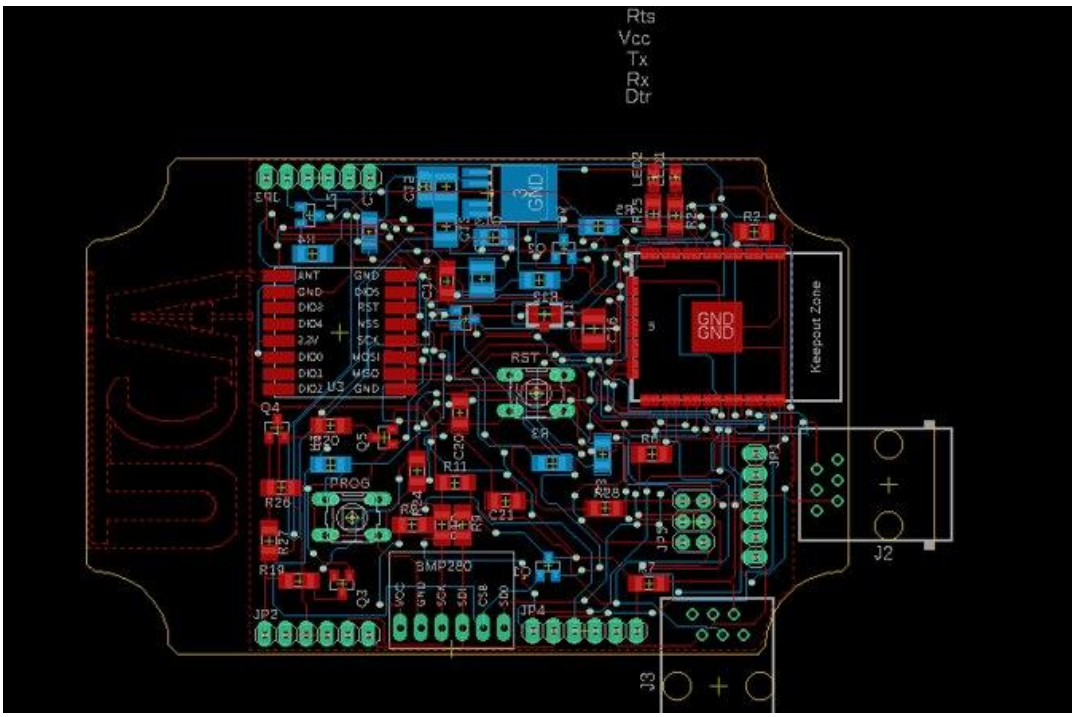


Schéma 3: PCB de la carte pour le module météo

Notre PCB respecte la taille de la carte fournie par le template soit 321.22x95.94 mm

D'après le logiciel Eagle nous avons utilisé une largeur de piste de 0.1524 mm. Nous ne sommes pas sûr de cette dimension car elle nous semble trop petite.

Même si tous les composants sont présents, nous avons un trop grand nombre de vias, ce qui est dû à la disposition de certains composants. Malgré ce nombre important de vias, la carte devrait être fonctionnelle, malheureusement avec cette crise sanitaire, nous n'avons pas pu imprimer notre carte et tester les différentes connections entre les composants.

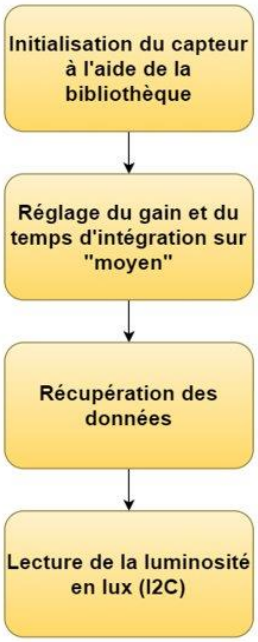
Ci-dessous un tableau pour résumer les atouts et les inconvénients de notre carte.

ATOUTS	INCONVENIENTS
Respecte la forme de la carte	Nombre de vias (155)
Tout les connecteurs aux bords de la carte	Disposition à retravailler
Respect des contraintes pour l'emplacement de l'antenne	

Programmation

Capteur de luminosité

Le capteur de luminosité utilisé pour le projet nous a été imposé. Nous avons utilisé le TSL2591. Il communique avec le microcontrôleur via le bus I2C. Nous avons donc câblé le capteur sur les PINs I2C de notre µC ainsi que les câbles d'alimentations en 3V et le GND. En sortie, le capteur de luminosité nous permet d'avoir un affichage de la luminosité ambiante en Lux.



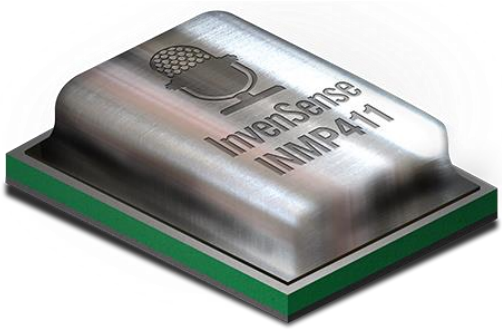
Capteur de température, humidité et pression

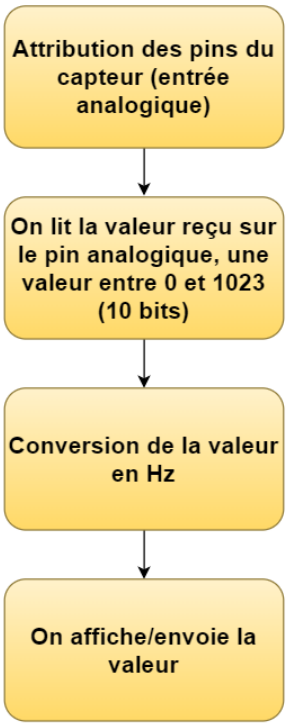
Le capteur que nous utilisons ensuite est le capteur de température, d'humidité et de pression. Nous avons utilisé le capteur BME280. Ce capteur communique aussi avec le microcontrôleur via le bus I2C. Nous avons donc relié les PINs I2C à celles de notre µC ainsi que les PINs d'alimentation. Ce capteur est très pratique car il nous permet l'affichage de 3 valeurs différentes via un seul et même capteur. En sortie, nous avons l'affichage de la température ambiante en °C, de la pression barométrique en hPa et l'humidité présente dans l'air en %. Nous avons dû calibrer ce capteur en abaissant la valeur de la température de 1°C pour avoir une valeur réelle.

Capteur sonore

Le capteur sonore est un microphone qui va enregistrer le bruit des insectes qui entrent dans la ruche afin de s'assurer qu'il s'agisse bien d'abeille et non de frelon. On branche ce capteur sur un pin analogique de notre esp32 pour récupérer une valeur sur 10 bits. Une conversion est nécessaire pour obtenir une valeur en Hz.

Le code n'a pas pu être réalisé car nous n'avons pas ce capteur.





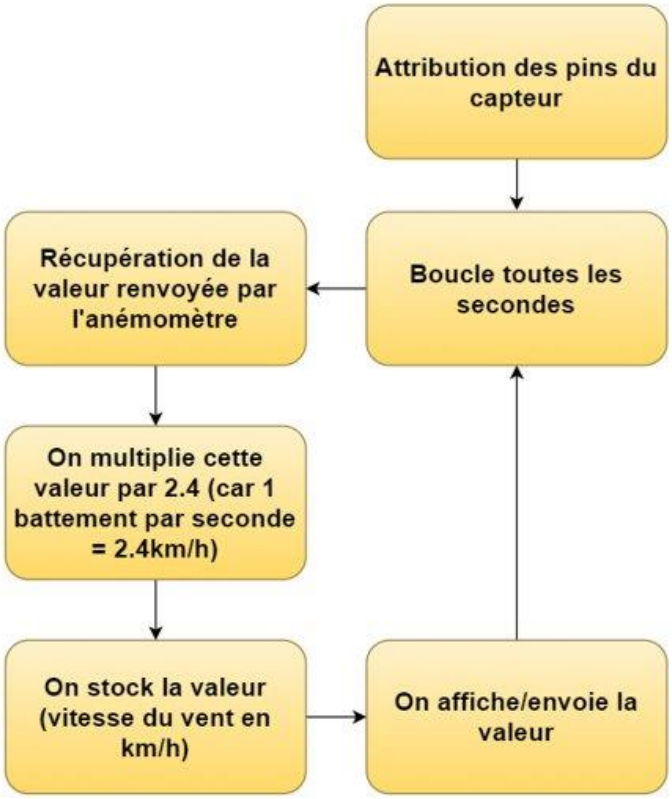
Anémomètre

L'anémomètre sert à mesurer la vitesse du vent. On récupère cette information grâce à un interrupteur à l'intérieur de l'anémomètre qui va « battre » à une certaine vitesse en fonction du vent. La datasheet nous dit qu'à 1 clappement par seconde, la vitesse du vent est de 2.4km/h.

Le mieux est je pense de faire une moyenne, boucler pendant 10 secondes, récupérer la valeur chaque seconde, faire la moyenne de nos 10 valeurs et enfin, la multiplier par 2.4 pour avoir la vitesse du vent en km/h.

Pour le câblage, on avait un adaptateur RJ11 pour le relier directement à notre plaque d'essai, en revanche, une fois la boite transporter pour le confinement, ces adaptateurs ne se trouvent plus dans la boite.

Le code n'a donc pas pu être réalisé, mais voici un pseudo code de ce qui aurait été réalisé :



Pluviomètre

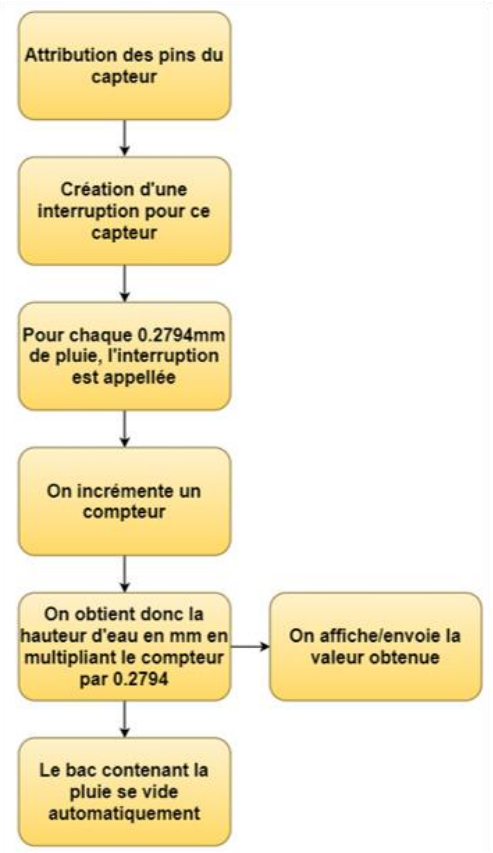
Le pluviomètre sert à mesurer le niveau d'eau lors d'intempérie. Pour le programme, on va devoir faire une interruption. Cette interruption va se déclencher (le pluviomètre envoie une impulsion) à chaque fois que le niveau de l'eau augmente de 0.2794mm (information provenant de la datasheet).

On va donc faire un compteur dans notre interruption, puis, quand on voudra récupérer la hauteur d'eau, on multiplie notre compteur par 0.2794 pour obtenir la hauteur de l'eau en millimètre.

Le bac contenant la pluie se vide automatiquement.

Pour le câblage, on avait un adaptateur RJ11 pour le relier directement à notre plaque d'essai, en revanche, une fois la boite transporter pour le confinement, ces adaptateurs ne se trouvent plus dans la boite.

Le code n'a donc pas pu être réalisé, mais voici un pseudo code de ce qui aurait été réalisé :



Girouette

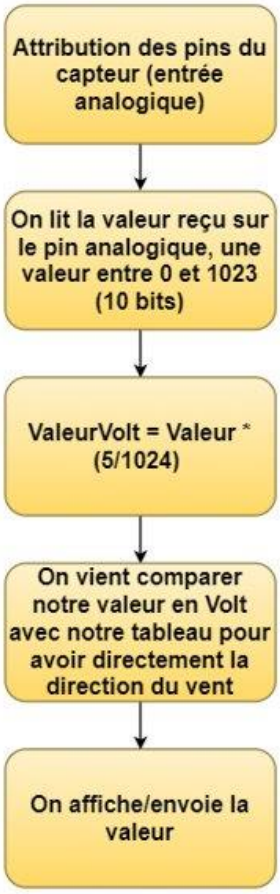
La girouette sert à indiquer la direction du vent. Le tableau ci-dessous nous donne la direction du vent en fonction de la tension mesurée. Dans la datasheet, on retrouve un tableau similaire mais avec des valeurs différentes pour la tension. On a refait ce tableau avec V=3.3V et R=10kOhm car ça correspond mieux à notre projet.

On va donc brancher notre girouette sur une entrée analogique, récupérer la valeur entre 0 et 1023 (10 bits) et la multiplier par 5/1024 pour obtenir la tension. Enfin, il n'y a qu'à comparer avec les valeurs du tableau pour connaitre la direction du vent.

Pour le câblage, on avait un adaptateur RJ11 pour le relier directement à notre plaque d'essai, en revanche, une fois la boite transporter pour le confinement, ces adaptateurs ne se trouvent plus dans la boite.

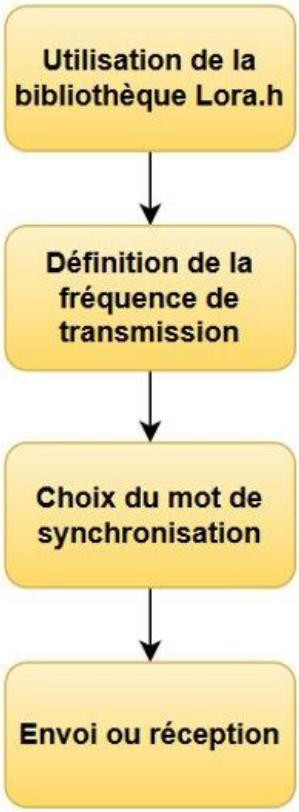
Le code n'a donc pas pu être réalisé, mais voici un pseudo code de ce qui aurait été réalisé :

Direction (Degrees)	Résistance (Ohms)	Voltage (V=3,3V, R=10k)
0	33000	2,53255814
22,5	6570	1,308449004
45	8200	1,486813187
67,5	891	0,269975209
90	1000	0,3
112,5	688	0,21242515
135	2200	0,595081967
157,5	1410	0,407800175
180	3900	0,925899281
202,5	3140	0,788584475
225	16000	2,030769231
247,5	14120	1,931840796
270	120000	3,046153846
292,5	42120	2,666845741
315	64900	2,85941255
337,5	21880	2,264868256



Module de communication

Pour échanger les informations et les envoyer à la base de données, nous utilisons le protocole LORA grâce au module RFM95W. Ce module communique avec notre µC via le protocole SPI. Nous avons donc câblé toutes les PINs du module à celles de notre µC ainsi que les PINs d'alimentations. Pour pouvoir utiliser le protocole LORA dans notre environnement nous avons du choisir la fréquence d'émission européenne en 868 MHz. Nous n'avons pas pu tester le module LORA sur notre projet car nous n'avions pas de récepteur LORA. Nous avons néanmoins fourni notre code à M. Peter qui en changeant les caractéristiques propres à sa configuration a pu tester notre programme et nous assurer de son bon fonctionnement.



Conclusion

Après 64h de projet et malgré la situation du coronavirus, nous avons réalisé le PCB du module météo sur Eagle, nous avons écrit tous les codes des capteurs et modules utiles au bon fonctionnement du module météo.

Nous pouvons voir ci-dessous un résumé des actions réalisées sur les capteurs :

Capteur/Module	Pseudo-code	Code fonctionnel	Test sur breadboard
Luminosité	OK	OK	OK
Pluviomètre	OK		
Anémomètre	OK		
Girouette	OK		
Température	OK	OK	OK
Humidité	OK	OK	OK
Sonore	OK		
Pression barométrique	OK	OK	OK
LORA	OK		

Nous avons observé qu'il nous manquait du matériel et qu'il été plus difficile de travailler en groupe à distance :

- Seulement une personne du groupe possédait l'équipement
- Nous avons écrit les codes du pluviomètre, de l'anémomètre, de la girouette et du LORA, cependant nous ne pouvons pas vérifier le bon fonctionnement de ces codes car nous n'avons pas d'adaptateur RJ45 pour tester les codes sur la carte ni de récepteur LORA.
- Avant le confinement il nous manquait le capteur sonore, c'est pourquoi nous n'avons toujours pas fait et testé le code de ce capteur.
- De plus, nous n'avons pas pu imprimer le PCB, donc tous les tests n'ont été fait que sur la breadboard.

Pour toutes ces raisons nous n'avons pas pu terminer notre projet, car il nous aurait fallu avoir à notre disposition le matériel qui nous manquait pour pouvoir enfin tester tous les composants en même temps sur le PCB imprimé.

Annexe

Code complet (capteur de lumière, température, humidité, pression + module de communication Lora)

```
#include <Wire.h>
#include <SPI.h>
#include <LoRa.h>
#include "cactus_io_BME280_I2C.h"
#include "Adafruit_TSL2591.h"
#include "images.h"

//Définitions des PINS
#define SCK     18    // GPIO5  -- SX1278's SCK
#define MISO    19   // GPIO19 -- SX1278's MISnO
#define MOSI    23   // GPIO27 -- SX1278's MOSI
#define SS      5   // GPIO18 -- SX1278's CS
#define RST     14   // GPIO14 -- SX1278's RESET
#define DI0     2   // GPIO26 -- SX1278's IRQ(Interrupt Request)
//Définition de la bande de fréquence LORA sur la badande européene 868 Mhz
#define BAND  868E6

/*    TLS2591 WIRING    */
// connect SCL to I2C Clock (PIN 36 on ESP32)
// connect SDA to I2C Data  (PIN 33 on ESP32)
// connect Vin to 3.3V
// connect GROUND to GND

/*    BME280 WIRING    */
// connect SCL to I2C Clock (PIN 36 on ESP32)
// connect SDA to I2C Data  (PIN 33 on ESP32)
// connect Vcc to 3.3V
// connect GROUND to GND

BME280_I2C bme(0x76); // I2C using address 0x76
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591);

int t, t0 = 0, deltaT = 2000;
unsigned long Balise_DernierEnvoi = 0; // heure du dernier envoi LoRa
unsigned int counter = 0;

void setup() //initialisation des modules avec initialisation des PINS, de la vitesse de communication
en liaison série
{
 pinMode(2, OUTPUT);
 Serial.begin(9600);
 while (!Serial);
```

```
 Serial.println();
 Serial.println("LoRa Sender Test");

 SPI.begin(SCK, MISO, MOSI, SS);
 LoRa.setPins(SS, RST, DI0);
 if (!LoRa.begin(866E6)) //test pour savoir si le module a réussi à démarrer
 {
  Serial.println("Starting LoRa failed!");
  while (1);
 }

 Serial.println("init ok");
 if (!bme.begin())
 {
  Serial.println("Could not find a valid BME280 sensor, check wiring!");
  while (1);
 }
 bme.setTempCal(-1);// Temp was reading high so subtract 1 degree
 Serial.println(F("Starting Adafruit TSL2591 Test!"));
 if (tsl.begin())
 {
  Serial.println(F("Found a TSL2591 sensor"));
 }
 else
 {
  Serial.println(F("No sensor found ... check your wiring?"));
  while (1);
 }
 tsl.setGain(TSL2591_GAIN_MED);  //Medium gain (LOW for bright light and HIGH for dim light)
 tsl.setTiming(TSL2591_INTEGRATIONTIME_300MS); //Medium intergration time  (100MS for bright
light to 600MS for dim light)
}

void loop()
{
 String Message = "";
 t = millis();
 if (t - t0 > deltaT)
 {
  t0 = t;
  bme.readSensor();

  uint32_t lum = tsl.getFullLuminosity();
  uint16_t ir, full;
  ir = lum >> 16;
  full = lum & 0xFFFF;

  Serial.print("\n");
  Serial.print("Pression en millibars : ");
  Serial.println(bme.getPressure_MB());

  Serial.print("Humidité en % : ");
```

```
  Serial.println(bme.getHumidity());

  Serial.print("Temperature en degre celsius : ");
  Serial.println(bme.getTemperature_C());

  Serial.print(F("Lux: ")); Serial.println(tsl.calculateLux(full, ir));

  //Lora
  if (Serial.available())
  {
    while (Serial.available())
    {
      Message += Serial.readString();
    }
    //Serial.println(Message);
    counter = 0;
    LoRa.beginPacket();
    Serial.print("Moi: ");
    Serial.println(Message);
    LoRa.print(Message);
    //LoRa.print(counter);
    LoRa.endPacket();
  }

  int packetSize = LoRa.parsePacket();   // onregarde si quelque chose est reçu
 }
}
```