

Station météo autonome – description

Octobre 2019

Simon GREGOROWICZ



Description sommaire de la station

La station est équipée d'un capteur BME280 capable de mesurer :

- **la température** (avec une précision de +/- 1°C) ;
- **l'humidité relative** (avec une précision de +/- 3%) ;
- **la pression atmosphérique** (avec une précision de +/- 1hPa).

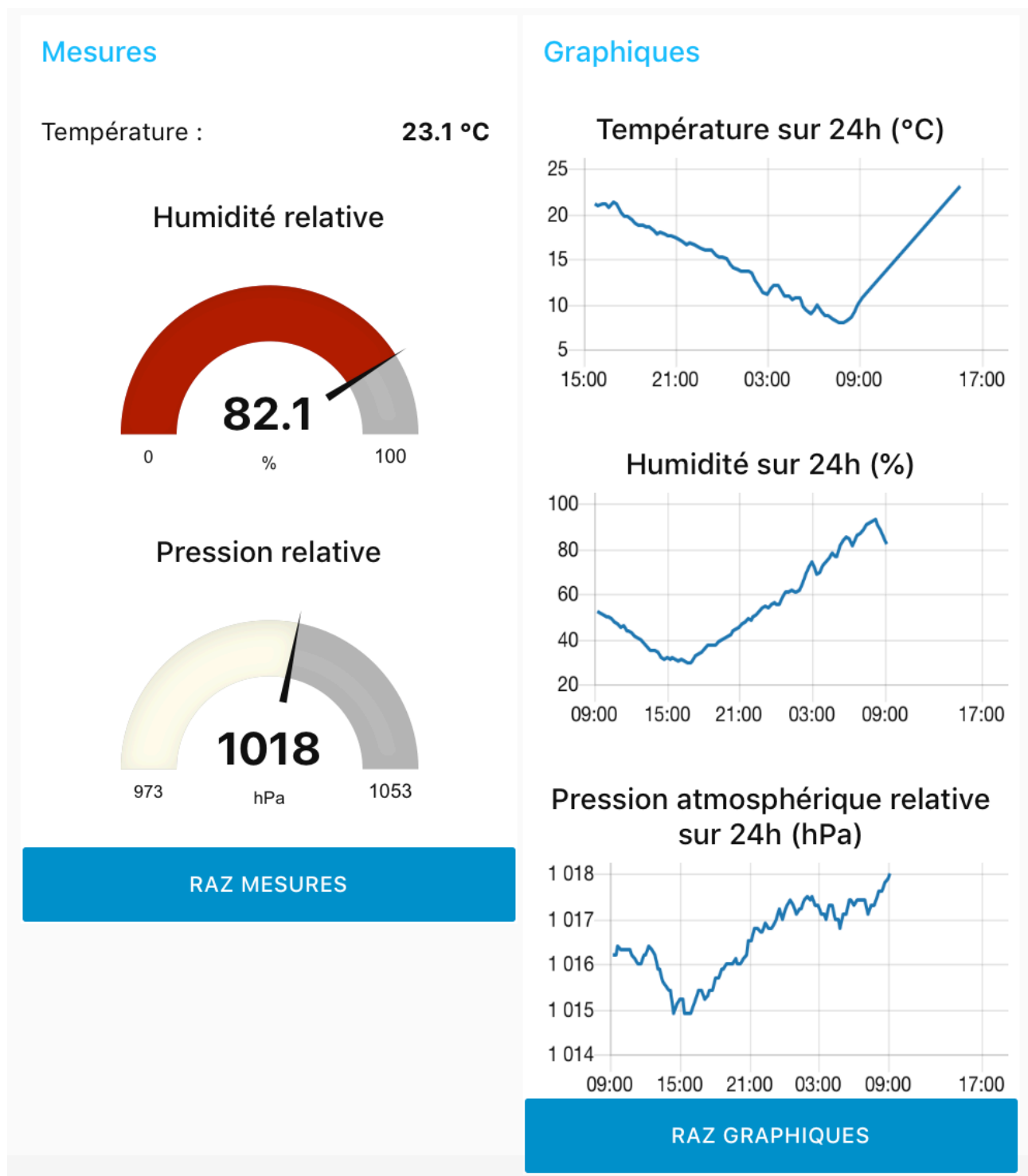
L'expérience montre que ce capteur bon marché fournit une précision suffisante dans le cadre d'un usage amateur, avec des valeurs légèrement par excès. Sa faible consommation électrique fait de lui un très bon candidat pour ce genre d'application.

Ce capteur est connecté en I2C à une **carte Wifi NodeMCU** (processeur ESP8266), programmée en C avec l'IDE Arduino. L'ensemble est alimenté par une **batterie rechargeable Li-Ion 18650 3.7V** d'une capacité de 2500mAh, par l'intermédiaire d'un module **Battery Shield** délivrant une tension de 3.3V. Afin de rendre l'ensemble autonome, le système est complété par un **panneau photovoltaïque polycristallin 6V** délivrant une puissance crête de 2W.

Les mesures sont transmises en Wifi à l'aide du protocole MQTT à un **broker MQTT** installé sur une carte **Raspberry Pi 2**. La gestion de l'interface utilisateur est confiée à **Node-RED**, les données sont consultables avec un simple navigateur (à l'aide d'un ordinateur, d'un téléphone portable, d'une tablette ou autre).

NOTA : Il est tout à fait possible de modifier la partie électronique pour l'adapter à ses besoins : utilisation d'un capteur DHT-22 par exemple, utilisation d'un autre type de carte, transfert des données via ondes radio avec un émetteur 433MHz (ou autre), ...

Les pièces constitutives de la station sont réalisées en **impression 3D**, à l'exception du mât en aluminium et de la visserie en acier inoxydable.

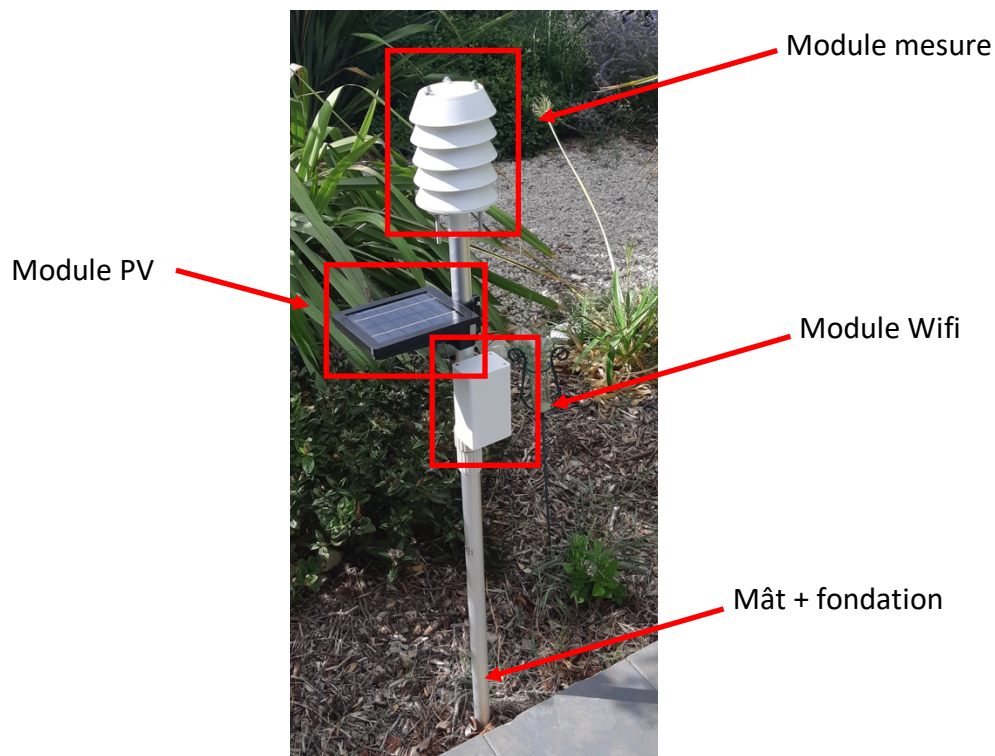


Dashboard Node-RED

Dans ce qui suit, on s'intéressera dans un premier temps à l'**enveloppe de la station** (aspect mécanique), puis dans un second temps à l'**aspect électronique**. Enfin des pistes seront données pour monter un serveur MQTT avec une carte Raspberry et pour configurer Node-RED.

1/ Construction de la station

La station est composée des modules suivants :



Les pièces en impression 3D sont imprimées avec du filament **ASA** (proche de l'ABS), particulièrement adapté pour une utilisation en extérieur.

1.1/ Mât

Un tube de récupération en aluminium a été utilisé, de dimensions suivantes :

- diamètre extérieur **28.5mm** ;
- diamètre intérieur **26mm** ;
- longueur totale **2m**.

A noter qu'il est tout à fait possible d'utiliser un tube de section différente, en adaptant la taille des pièces imprimées en 3D (les fichiers Fusion 360 étant fournis).

La stabilité du mât est assurée en l'encastrant de 50cm dans le sol. Le module mesure est donc positionné à 1.50m de hauteur (hauteur normalisée Météo France).

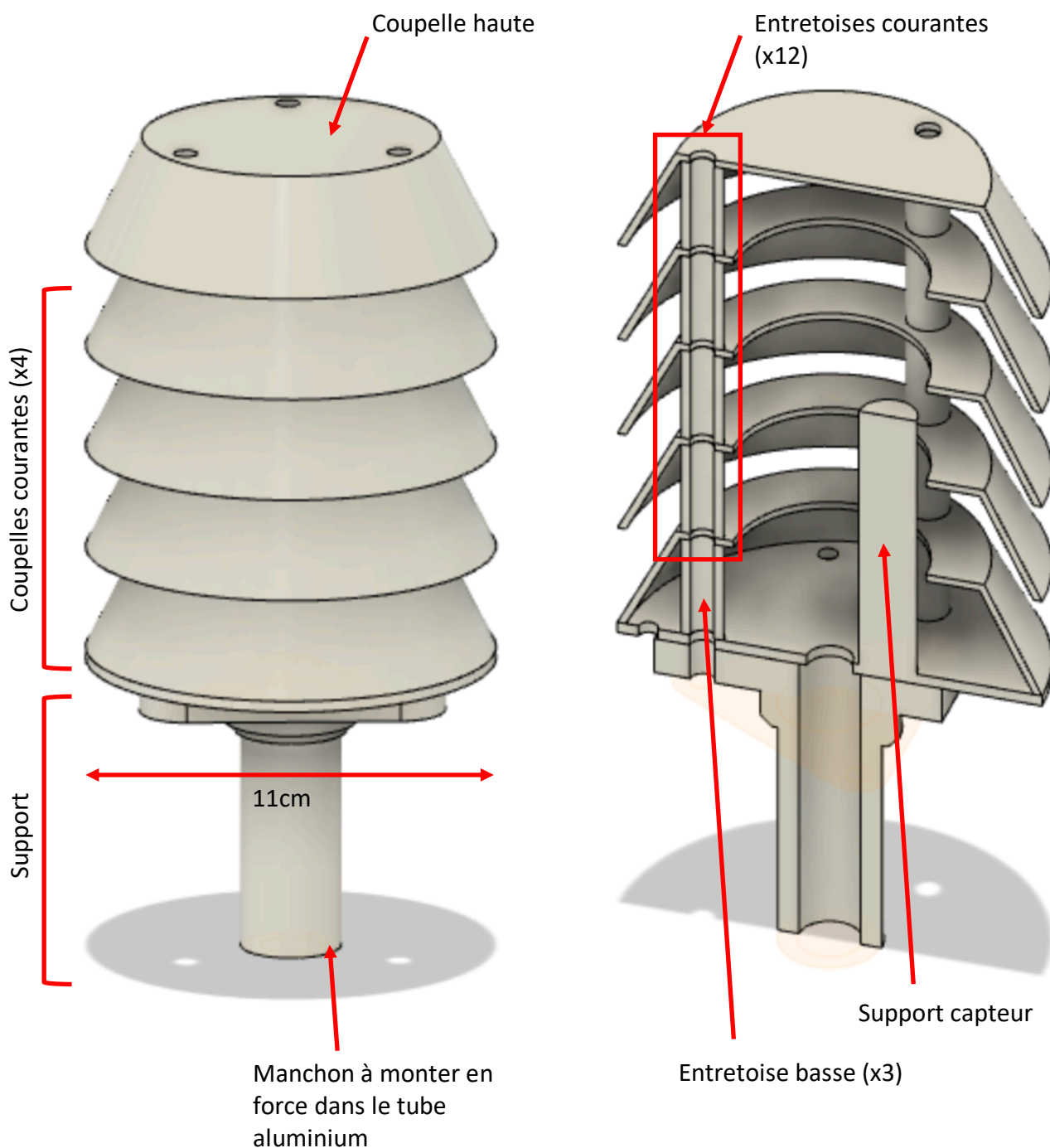
Une alternative possible (permettant de faciliter le déplacement de la station) est de limiter la hauteur du mât à 1.50m, et de l'encastrer dans une fondation en béton. Les calculs de stabilité montrent qu'une semelle rectangulaire simplement posée sur le sol de 33cm x 33cm x 10cm de hauteur est nécessaire, permettant ainsi de résister à une pression de vent de l'ordre de 100daN/m².

Les trous sont réalisés dans le mât au niveau des modules Wifi et PV afin de permettre le passage des fils électriques, logés à l'intérieur du tube (par soucis esthétique et afin de les protéger des agressions du milieu extérieur).

1.2/ Module mesure

Le module mesure accueille le capteur BME280. Compte-tenu de la chaleur dégagée par la carte NodeMCU, et afin de ne pas fausser la mesure de température, il a été fait le choix de déporter la carte dans un module dédié (module Wifi).

Le design du module s'inspire des équipements utilisés par Météo France. Pour plus de détails, consulter le site <http://montpellier.meteo.free.fr/Soucabris/Soucabri.htm>. On se référera avec intérêt à ce site pour le positionnement de la station météo sur le site, compte tenu de sa configuration et des avoisinants.



Le module se compose de **22 pièces simples, très faciles à imprimer**. Comme pour toutes les pièces de la station, **aucun support n'est nécessaire**. L'espacement entre les coupelles est maintenu à l'aide d'entretoises dans lesquelles viennent se loger **3 tiges filetées M6 longueur 20cm en acier inoxydable**. L'ensemble est solidarisé par **6 écrous M6 en acier inoxydable**, à raison de 3 unités en partie supérieure et 3 unités en partie inférieure.

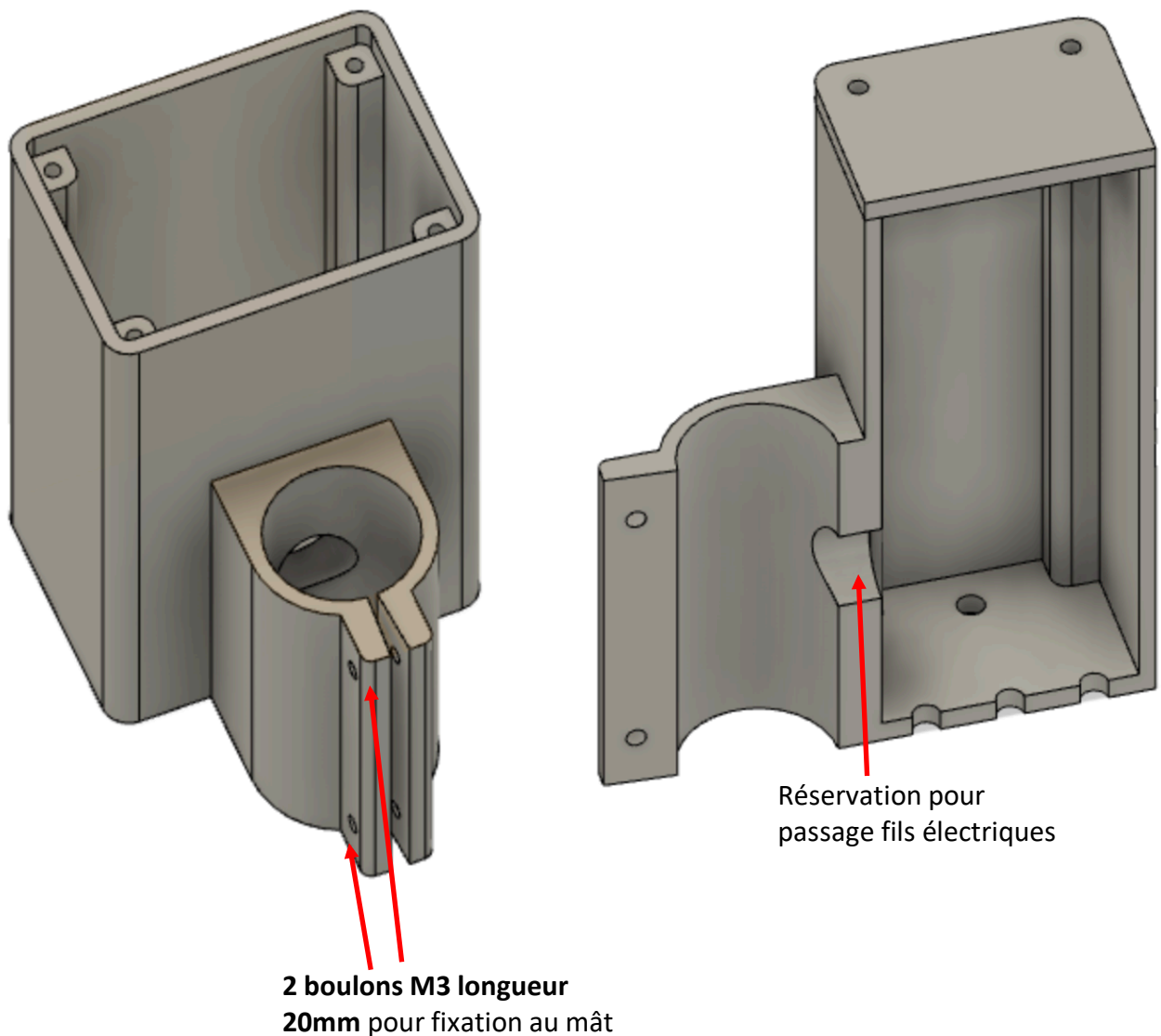
Le manchon est troué en son centre pour permettre le passage des fils électriques.

Le capteur BME280 est positionné sur la tige support à l'aide de liens autobloquants, approximativement à mi-hauteur du module.

Compte tenu du rôle structural de la pièce support, il est conseillé d'imprimer cette pièce avec un remplissage de 40% minimum.

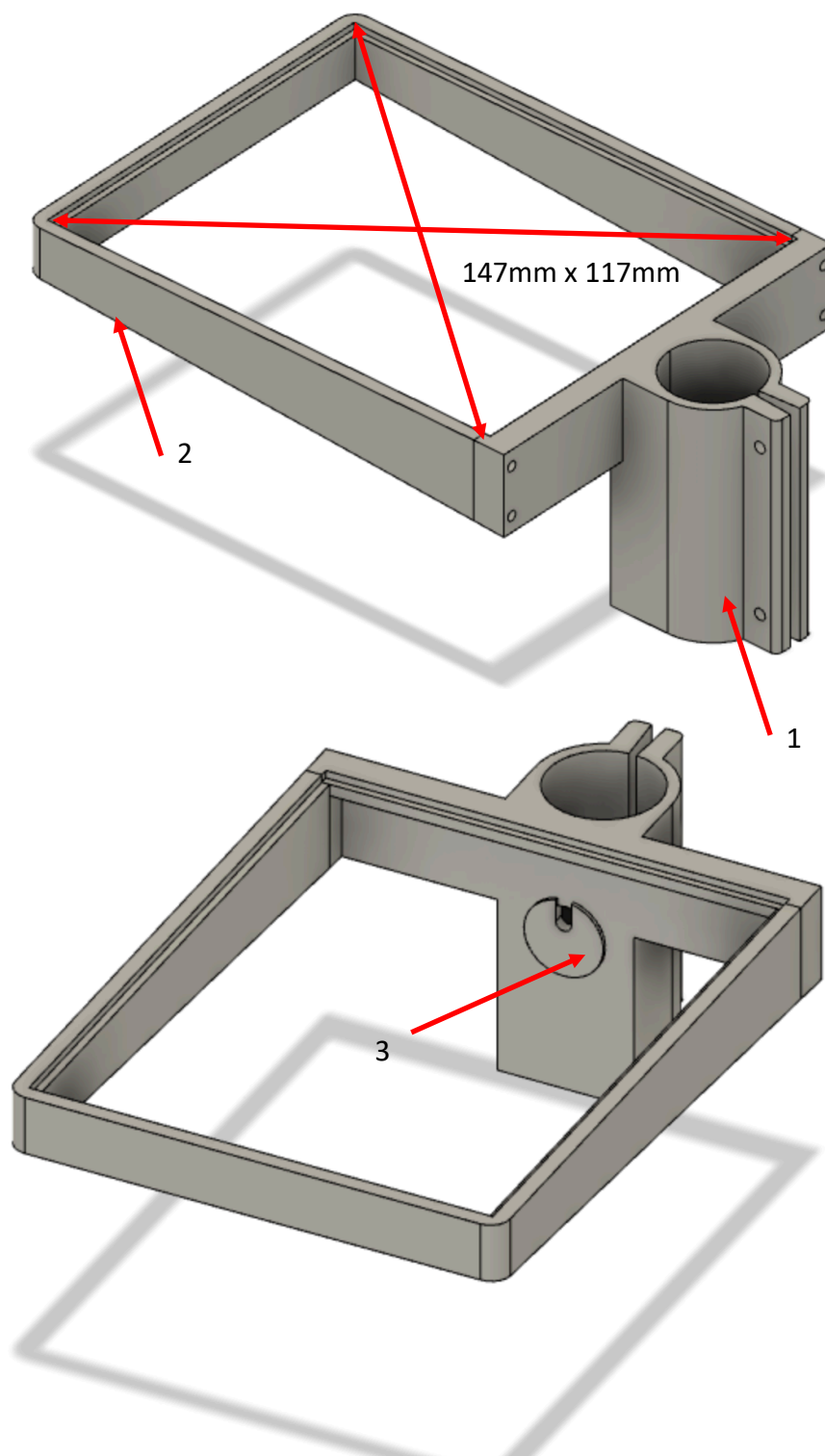
1.3/ Module Wifi

Ce module accueille la carte NodeMCU ainsi que le Battery Shield et sa batterie. Il se compose de deux pièces très faciles à imprimer (la base et le couvercle), solidarisées par **4 vis M3 longueur 20mm en acier inoxydable**.



1.4/ Module PV

Ce module se compose de 3 pièces simples. Les pièces 1 et 2 peuvent être fusionnées si les dimensions du plateau de l'imprimante le permettent. Dans le cas contraire, elles sont reliées à l'aide de **4 vis M3 longueur 20mm en acier inoxydable**. La fixation au mât est en tout point analogue à celle du module Wifi. Le bouchon 3 permet de protéger l'intérieur du mât.



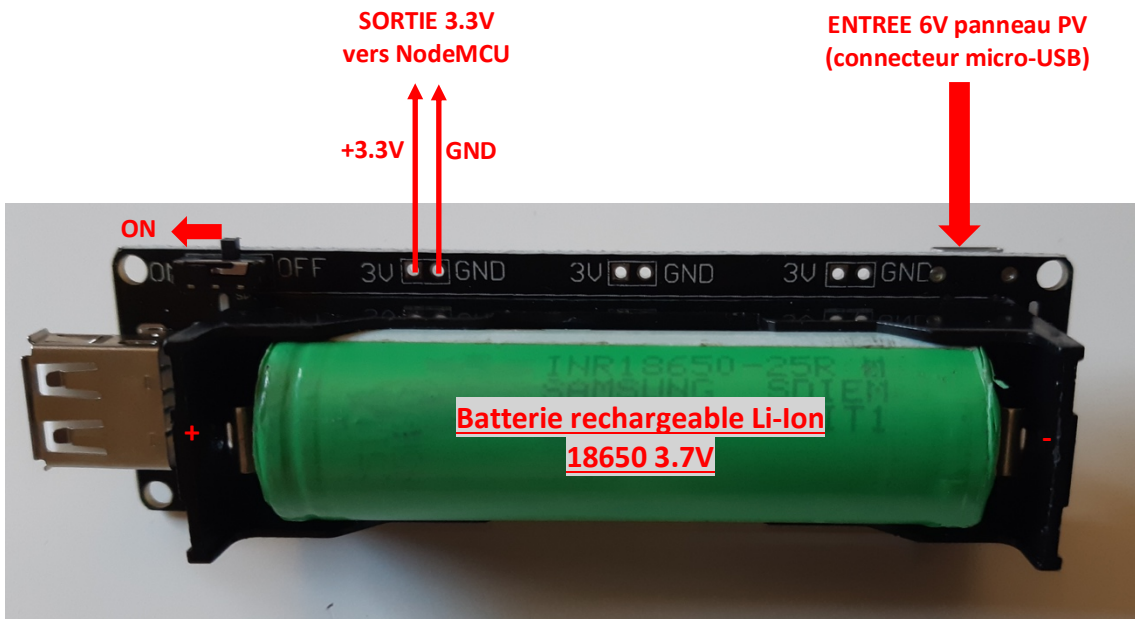
Les dimensions du module peuvent être adaptées en fonction des dimensions du panneau photovoltaïque. Il est conseillé d'imprimer les pièces 1 et 2 avec un remplissage de 40% minimum.

2/ Montage électronique

2.1/ Composants

Battery shield

Ce module est en charge de l'alimentation en 3.3V de la carte NodeMCU et du capteur BME280 à partir d'une batterie rechargeable Li-Ion 18650 3.7V. La batterie est rechargée par le panneau photovoltaïque, connecté au module. Le module gère simultanément l'alimentation de la station et le rechargement de la batterie (passthrough). Il faut prêter attention à **respecter la polarité de la batterie**, certains modules n'étant pas protégés.



Ce type de modules accepte généralement une tension d'entrée de 5 à 8V, compatible avec le panneau photovoltaïque sortant 6V.

Il faut prêter attention à l'intensité maximale du courant de charge acceptée par le module : valant classiquement 0.5A, il faudra choisir un panneau photovoltaïque 6V de puissance crête inférieure ou égale à $P=U*I=6*0.5=3W$. Le panneau de 2W proposé ici convient donc parfaitement.

En 3.3V, le courant de sortie est classiquement de 1A, valeur largement suffisante pour alimenter la carte NodeMCU et le capteur.

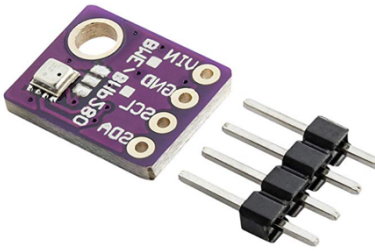
Pour d'autres solutions d'alimentation de la carte NodeMCU, on pourra consulter le site <https://projetsdiy.fr/esp8266-guide-de-choix-achat-projets-diy/>

Batterie rechargeable Li-Ion 18650 3.7V 2500mAh

Une batterie de capacité supérieure (3500mAh par exemple) convient bien entendu également.



Capteur BME280



Le capteur est connecté à la carte NodeMCU et utilise le protocole I2C (ou SPI). La tension d'alimentation du capteur est de **3.3V**. Certaines cartes (Adafruit par exemple) permettent également de l'alimenter en 5V. Dans ce cas, avec le montage proposé, il faut placer le commutateur sur 3.3V. Ce capteur est très économe en énergie (3.6µA en fonctionnement et 0.1µA en veille pour le capteur seul), ce qui le rend intéressant pour des installations autonomes.

Carte NodeMCU

Cette carte possède un module ESP8266, très bien documenté sur internet, permettant de **communiquer en Wifi**.

Le téléversement du programme est aisé avec le port micro-USB (veiller à prendre un câble gérant les données).

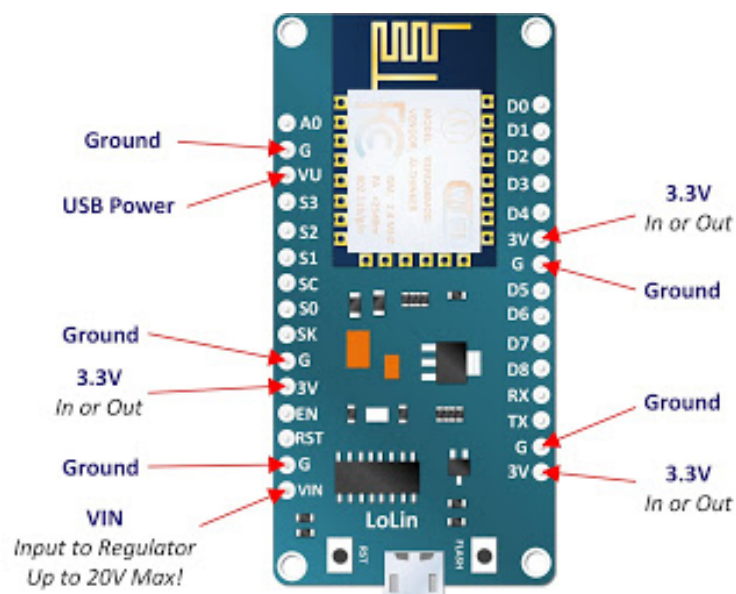
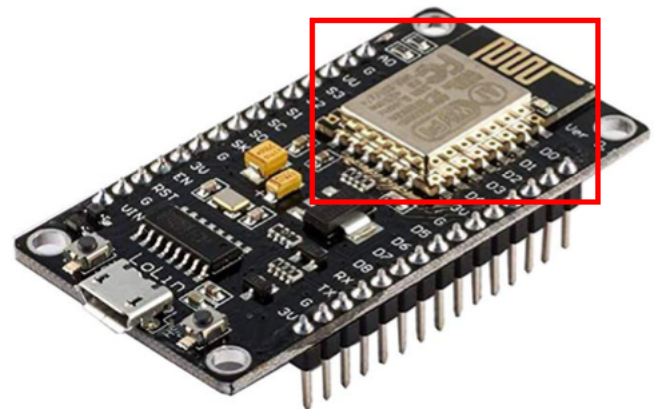
La carte fonctionne en 3.3V (tension utilisée dans le cadre du projet) ou 5V.

La consommation de la carte NodeMCU est assez élevée, particulièrement lorsque le Wifi est utilisé.

NOTA : il est tout à fait possible d'opter pour une autre carte ESP (Wemos D1 mini ou autre).

Plus d'informations sur la consommation électrique des cartes courantes : <http://riton-duino.blogspot.com/2018/12/consommation-dune-carte-arduino.html>

Module ESP et antenne



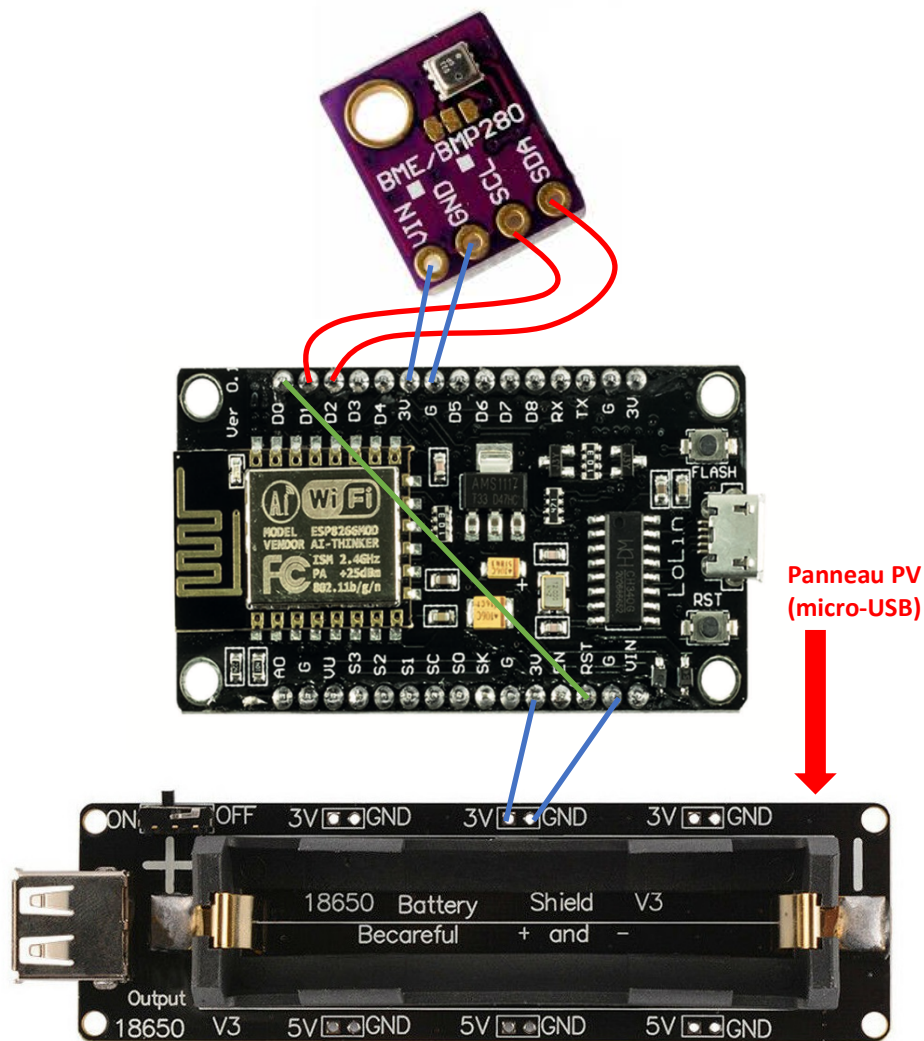
Panneau photovoltaïque



Panneau **6V** délivrant **une puissance crête de 2W**, de dimensions **145mm x 115mm**.

Prévoir en extrémité de câble un connecteur **micro-USB**.

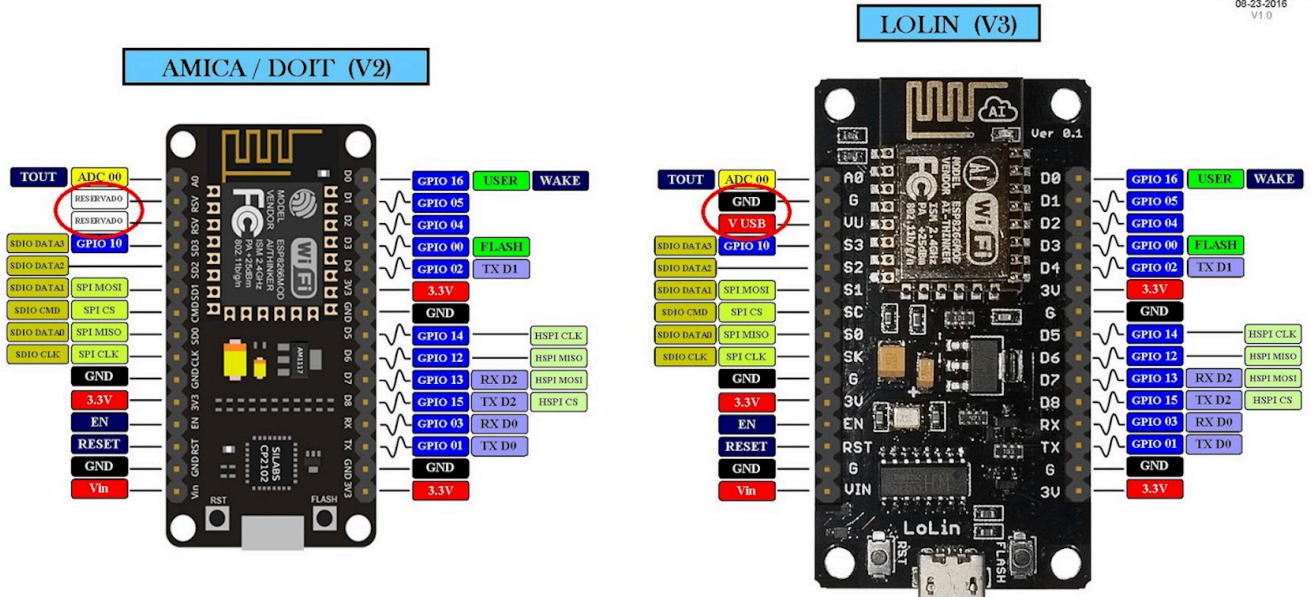
2.2/ Câblage



Le fil reliant DO à RST (en vert ci-dessus) est utilisé pour le redémarrage de la carte NodeMCU après mise en veille. **Lors du téléversement du programme, il est nécessaire de le déconnecter.**

Remarque : avec les cartes type NodeMCU, il faut prêter attention au fait que la numérotation des PINs de la carte ne correspond pas à celle de l'IDE Arduino. Il faut se référer au schéma de correspondance ci-dessous.

PINOUT NodeMCU 1.0



2.3/ Diminution de la consommation électrique du système

En fonctionnement, la carte NodeMCU et le capteur BME280 requièrent un courant d'environ **80mA**. Sans apport par le panneau photovoltaïque, l'autonomie du système serait ainsi d'un jour environ avec la batterie retenue. Cette autonomie est clairement insuffisante. La carte NodeMCU possède toutefois plusieurs modes de veille, dont le mode **deep-sleep**. Dans ce mode, quasiment toutes les fonctions de la carte sont déconnectées et la consommation électrique est beaucoup plus faible.

Item	Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi	OFF	OFF	OFF
System clock	ON	OFF	OFF
RTC	ON	ON	ON
CPU	ON	Pending	OFF

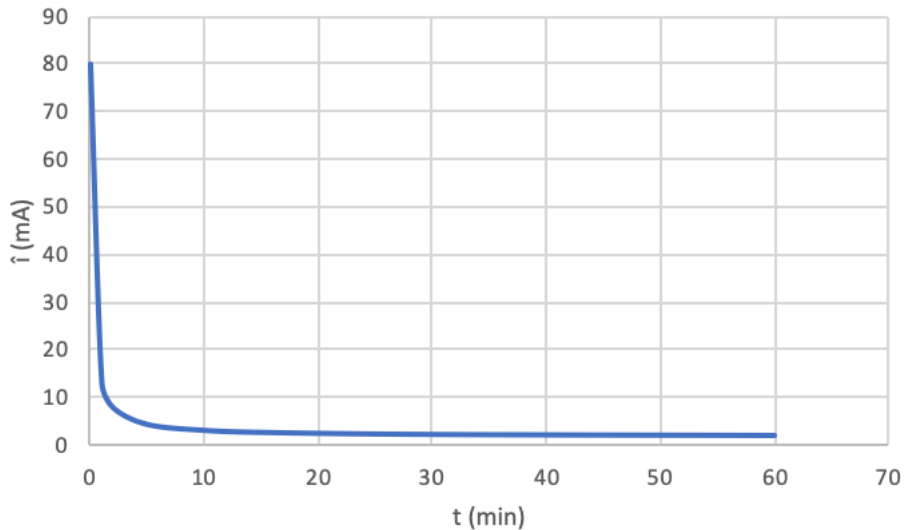
Partant du constat qu'il est inutile d'effectuer des mesures en continu, le moyen le plus simple de rendre le système moins énergivore est d'effectuer des mesures à intervalle régulier, et de mettre la carte en deep-sleep entre les mesures.

En considérant¹ :

- qu'en fonctionnement le système consomme 80mA ;
- qu'en deep-sleep, le système consomme 2mA ;
- que la durée nécessaire à la connexion au réseau Wifi, aux mesures et à l'envoi des données est de 10s ;
- et en notant t (en minutes) la durée de mise en veille entre les mesures.

L'intensité moyenne du courant nécessaire au fonctionnement du système vaut :

$$\hat{i}(t) = (2 \cdot t + 80 \cdot 10/60) / (t + 10/60)$$



On constate qu'en **espacant les mesures de 15 minutes**, la consommation moyenne du système passe à **3mA**. Cette valeur paraît être un bon compromis précision/consommation.

2.4/ Dimensionnement de la capacité de la batterie et du panneau photovoltaïque

Le dimensionnement par le calcul est délicat *à priori*, puisqu'il dépend du rendement des transformateurs du Battery Shield, que l'on ne connaît pas avec précision. Par ailleurs la capacité réelle de la batterie peut être assez éloignée de sa capacité théorique. Enfin il conviendrait de prendre en compte la consommation du Battery Shield à proprement parler.

En première approche, l'autonomie théorique sans apport solaire, à 50% de la capacité de la batterie est d'environ **2 semaines** (avec une batterie de 2500mAh et des mesures espacées de 15 minutes).

Dans le sud de la France, le panneau doit théoriquement bénéficier de l'équivalent de **20 minutes de soleil par jour en été et 1h30 en hiver** pour compenser la consommation du système, ce qui paraît tout à fait réalisable.

Pour une étude plus précise, il faudrait raisonner *à posteriori* en mesurant les performances réelles du système.

¹ Ces valeurs n'intègrent pas la consommation du battery shield.

2.5/ Programme

Le programme est compilé et téléversé à l'aide de l'IDE Arduino. Il faut au préalable le configurer pour la carte NodeMCU (la manipulation est simple et bien documentée sur internet). Il faut également prêter attention à installer sur l'ordinateur le driver USB de la carte (CH340 par exemple).

```
#include <ESP8266WiFi.h> // bibliothèque pour connecter un ESP8266 en WiFi
#include <PubSubClient.h> // bibliothèque pour envoyer ou recevoir des messages en MQTT
#include <Wire.h> // bus I2C
#include <Adafruit_Sensor.h> // capteurs Adafruit
#include <Adafruit_BME280.h> // spécifique BME280
// Cablage BME280
// - SCL -> D1
// - SDA -> D2
// Régler le BME280 en 3V
// Relier D0 à RST par un fil (à enlever lors du téléversement)
// pour que la carte puisse redémarrer après mise en "deep sleep"

#define wifi_ssid "à compléter"
#define wifi_password "à compléter" // attention: ne fonctionne pas avec une clé WEP
#define mqtt_server "adresse IP du serveur MQTT à compléter"
#define mqtt_user "à compléter"
#define mqtt_password "à compléter"
#define temperature_topic "à compléter" //topic température, exemple "modext/temp"
#define humidity_topic "à compléter" //topic humidité, exemple "modext/hum"
#define press_topic "à compléter" //topic pression, exemple "modext/press"
#define interval 15 //intervalle de mesures en minutes
#define altitude 20 //altitude en m au dessus de la mer (pour correction baromètre)
int tempsconnectionmax = 30; // temps en secondes maximum pour se connecter au réseau WIFI ou
au serveur MQTT
// passé ce délai, le module passe en deep sleep jusqu'au prochain redémarrage
long now;

long lastMsg = 0; // Horodatage du dernier message publié sur MQTT
Adafruit_BME280 bme; // objet capteur BME280
WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);
  // Test du capteur BME280
  bool status = bme.begin(0x77);
  if (!status) {
    Serial.println("Impossible de se connecter au capteur BME280");
    while (1);
  }
  Serial.println("Capteur BME280 connecté");
  now = millis();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
```

```

//client.setCallback(callback);
//----
now = millis();
if (!client.connected()) {
  reconnect();
}
client.loop();

//Lecture de l'humidité ambiante
float h = bme.readHumidity();
// Lecture de la température en Celcius
float t = bme.readTemperature();
// Lecture de la pression en hPa et conversion en pression relative au niveau de la mer
float p = bme.readPressure() / 100.0F + 1*altitude/8.36;
if (isnan(t) || isnan(h) || isnan(p)) {
  Serial.println("Echec de lecture ! Verifiez votre capteur BME280");
  return;
}
Serial.print("Temperature : ");
Serial.print(t);
Serial.print(" | Humidite : ");
Serial.println(h);
Serial.print(" | Pression relative : ");
Serial.println(p);
client.publish(temperature_topic, String(t).c_str(), true); //Publie la température sur le topic
temperature_topic
client.publish(humidity_topic, String(h).c_str(), true); //Et l'humidité
client.publish(press_topic, String(p).c_str(), true); //Et enfin la pression
delay(1000);
ESP.deepSleep((interval *60)*1000000L);
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connexion à ");
  Serial.println(wifi_ssid);
  WiFi.begin(wifi_ssid, wifi_password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    if(millis() - now > 1000*tempsconnectionmax) {
      Serial.println("Impossible de se connecter au réseau WIFI, passage en mode veille");
      ESP.deepSleep((interval *60)*1000000L);
    }
  }
  Serial.println("");
  Serial.println("Connexion WiFi etablie ");
  Serial.print("=> Adresse IP : ");

```

```

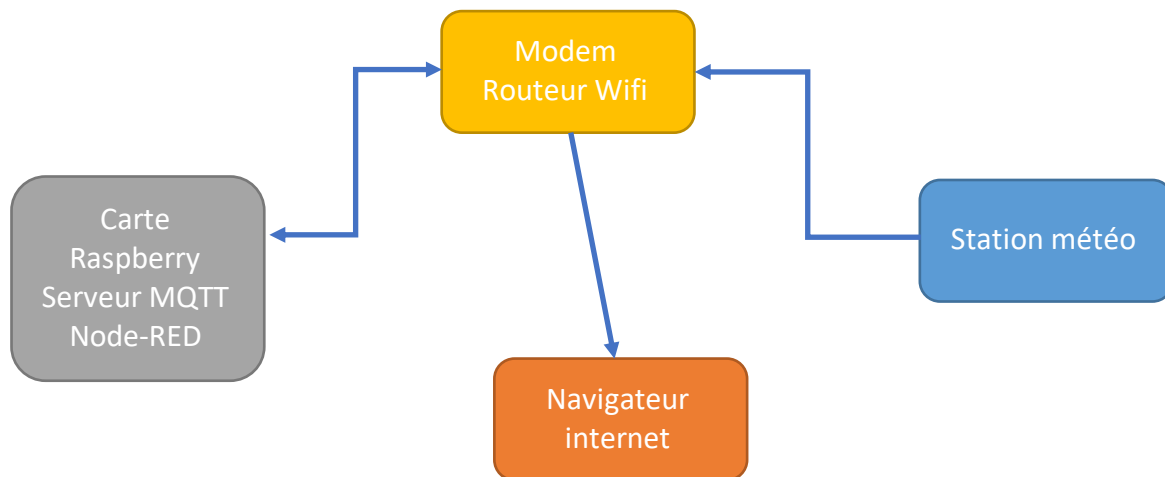
Serial.print(WiFi.localIP());
}

void reconnect() {
  //Boucle jusqu'à obtenir une reconnexion
  while (!client.connected()) {
    Serial.print("Connexion au serveur MQTT...");
    if (client.connect("ESP8266Client", mqtt_user, mqtt_password)) {
      Serial.println("OK");
    } else {
      Serial.print("KO, erreur : ");
      Serial.print(client.state());
      if(millis() - now > 1000*tempsconnectionmax) {
        Serial.println("Impossible de se connecter au serveur MQTT, passage en mode veille");
        ESP.deepSleep((interval *60)*1000000L);
      }
      Serial.println(" On attend 1 seconde avant de recommencer");
      delay(1000);
    }
  }
}

void loop() {
}

```

3/ Serveur MQTT sous Raspberry, configuration de Node-Red



La procédure est décrite de manière exhaustive et détaillée sur Youtube (Guy TechLab) :

https://www.youtube.com/watch?v=FU6Henjf_Qs
<https://www.youtube.com/watch?v=ubqzvbox5dc>
<https://www.youtube.com/watch?v=9v3j-TTgG6M>
<https://www.youtube.com/watch?v=N-Cko9s19uQ>