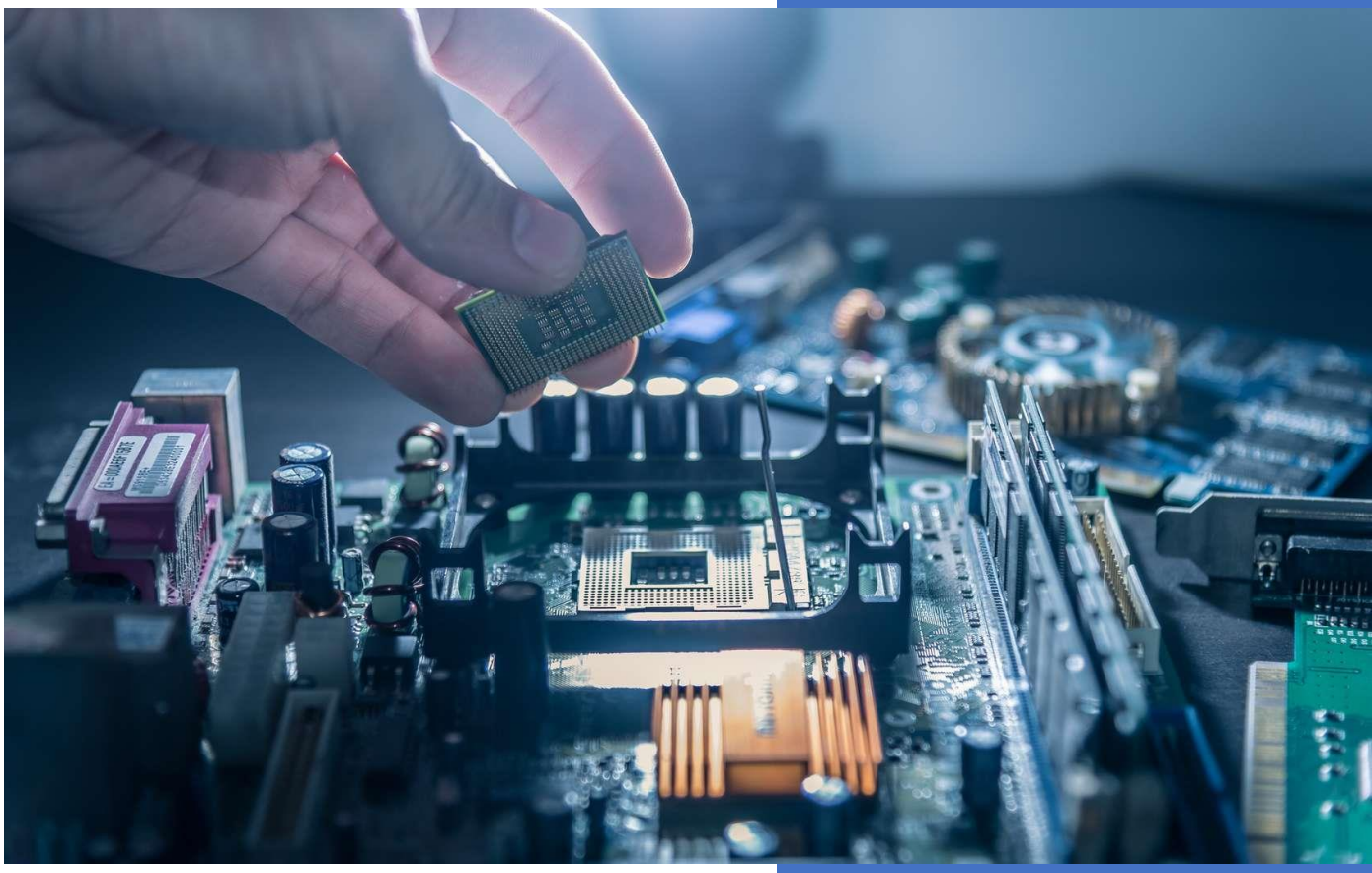
# Documentation Technique



Enzo Miragliotta

Timeo Villette

Onur Akmese

Adrien Portal

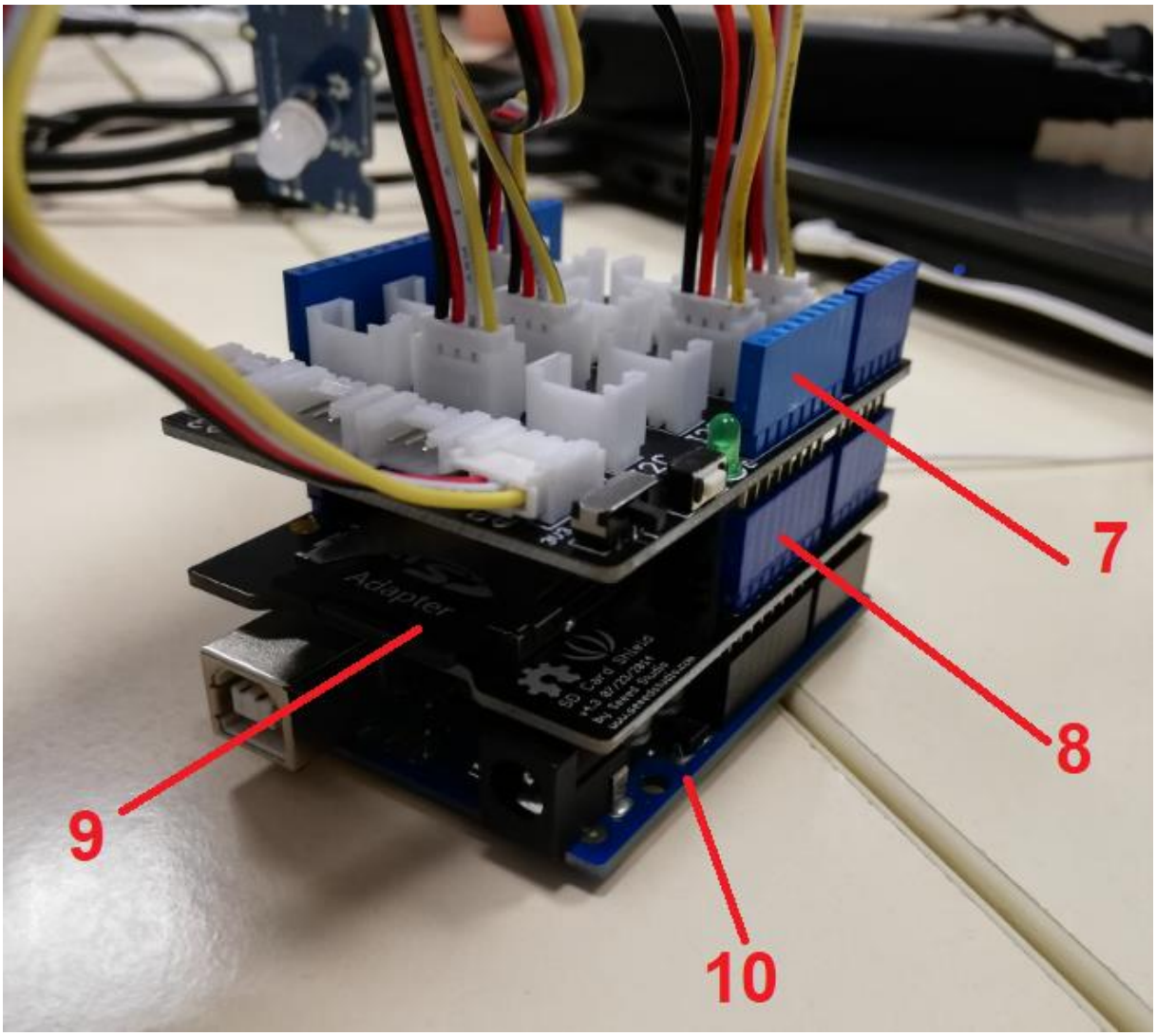
# *Sommaire :*

## Table des matières

# Introduction

Ce document est la documentation technique de la station météo. Cette documentation va vous permettre de comprendre le fonctionnement global du système avec flux d'information, l'architecture générale du programme. Nous vous présenterons aussi toutes les caractéristiques des composants de notre dispositif.

## Composants du dispositif




***Liste exhaustive des composants du système :***

| | |
|---|---|
| **1** | LED RGB |
| **2** | Boutons pressoirs Vert et Rouge |
| **3** | Horloge RTC |
| **4** | Capteur triple BME280 (température/pression/hygrométrie) |
| **5** | GPS |
| **6** | Capteur de lumière : Light Grove Sensor |
| **7** | Base Shield (communication SPI, I2 C et analogique) |
| **8** | Shield Arduino carte SD Seeed SD Card V4.3 |
| **9** | Carte SD |
| **10** | Carte microcontrôleur Arduino Uno |

*Arduino Uno*

La carte Arduino Uno est basée sur un ATMega328 cadencé à 16 MHz. Cette carte est le cœur de notre dispositif. Tous les composants sont connectés sur cette carte. Cette carte permet de stocker et d'exécuter le programme de la station météo.

Caractéristiques :

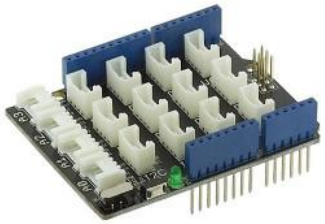
- Microprocesseur : ATMega 328
- Cadencement : 16 MHz
- Mémoire SRAM : 2 kB
- Mémoire flash : 32 kB
- Mémoire EEPROM : 1 kB
- Alimentation : via port USB ou 7 V via le connecteur d'alimentation
- Interfaces : 14 broches Entrées/Sorties, 6 entrées analogiques
- Bus série, I2C et SPI

*Module Grove Base Shield 10303000*

Le module Grove Base Shield est une carte d'interface permettant de raccorder facilement, rapidement et sans soudure les capteurs sur une carte compatible Arduino.

Caractéristiques :

- LED : indicateur de reset
- Reset : utilisation d'un bouton poussoir présent sur le module
- Connecteurs : 16 x 4 broches

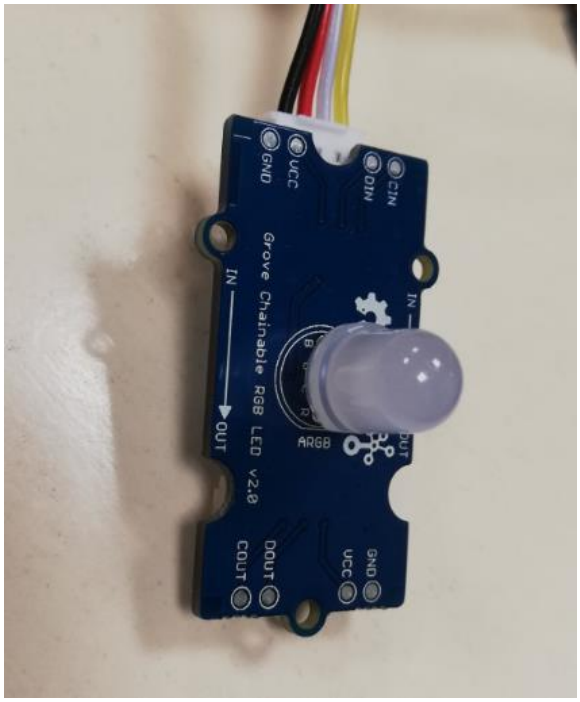


*LED RGB Grove*

Le module LED RGB Grove est un module LED permettant d'afficher la couleur souhaitée à partir d'un code RGB. Ce module est compatible avec la carte Arduino Uno.

Caractéristiques :

- Interface : compatibilité Grove
- Alimentation : 5 Vcc
- Consommation : 20 mA
- Couleur RGB



Pour choisir la couleur voulue, vous pouvez vous référer au graphique ci-dessous :

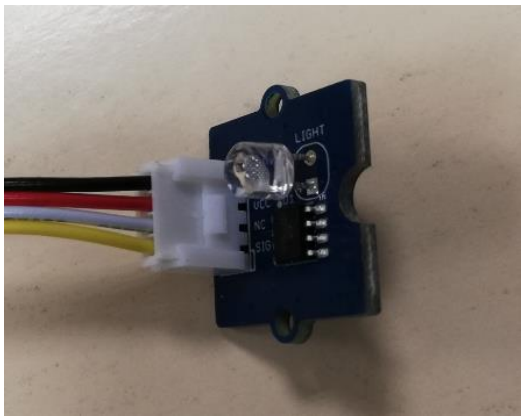
| Color | Color name | Hex | (R,G,B) |
|---|---|---|---|
| | Black | #000000 | (0,0,0) |
| | White | #FFFFFF | (255,255,255) |
| | Red | #FF0000 | (255,0,0) |
| | Lime | #00FF00 | (0,255,0) |
| | Blue | #0000FF | (0,0,255) |
| | Yellow | #FFFF00 | (255,255,0) |
| | Cyan | #00FFFF | (0,255,255) |
| | Magenta | #FF00FF | (255,0,255) |
| | Silver | #C0C0C0 | (192,192,192) |
| | Gray | #808080 | (128,128,128) |
| | Maroon | #800000 | (128,0,0) |
| | Olive | #808000 | (128,128,0) |
| | Green | #008000 | (0,128,0) |
| | Purple | #800080 | (128,0,128) |
| | Teal | #008080 | (0,128,128) |
| | Navy | #000080 | (0,0,128) |

*Détecteur de lumière Grove*

Ce capteur de lumière compatible Grove permet de détecter la présence de lumière. La tension de sortie analogique évolue de 0 à + Vcc suivant l'intensité lumineuse mesurée.

Caractéristiques :

- Interface : compatible Grove
- Alimentation : 3,3 à 5 Vcc
- Consommation : 0,5 à 3 mA
- Temps de réponse : 20 à 30 ms

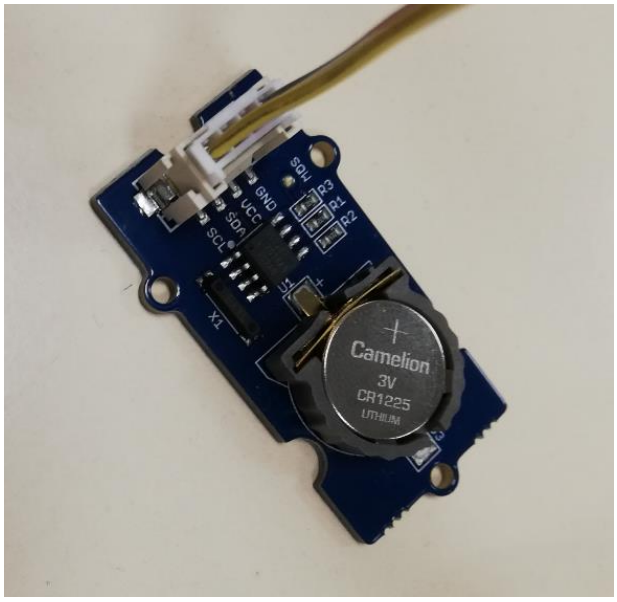


*Horloge temps réel Grove*

Ce module RTC I2C compatible Grove est basé sur le DS1307 et donne la date et l'heure au format 12h ou 24h, en tenant compte des années bissextiles.

Caractéristiques :

- Interface : compatible Grove
- Alimentation : 5 Vcc
- Consommation : 100 à 500 nA
- Sauvegarde : 1 pile 3 V

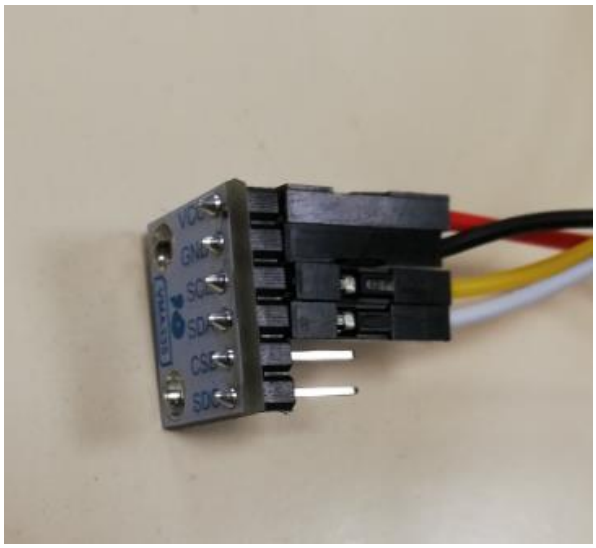



*Capteur de pression, d'humidité et de température*

Ce capteur est basé sur le circuit BME280. Il permet de mesurer la température, l'humidité et la pression atmosphérique. Il communique avec l'Arduino Uno via le bus I2C ou SPI.

Caractéristiques :

- Alimentation : 3,3 Vcc
- Plages de mesures :
    - Température : - 40 °C à 85 °C
    - Humidité : 0 à 100 % HR
    - Pression : 30 à 110 kPa
- Précision des mesures :
    - Température ± 1 °C
    - Humidité ± 3 %
    - Pression ± 1 hPa

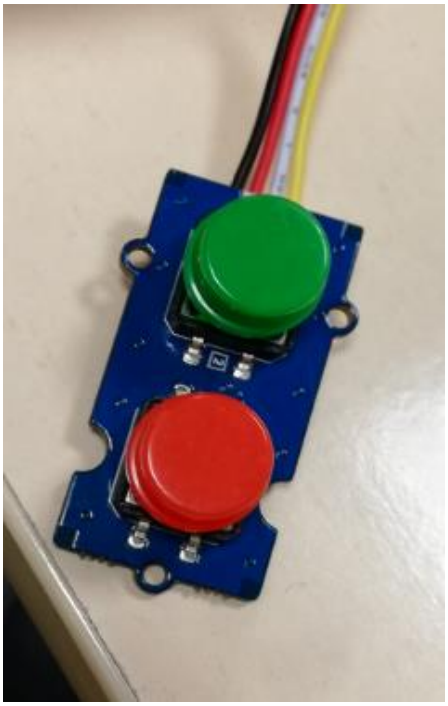


*Module 2 boutons Grove*

Ce module compatible Grove comportant deux boutons-poussoirs permet de faire passer l'état de deux sorties digitales à l'état bas (0 Vcc) lors d'une pression.

Caractéristiques :

- Alimentation : 3 à 5 Vcc
- Interface : compatible Grove



*Shield carte SD*

Le shield carte SD est une carte d'interface compatible Arduino permettant d'ajouter un espace de stockage.

Caractéristiques :

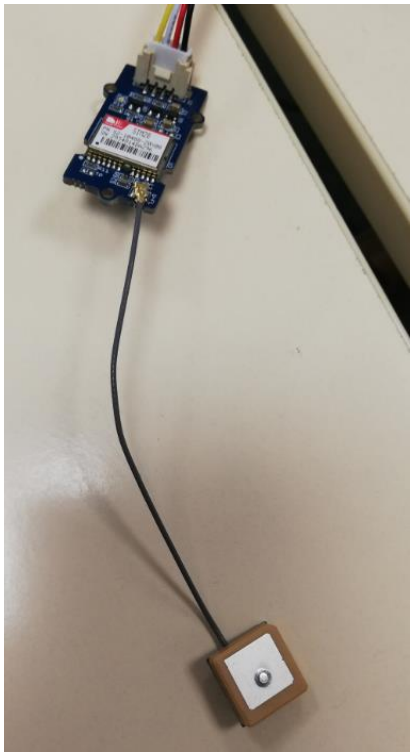
- Alimentation : 5 Vcc
- Consommation : 100 mA
- Adaptateur pour micro-SD

*Module GPS Grove*

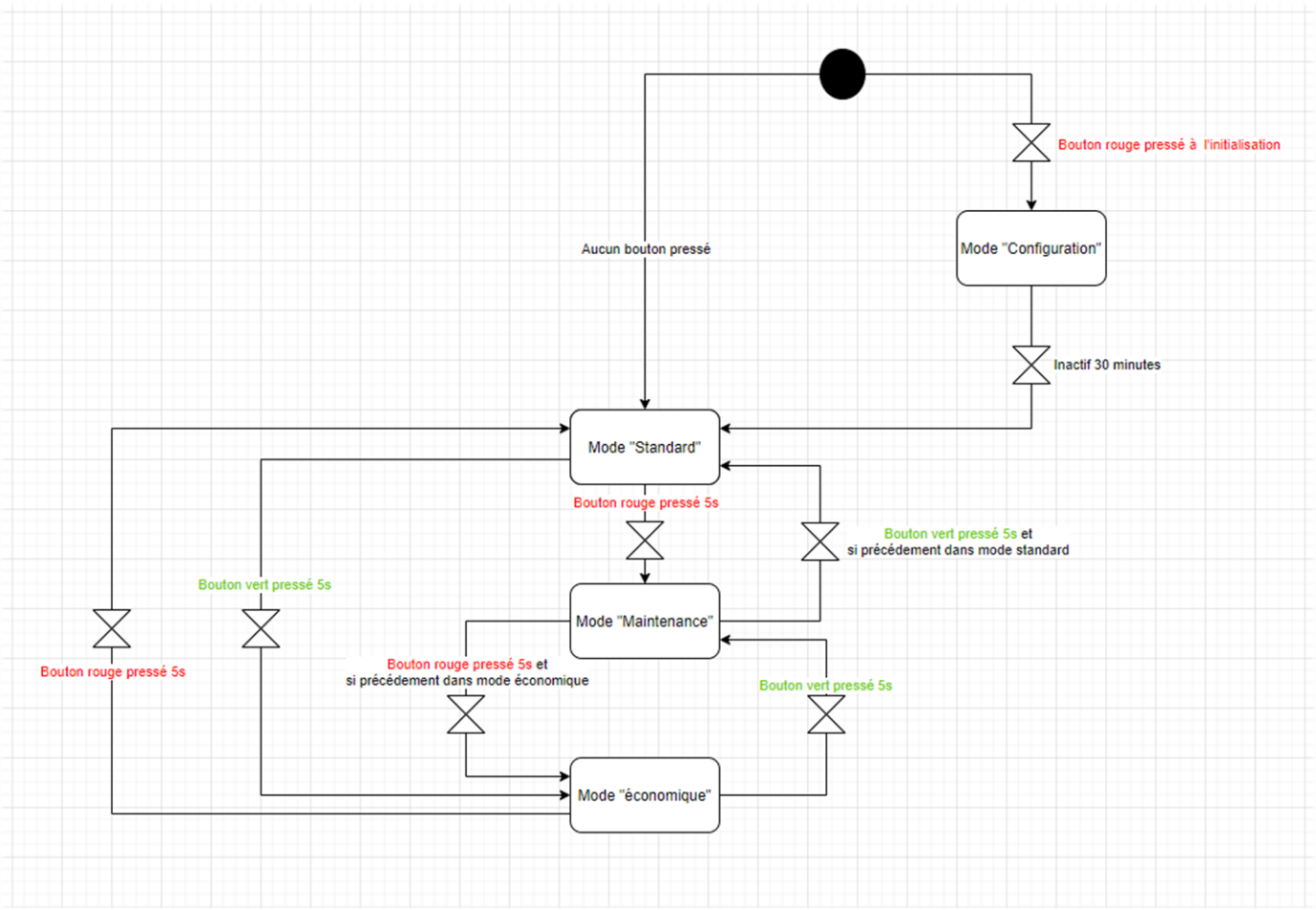
Ce capteur n'a pas été utilisé lors de la maquette, cependant il a été pris en compte dans notre maquette. Ce module permet de connaître notre position en temps réel. Il communique avec l'Arduino.

<u>Caractéristiques :</u>

- Alimentation : 3,3 ou 5 Vcc
- Consommation : 40 mA
- Sensibilité : - 160 dBm



## Schématisation des flux d'informations



Le système est capable de relever : la luminosité ambiante (Capteur de lumière), la température de l'air, l'hygrométrie, la pression (capteur triple) et la position géographique du système (via GPS). Le système possède également une horloge RTC lui permettant de se repérer dans le temps.

## Liste et utilisation des modes :

Le système a été pensé afin de fonctionner grâce à 4 modes :

- Mode standard (mode basique de fonctionnement du système)
- Mode configuration (permet de configurer le système)
- Mode économique (mode de fonctionnement spécifique permettant d'économiser la batterie)
- Mode maintenance (permet d'avoir accès aux données des capteurs directement depuis une interface série et permet de changer en toute sécurité la carte SD sans risque de corrompre les données)

### a) Mode standard :

Ce mode est le mode de fonctionnement principal du système. C'est-à-dire que dans ce mode le système récupère à intervalle régulier les données des capteurs. Au démarrage (mise sous tension du système), si l'utilisateur ne fait aucune action, le système bascule alors automatiquement en mode standard. L'utilisateur peut également repasser en mode standard depuis le mode éco grâce à un appui de 5 secondes sur le bouton vert, et depuis le mode maintenance grâce à un appui de 5 secondes sur le bouton rouge. Lorsque le système est en mode standard la LED RGB s'illumine de façon continue en vert.

### b) Mode configuration :

Le mode configuration sert à régler les différents paramètres du système. Ce dernier est accessible au démarrage du système. Il faudra au préalable avoir connecté le système à un ordinateur, avoir lancé l'IDE Arduino et ouvert le moniteur série. Une fois le système mis sous tension, veuillez appuyer sur le bouton pressoir Rouge et le système basculera automatiquement en mode configuration, la LED RGB du système s'allumera alors en jaune et ce, de manière continue. Dans ce mode, l'acquisition des capteurs est désactivée.

Dans ce mode, vous pouvez influer sur différentes valeurs :

| Paramètre | Domaine de définition des valeurs | Valeur par défaut | Description |
|---|---|---|---|
| LOG_INTERVAL | (1-360) | 10 | Définition de l'intervalle entre 2 mesures, 10 minutes par défaut. |
| FILE_MAX_SIZE | {100-32000} | 4096 | Définition de la taille maximale (en octets) d'un fichier de log, une taille de 4ko provoque son archivage. |

| | | | |
|---|---|---|---|
| RESET | X | X | Permet de rétablir les valeurs par défauts (sauf pour l'horloge) |
| LUMIN | {0,1} | 1 | Définition de l'activation (1) / désactivation (0) du capteur de luminosité |
| LUMIN_LOW | {0-1023} | 255 | Définition de la valeur en dessous de laquelle la luminosité est considérée comme "faible" |
| LUMIN_HIGH | {0-1023} | 768 | Définition de la valeur au-dessus de laquelle la luminosité est considérée comme "forte" |
| TEMP_AIR | {0,1} | 1 | Définition de l'activation (1) / désactivation (0) du capteur de température de l'air |
| MIN_TEMP_A IR | {-40-85} | -10 | Définition du seuil de température de l'air (en °C) en dessous duquel le capteur se mettra en erreur. |
| MAX_TEMP_ AIR | {-40-85} | 60 | Définition du seuil de température de l'air (en °C) au-dessus duquel le capteur se mettra en erreur. |
| HYGR | {0,1} | 1 | Définition de l'activation (1) / désactivation (0) du capteur d'hygrométrie |
| HYGR_MINT | {-40-85} | 0 | Définition de la température en dessous de laquelle les mesures d'hygrométrie ne seront pas prises en compte. |
| HYGR_MAXT | {-40-85} | 50 | Définition de la température au-dessus de laquelle les mesures d'hygrométrie ne seront pas prises en compte. |
| PRESSURE | {0,1} | 1 | Définition de l'activation (1) / désactivation (0) du capteur de pression atmosphérique. |
| PRESSURE_M IN | {300-1100} | 850 | Définition du seuil de pression atmosphérique (en HPa) en dessous duquel le capteur se mettra en erreur. |
| PRESSURE_M AX | {300-1100} | 1080 | Définition du seuil de pression atmosphérique (en HPa) au-dessus duquel le capteur se mettra en erreur |
| CLOCK | H {0-23} ;<br>M {0-59} ;<br>S {0-59} | A l'heure sur le fuseau de Paris | Configuration de l'heure. |
| DATE | M {1-12} ;<br>J {1-31} ;<br>A {2000-2999} | | Configuration de la date. |
| DAY | {MON,TUE,<br>WED,THU,<br>FRI,SAT,SUN} | | Configuration du jour de la semaine. |
| VERSION | {0-999} | 1.0 | Configuration de la version système. |
| TIMEOUT | {0-999} | 30 | Configuration de la durée en seconde |

**Liste exhaustive des commandes pouvant être entrées dans le mode configuration**

Toutes les modifications se font depuis des commandes à entrer dans le moniteur série. Le programme embarqué par le système vous guidera grâce à des séries de questions. Vous devrez alors entrer les commandes nécessaires et les valeurs que vous voudrez afin de configurer le système selon votre convenance.

Il est à noter qu'au bout de 30 minutes d'inactivité, le système alors en mode configuration passera automatiquement en mode standard.

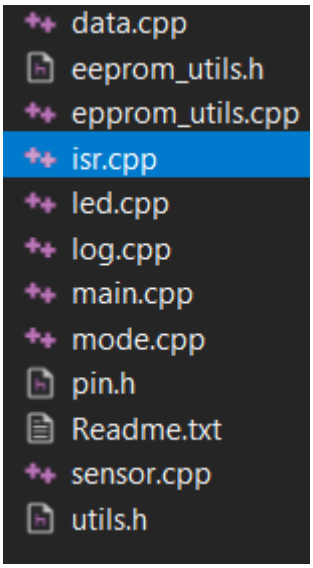
### c) Mode économique :

Accessible uniquement depuis le mode standard, il permet d'économiser de la batterie en désactivant certains capteurs et traitements. On y accède en appuyant pendant 5 secondes sur le bouton vert. En appuyant 5 secondes sur le bouton rouge, le système rebascule en mode standard. La LED RGB s'illumine en bleu de façon continue quand le système est en mode économique.

### d) Mode maintenance :

Accessible depuis les mode standard et économique, il permet d'avoir accès aux données des capteurs directement depuis une interface série et permet de changer en toute sécurité la carte SD sans risque de corrompre les données. On y accède en appuyant pendant 5 secondes sur le bouton rouge. En appuyant sur le bouton rouge pendant 5 secondes, le système rebascule dans le mode précédent.

## Justification choix dans le code




Nous avons créé des fichiers d'en-tête où nous déclarons nos variables, fonction, constante, structure, pointeur, tableau…

Ces fichiers d'en-tête nous les appelons par la suite dans les fichiers .cpp là où se trouve toute la partie pratique du code.

### Bibliothèques :

```
// On inclut les bibliothèques
#include <stdio.h>
#include <stdlib.h>
#include <Arduino.h>
#include <ChainableLED.h>

// On inclut nos bibliothèques pour la communication SPI et avec la carte SD
#include <SPI.h>
#include <SD.h>

// Inclusion des bibliothèques pour le capteur BME280(Pression, Température, Humidité)
#include <BME280I2C.h>
#include <Wire.h>

// Inclusion de la bibliothèque pour le capteur RTC
#include "DS1307.h"

// Inlcusion de la bibliothèque pour le capteur GPS
#include <SoftwareSerial.h>

// On inclut le fichier d'en-tete qui definit les numeros de ports
#include "pin.h"
```

En effet, la première chose que nous comptons faire est de « dire » au programme et donc à la carte Arduino quelles sont les librairies indispensables à l'exécution du programme. L'ajout des bibliothèques au programme se fait grâce à la commande « #include <nomdelalibrairie>.h ».

Nous avons inclus plusieurs librairies notamment pour certains modules ou composants de notre système (librairie GPS, librairie de la LED RGB…). C'est donc pour cette raison que nous avons choisi de réserver une partie de notre programme à l'inclusion de librairie.

Mais nous avons également inclus les librairies nécessaires au fonctionnement des capteurs.

## Switchcase :

Tout d'abord, les switchcases permettent de réduire le code et donc l'espace nécessaire.

```
switch (pdata->params.MODE) {
case STANDARD:
    pdata->led->setColorRGB(0, 255, 0);
    break;
case ECO:
    pdata->led->setColorRGB(0, 0, 255);
    break;
case MAINTENANCE:
    pdata->led->setColorRGB(255, 140, 0);
    break;
case CONFIG:
    pdata->led->setColorRGB(255, 255, 255);
    break;
```

## Pointeur :

Ensuite, nous utilisons des pointeurs pour une utilisation dynamique de la mémoire. De plus, la fonction malloc nous permet de réserver la mémoire nécessaire pour une variable.

```
//appel du pointeur data de la liste chainée
data *pdata;
```

```
pdata->led = new ChainableLED(PIN_RGB1, PIN_RGB2, 0);
```

Exemple d'utilisation de malloc :

```
pdata = malloc(sizeof(data));
```

## Structure :

Nous avons utilisé des structures plutôt que des tableaux car elles permettent de regrouper des variables de types différents. Exemple d'utilisation des structures :

```
struct mesure {
    signed int lumin[10];
    signed char temp[10];
    signed char hygr[10];
    signed int pressure[10];
};
```

## Liste chainée

A l'aide des pointeurs et des structures, nous avons utilisé des listes chaînées pour l'enregistrement des mesures. Lors de la déclaration d'un tableau, le nombre de valeur contenues dans le système doit être connu. Ce n'est pas nécessaire pour une liste chaînée, ce qui permet d'y ajouter autant de valeurs que l'on veut. Exemple d'utilisation d'une liste chaînée :

```
// liste chainée
typedef struct data {
    struct params params;
    struct timer timer;
    struct mesure mesure;
    ChainableLED *led;
    RTC_DS1307 rtc;
    DateTime now;
    String gps;
} data;
```

Pour conclure notre code est léger, faible en espace mémoire et rapide à l’exécution grâce aux éléments insérer dans notre code (switchcase, pointeur, liste chainée et structure…).