

DOCUMENTATION TECHNIQUE

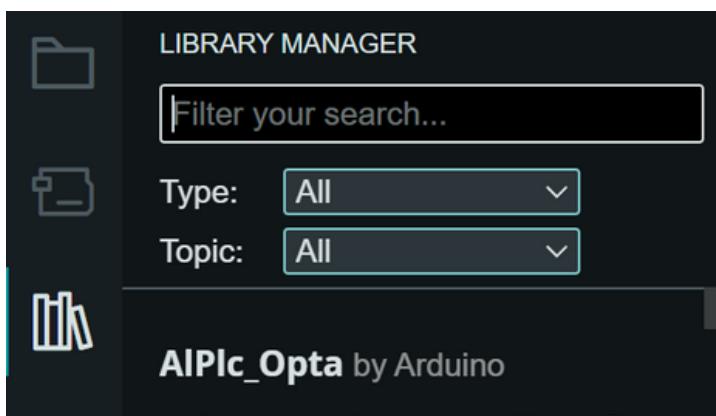
◆ Introduction

Le projet de station météo embarquée pour navires est une initiative menée par l'Agence Internationale pour la Vigilance Météorologique (AIVM). L'objectif est de doter des navires de surveillance de stations capables de mesurer les paramètres environnementaux critiques en haute mer, permettant ainsi de surveiller les conditions météorologiques susceptibles de conduire à des catastrophes naturelles comme les cyclones. Ce système, prototypé sur une carte Arduino, est conçu pour être simple, efficace, et utilisable par un membre d'équipage sans formation technique poussée.

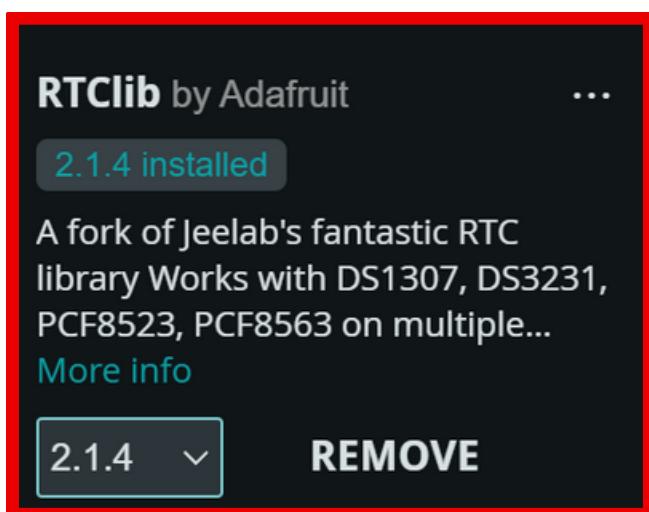
Le prototype actuel repose sur un microcontrôleur AVR ATmega328 et une série de capteurs qui enregistrent des données environnementales. Le système est flexible, offrant quatre modes de fonctionnement principaux qui répondent aux différents besoins d'acquisition, de configuration et de maintenance des données.

◆ Les bibliothèques :

Rechercher les bibliothèques ici :

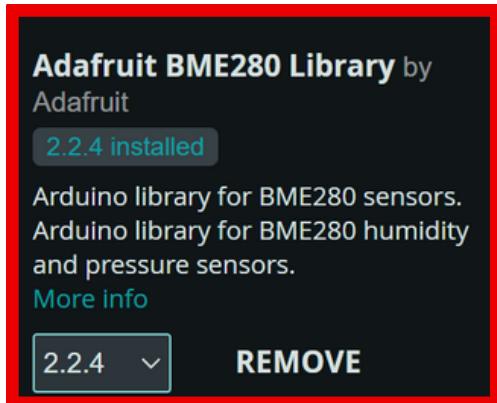


Installation de RTCLib :



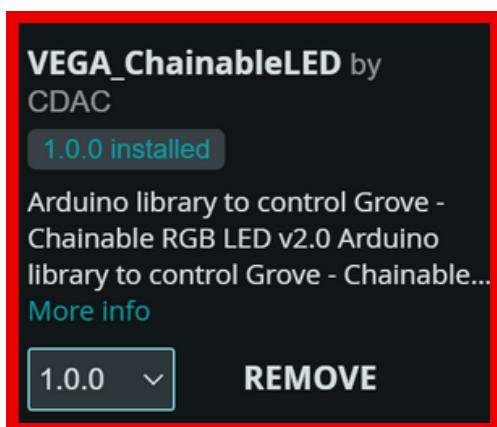
▼ Cette bibliothèque est nécessaire pour utiliser un module d'horloge en temps réel (RTC) comme le DS3231 ou le DS1307. Elle fournit des fonctions pour récupérer l'heure et la date, assurant une horloge indépendante de l'alimentation de l'Arduino.

Installation de BME280 :



- ▼ Cette bibliothèque permet d'interagir avec le capteur BME280 d'Adafruit, qui mesure la température, la pression et l'humidité. Elle simplifie l'acquisition des données en offrant des méthodes prêtes à l'emploi.

Installation ChainableLED :



- ▼ Cette bibliothèque contrôle des LEDs chainables (comme les modules de LEDs RGB en série). Elle permet de configurer individuellement la couleur et l'intensité de chaque LED dans une chaîne, pratique pour des signalisations d'erreur ou des effets lumineux.

◆ Matériels Requis :

Microcontrôleur : AVR ATmega328, intégré à une carte Arduino pour contrôler l'ensemble du système.

Lecteur de Carte SD : Pour la sauvegarde des données.

Horloge RTC : Pour l'horodatage précis des enregistrements.

LED RGB : Indicateur visuel de l'état du système.

Boutons Poussoirs : Deux boutons pour basculer entre les modes et interagir avec le système.

Capteurs Intégrés :

- Pression atmosphérique
- Température de l'air
- Hygrométrie (humidité)
- GPS
- Luminosité

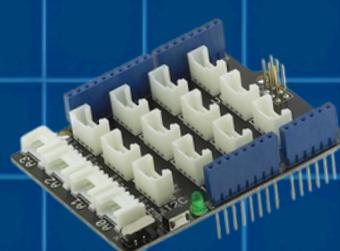
◆ Matériels Détaillés :



..... ➤ 1 Carte Arduino UNO
ref : Arduino UNO R3



..... ➤ 1 Shield de carte SD pour Arduino
ref : SD Card shield v4.3 07/23/2014



..... ➤ 1 Base Shield pour Arduino
ref : Base Shield v2.1 08/31/2015



..... ➤ 1 câble USB-A vers USB-B
ref : Cable pour carte Arduino Uno USB30CMARD



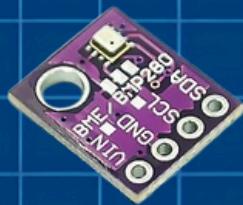
..... ➤ 4 câbles de connectiques
ref : Cordon Grove - JST/SJ 4 broches



..... ➤ 1 câble de connectique avec un côté, des broches séparés
ref : DeLOCK Conversion IOT Grove Câble 4 broches mâle vers 4 x Jumper femelle 20 cm (86947)



..... ➤ 1 Capteur RTC
ref : RTC v1.2



1 Capteur de température,
pression, humidité
ref : BME280 WPSE335



1 Capteur de luminosité
ref : Grove - Light sensor v1.2 01/20/2016



1 LED RGB

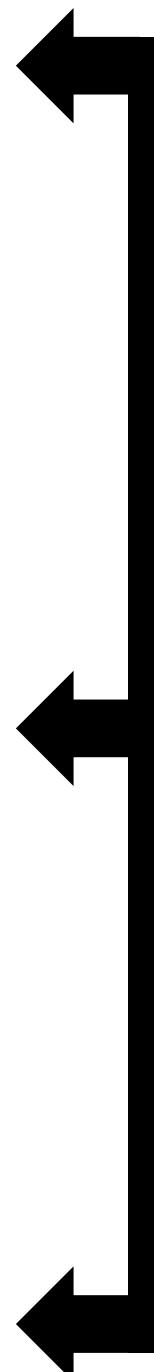
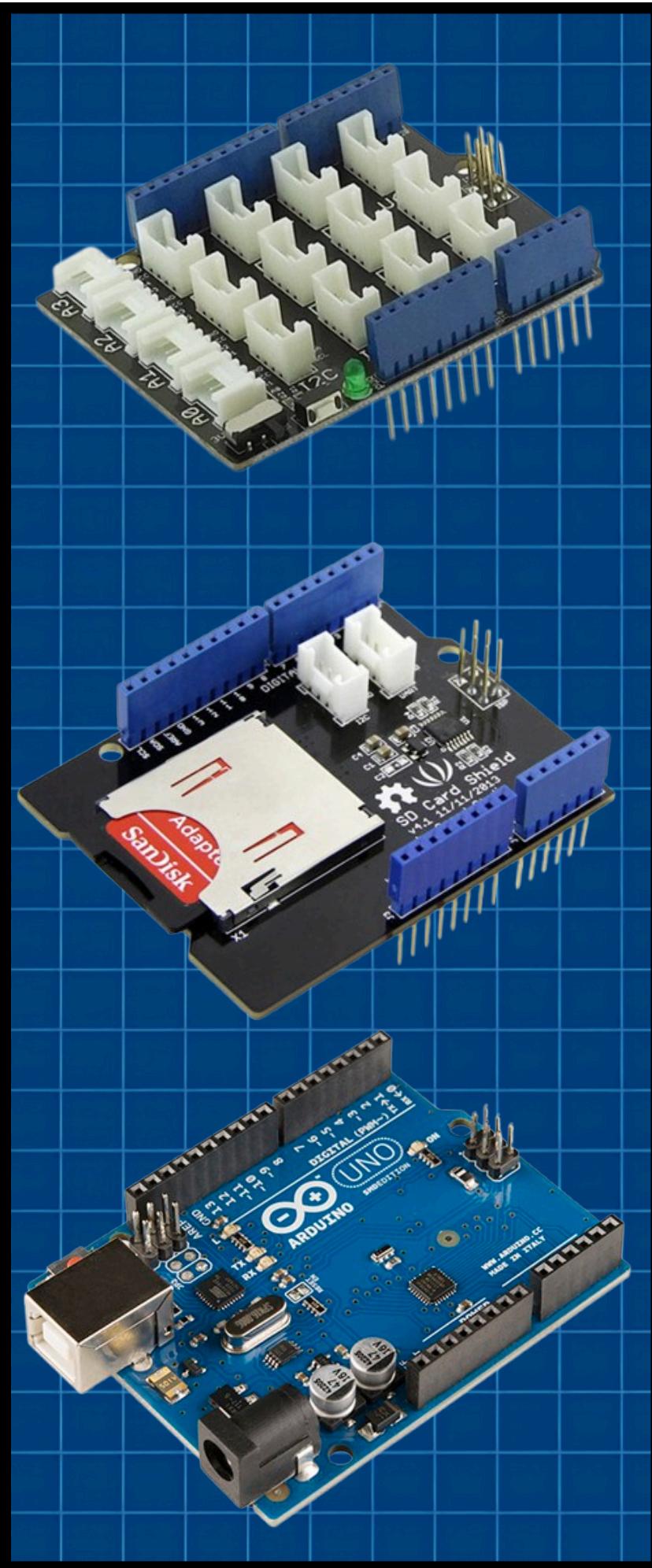
ref : Grove Chainable RGB LED v2.0



1 capteur regroupant 2 boutons ,
un vert et un rouge

ref : Grove - Dual Button v1.0.

◆ Les Branchements :



EXPLICATION :

Emboiter les 3 composants dans cet ordre délicatement en faisant correspondre les broches aux ports

Arduino UNO R3:



L'Arduino Uno R3 est équipé d'un microcontrôleur ATmega328P fonctionnant sur une architecture 8 bits AVR. Il utilise une tension de 5V pour son fonctionnement et peut être alimenté par une tension de 7 à 12V via un connecteur externe ou par 5V en utilisant le port USB. La carte dispose de 14 broches numériques, dont 6 peuvent générer des signaux PWM, et de 6 broches d'entrée analogiques, permettant une grande flexibilité pour des projets variés d'acquisition et de contrôle de signaux.

SD Card shield v4.3 07/23/2014 :



Le module SD Card Shield v4.3 (07/23/2014) est conçu pour ajouter un stockage extensible aux projets Arduino en utilisant des cartes SD. Il fonctionne à une tension de 5V, fournie directement par la carte Arduino, et communique via l'interface SPI. Ce shield est compatible avec les cartes SD et SDHC formatées en FAT16 ou FAT32, offrant un espace de stockage étendu pour la sauvegarde de données. Il comporte un connecteur de carte SD standard, un circuit intégré pour la conversion de niveaux logiques entre le microcontrôleur Arduino et la carte SD, ainsi que des broches d'en-tête passantes pour une compatibilité avec d'autres shields.

Base Shield v2.1 08/31/2015 :



Le Base Shield v2.1 (08/31/2015) est un shield de connectivité conçu pour simplifier l'ajout de modules Grove à une carte Arduino. Fonctionnant à une tension de 5V, il dispose de plusieurs ports compatibles avec la gamme Grove, permettant une connexion rapide des capteurs, actionneurs et modules sans nécessiter de soudure. Le shield comprend des ports pour E/S numériques, analogiques, UART et I2C, organisés pour faciliter la gestion des périphériques multiples. Les en-têtes d'empilage sont également compatibles avec d'autres shields, ce qui en fait une solution pratique pour des prototypes modulaires et évolutifs.

Câble pour carte Arduino Uno USB30CMARD :



Le câble USB pour carte Arduino Uno (modèle USB30CMARD) est un câble USB de 30 cm permettant de connecter l'Arduino Uno R3 (et d'autres modèles compatibles) à un ordinateur. Il assure l'alimentation de la carte en 5V et permet de transférer les données pour la programmation et la communication série via une interface USB Type-A vers USB Type-B. Avec sa longueur compacte, ce câble est idéal pour les configurations de bureau ou pour les projets nécessitant une connexion stable et proche de l'ordinateur.

Cordon Grove - JST/SV 4 broches :



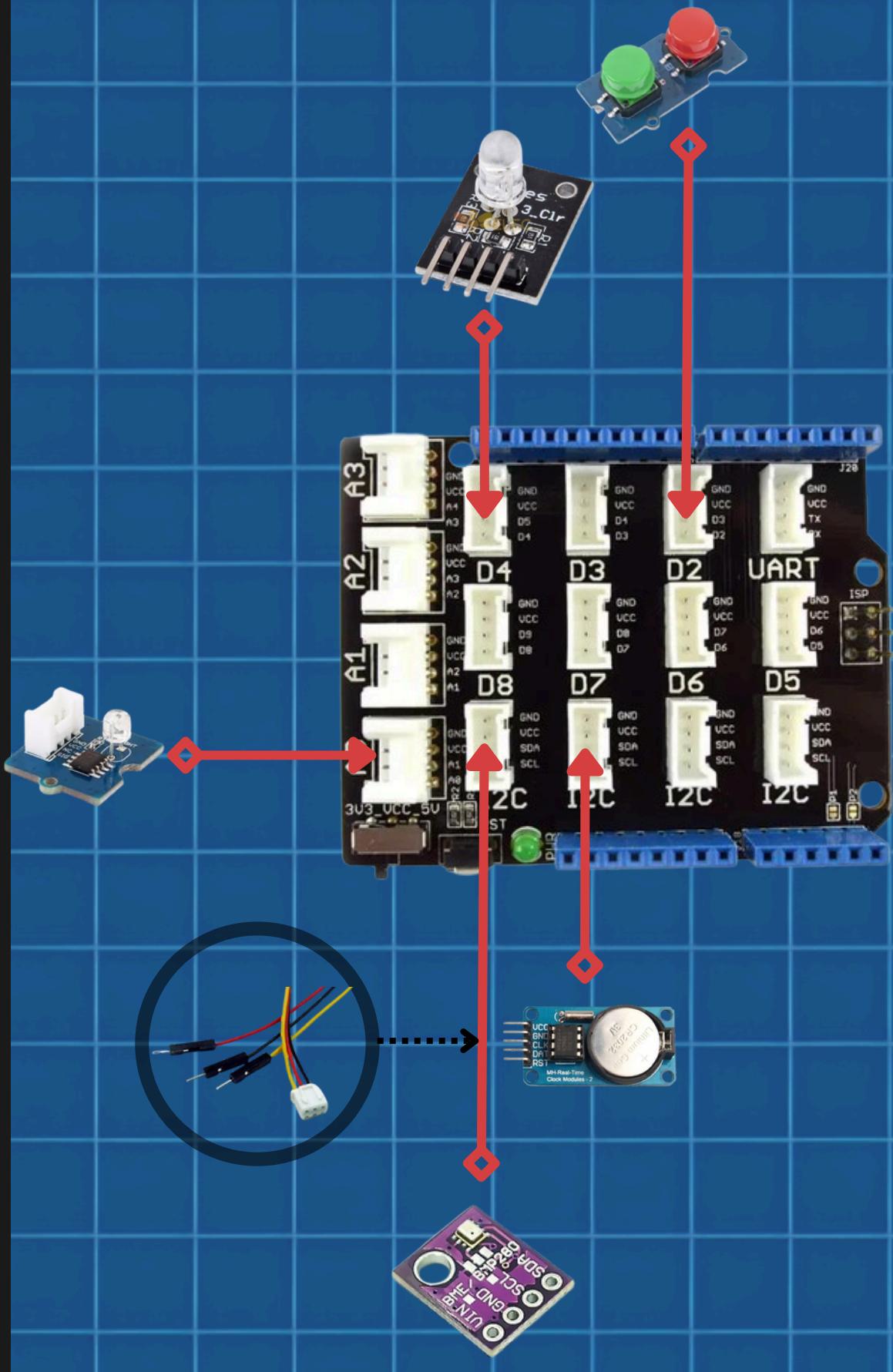
Le cordon Grove JST/SV 4 broches est un câble de connexion conçu pour les modules Grove, permettant une intégration facile et rapide de capteurs et d'actionneurs avec des cartes Arduino et d'autres microcontrôleurs. Doté de connecteurs JST à chaque extrémité, ce câble assure une connexion sécurisée et fiable. Sa conception à 4 broches permet de transmettre des signaux numériques et analogiques, facilitant ainsi la communication entre les dispositifs. Ce cordon est idéal pour les projets nécessitant une connexion directe et modulable, tout en maintenant un câblage ordonné et sans encombrement.

DeLOCK Conversion IOT Grove Câble 4 broches mâle vers 4 x Jumper femelle 20 cm (86947) :



Le câble DeLOCK Conversion IoT Grove est un câble de 20 cm qui permet de convertir une connexion Grove à 4 broches mâle en quatre connecteurs femelles jumper. Ce câble est idéal pour les utilisateurs souhaitant intégrer des modules Grove dans des projets plus complexes, offrant la flexibilité d'utiliser des câbles jumper standard pour des connexions personnalisées. Sa conception à 4 broches permet de transmettre à la fois des signaux numériques et analogiques, facilitant ainsi la communication avec divers capteurs et actionneurs. Ce câble est parfait pour les prototypes et les applications nécessitant une configuration de câblage adaptative et modulable.

MONTAGE VUE DE HAUT



BME280 :



Pour ce qui est des interfaces de communications pour le BME280 nous utilisons la liaison I2C pour plusieurs raison d'une part pour la communication simplifiée car l'I2C est un bus de communication qui permet de connecter plusieurs périphériques avec seulement deux fils : un pour les données (SDA) et un pour l'horloge (SCL). Cela rend le montage plus simple, surtout quand on dispose de plusieurs capteurs. De plus le BME280 est compatible avec l'I2C, ce qui facilite son intégration dans les projets Arduino et autres microcontrôleurs. L'I2C est largement supporté, donc il y a souvent des bibliothèques préexistantes, ce qui simplifie la programmation.

Grove - Dual Button v1.0 :

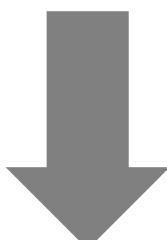


Pour le Grove - Dual Button v1.0, nous avons utilisé une entrée digitale car elle est adaptée à la détection des états de bascule du bouton, qui sont soit activés (1) soit désactivés (0). En effet, lorsqu'un bouton est pressé, il agit comme un interrupteur, ne permettant que deux valeurs d'état possibles, représentées numériquement par un niveau logique haut (1) ou bas (0).

Grove - Chainable RGB LED v2.0 :



Pour la Grove - Chainable RGB LED v2.0, nous avons utilisé une sortie digitale pour contrôler l'état de la LED, car chaque couleur peut être activée ou désactivée par un signal numérique. Contrairement à une LED standard qui se limite à deux états (allumé ou éteint), la LED RGB chainable permet de générer une large gamme de couleurs grâce à un contrôle numérique sophistiqué. Chaque couleur (rouge, vert, bleu) de la LED est contrôlée par un signal numérique, où un état haut (1) correspond à l'activation de la couleur et un état bas (0) à sa désactivation. Ce système binaire simplifie le contrôle de chaque couleur primaire, permettant de créer des combinaisons de couleurs en modifiant les états de chaque couleur.



RTC v1.2 :



Pour la RTC v1.2 (Real-Time Clock), nous avons utilisé une liaison I2C car elle permet une communication rapide et efficace entre la carte Arduino et l'horloge en temps réel, avec plusieurs avantages clés :

- Communication bidirectionnelle et multiplexée : L'I2C permet une communication bidirectionnelle, ce qui signifie que l'Arduino peut envoyer des commandes à la RTC pour, par exemple, définir une heure, tout en recevant en retour des informations précises sur l'heure et la date. De plus, plusieurs périphériques peuvent partager le même bus I2C en utilisant des adresses uniques, permettant d'ajouter facilement d'autres capteurs ou composants sans avoir à multiplier les connexions.
- La liaison I2C garantit une synchronisation stable entre l'Arduino et la RTC. Grâce au signal d'horloge partagé (SCL), les données sont transmises de manière coordonnée, minimisant les risques d'erreurs de communication et assurant une lecture précise du temps, ce qui est essentiel pour des applications nécessitant une mesure de temps fiable.

Grove - Light Sensor v1.2 :



Pour le Grove - Light Sensor v1.2, nous avons utilisé une entrée analogique car le capteur mesure la luminosité ambiante en produisant une tension proportionnelle à l'intensité de la lumière, nécessitant une lecture analogique pour obtenir des informations précises sur la luminosité.

L'utilisation d'une entrée analogique pour ce capteur de lumière se justifie par les raisons suivantes :

- Lecture de valeurs continues : La luminosité ambiante varie de façon continue, et le capteur produit une tension proportionnelle à cette intensité lumineuse. Une entrée analogique permet de lire cette variation en convertissant la tension mesurée en une valeur numérique correspondante, typiquement entre 0 et 1023 sur l'Arduino, représentant des niveaux de luminosité de faible à très intense.
- Précision des mesures : Contrairement à une entrée digitale, qui ne pourrait détecter qu'un état "allumé" ou "éteint", l'entrée analogique permet de capturer des variations subtiles de la lumière. Cela est essentiel pour des applications où l'intensité lumineuse doit être mesurée avec précision, comme dans les projets de contrôle automatique de la luminosité.

◆ Le fonctionnement de la station météo :

Le système propose quatre modes principaux de fonctionnement, chacun avec des fonctionnalités spécifiques et des interactions définies.

Mode 1 : Mode Standard

Objectif : Acquisition régulière des données environnementales.

Description :

Ce mode démarre automatiquement sans qu'aucun bouton ne soit pressé au démarrage.

Le système lit les capteurs à intervalles réguliers (par défaut, toutes les 10 minutes, configurables via LOG_INTERVAL).

Les données sont sauvegardées dans des fichiers sur la carte SD. Chaque fichier est limité à 2 Ko (FILE_MAX_SIZE), après quoi un nouveau fichier est créé avec un numéro de révision.

Si un capteur ne répond pas dans un délai (TIMEOUT) de 30 secondes, la valeur "NA" est enregistrée.

Mode 2 : Mode Configuration

Objectif : Permettre la configuration des paramètres du système.

Description :

- Accessible en maintenant le bouton rouge enfoncé lors du démarrage.
- L'acquisition des capteurs est suspendue pour permettre l'interaction avec l'interface série.
- Commandes configurables via l'interface série :
 - LOG_INTERVAL : Définir l'intervalle de mesure.
 - FILE_MAX_SIZE : Ajuster la taille maximale des fichiers de log.
 - TIMEOUT : Définir le délai de réponse des capteurs.
 - RESET : Réinitialisation des paramètres aux valeurs par défaut.
 - VERSION : Affiche la version du programme et un numéro de lot.
 - CLOCK et DATE : Définissent l'heure et la date via le RTC.
- Après 30 minutes d'inactivité, le système retourne automatiquement en mode standard.

Mode 3 : Mode Maintenance

Objectif : Permettre l'accès aux données via l'interface série et remplacer la carte SD.

Description :

- Accessible depuis le mode standard ou économique par un appui prolongé (5 secondes) sur le bouton rouge.
- L'acquisition et l'enregistrement des données sont suspendus, mais les données déjà stockées restent accessibles.
- La carte SD peut être retirée en toute sécurité pour éviter la corruption des fichiers.
- Un nouvel appui prolongé sur le bouton rouge fait basculer le système vers le mode précédent.

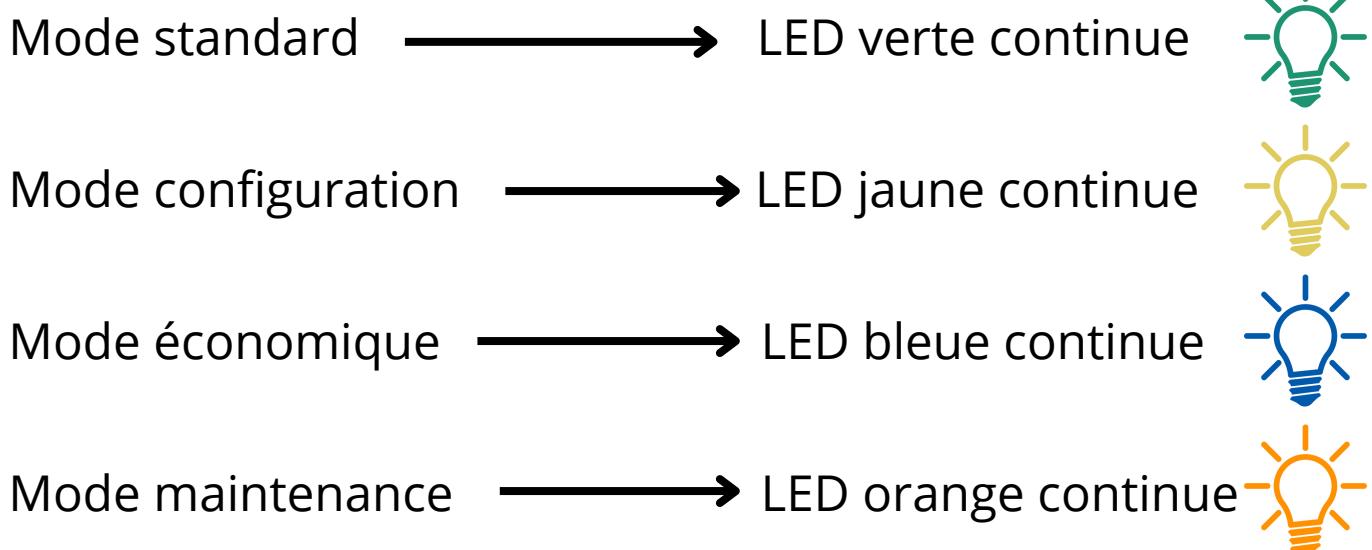
Mode 4 : Mode Économique

Objectif : Optimiser la consommation d'énergie.

Description :

- Accès depuis le mode standard par un appui prolongé de 5 secondes sur le bouton vert.
- Ce mode réduit l'activité du système en désactivant certains capteurs et en doublant l'intervalle d'acquisition (LOG_INTERVAL).
- Un appui long sur le bouton rouge permet de revenir en mode standard.

Le système indique à l'utilisateur le mode actuel grâce à la LED RGB



Lorsque que le mode sélectionné par l'utilisateur est le Mode Configuration :

- ▼ L'utilisateur peut modifier la configuration grâce aux commandes configurables. La configuration sera alors sauvegardé dans l'EEPROM
- ▼ La configuration par défaut du système est aussi sauvegarder dans l'EEPROM.
- ▼ C'est quoi l'EEPROM ? Dans un système embarqué, l'EEPROM (Electrically Erasable Programmable Read-Only Memory) est une mémoire non volatile utilisée pour stocker des données configurables qui doivent persister même après redémarrage, comme des paramètres ou des réglages de capteurs.

Le système propose l'affichage des erreurs des capteurs avec la LED RGB à une fréquence de 1hz.

- Erreur d'accès à l'horloge RTC : LED rouge et bleue (durée identique). 
- Erreur d'accès aux données du GPS : LED rouge et jaune (durée identique). 
- Erreur d'accès aux données d'un capteur : LED rouge et verte (durée identique). 
- Données incohérentes reçues d'un capteur : LED rouge et verte (durée plus longue pour le vert), indiquant une vérification matérielle nécessaire. 
- Carte SD pleine : LED rouge et blanche (durée identique). 
- Erreur d'accès ou d'écriture sur la carte SD : LED rouge et blanche (durée plus longue pour le blanc), signalant un problème de stockage. 



En mode maintenance, il n'y pas d'erreur lorsque l'utilisateur enlève la carte SD

Lorsque que l'acquisition des données se fait et que le système les sauvegarde sur la carte SD alors :

- ▼ Les noms des fichiers prendront la forme suivante :
200531_0.LOG (Année=20, mois=05, jour=31, numéro de révision=0)

◆ Explication du Programme C++ :

Code en C++ de la station météo :

- ▼ code C++ dans **les pièces jointes**

Explication des variables

Variables associées aux broches :

- **boutonRougePin** et **boutonVertPin** : Pins 2 et 3, assignées aux boutons rouge et vert du module Grove Dual Button. Ces entrées gèrent le basculement de modes en répondant aux appuis et durées d'appui.
- **lightSensorPin** : Pin analogique A0, dédiée au capteur de luminosité Grove. Cette entrée analogique mesure la lumière ambiante.
- **chipSelect** : Définit la pin 4 comme la broche CS (Chip Select) pour la carte SD, essentielle pour initialiser et accéder au stockage.

Objets et modules capteurs :

- **leds** : Objet ChainableLED connecté aux pins D4 et D5, configuré avec une LED RGB pour indiquer les modes et erreurs par des codes couleur.
- **bme** : Capteur BME280 pour mesurer température, pression et humidité, utilisé dans le suivi environnemental.
- **rtc** : Horloge RTC_DS1307 permettant de gérer la date et l'heure, utilisée pour le suivi des événements et le stockage de données.

Variables de configuration des capteurs :

- **LUMIN** : Active ou désactive le capteur de luminosité (0 = inactif, 1 = actif).
- **LUMIN_LOW** et **LUMIN_HIGH** : Définissent les seuils bas et haut pour la luminosité. Si la luminosité mesurée tombe en dehors de ces valeurs, des actions spécifiques peuvent être déclenchées (ex. déclenchement d'alertes).
- **TEMP_AIR** : Active ou désactive le capteur de température de l'air.
- **MIN_TEMP_AIR** et **MAX_TEMP_AIR** : Températures minimum et maximum pour le capteur de température, encadrant une plage de fonctionnement sûre. Toute lecture au-delà de ces seuils signale un état d'erreur.
- **HYGR** : Active ou désactive le capteur d'hygrométrie (humidité de l'air).
- **HYGR_MINT** et **HYGR_MAXT** : Températures minimum et maximum pour une mesure fiable de l'humidité.
- **PRESSURE** : Active ou désactive le capteur de pression.
- **PRESSURE_MIN** et **PRESSURE_MAX** : Pression minimale et maximale en hPa, définissant les seuils d'alerte pour le capteur de pression.

Variables de mode et état :

- **mode** : Décrit le mode de fonctionnement actuel (ex. "standard", "maintenance", "économique", "configuration").
- **lastMode** : Stocke le mode précédemment actif, permettant de basculer facilement d'un mode à un autre.
- **enErreur** : Indique si le système est en mode d'erreur. Ce flag est activé si des capteurs ou la carte SD présentent des problèmes.

Variables d'inactivité et d'intervalle :

- **tempsInactivite** : Marqueur temporel pour suivre la dernière activité, afin de revenir automatiquement en mode "standard" après une période d'inactivité.
- **INACTIVITE_TIMER** : Délai en millisecondes avant de revenir en mode "standard" après une période d'inactivité.
- **LOG_INTERVAL** et **intervalAcquisition** : Définit l'intervalle de temps en millisecondes entre les enregistrements des données des capteurs.
- **dernier_acquisition** : Stocke le dernier moment où les données des capteurs ont été enregistrées, permettant de gérer les intervalles d'acquisition sans surcharge de données.

Variables de détection d'appui sur les boutons :

- **boutonRougeEtat, boutonRougeDernierEtat, dernierTempsRouge, tempsAppuiRouge** : Variables spécifiques au bouton rouge. Elles enregistrent son état actuel et précédent, le temps écoulé depuis le dernier changement d'état et la durée de chaque appui.
- **rougeAppuiEnCours** : Indique si le bouton rouge est maintenu enfoncé.

- **boutonVertEtat, boutonVertDernierEtat, dernierTempsVert, tempsAppuiVert** : Variables spécifiques au bouton vert, avec les mêmes fonctions que pour le bouton rouge.
- **vertAppuiEnCours** : Indique si le bouton vert est maintenu enfoncé.
- **Delay_anti_rebond** : Durée en millisecondes pour éviter les rebonds mécaniques des boutons.
- **attenteLong** : Durée minimale en millisecondes pour enregistrer un appui long, ce qui déclenche un changement de mode selon la durée d'appui.

Variables de gestion de clignotement de LED :

- **dernier_changement_etat** : Dernière fois que l'état de clignotement de la LED a été modifié.
- **changement_etat** : Indicateur de changement d'état de la LED (actif/inactif) pour le clignotement, utilisé pour alterner les couleurs dans la fonction clignoter_LED.

Variables d'affichage et de maintenance :

- **dernierLogMaintenance** : Marqueur pour le dernier affichage des données en mode "maintenance".
- **MAINTENANCE_INTERVAL** : Intervalle d'affichage des données (en ms) en mode maintenance, contrôlant la fréquence de la mise à jour des lectures sur le moniteur série.

Fonctionnalités et gestion des paramètres :

- **messageEconomieAffiche** : Flag pour indiquer si le message du mode économie a déjà été affiché, afin d'éviter de répéter l'affichage.

Explication des fonctions

Fonctions de configuration et d'initialisation :

- **setup()** : Fonction initiale qui configure toutes les broches et initialise les capteurs (BME280, RTC, carte SD, etc.). Elle effectue des vérifications d'erreur et configure les états initiaux des capteurs et des LEDs.
- **configurerCapteurs()** : Fonction de configuration des capteurs, notamment pour lire et appliquer les paramètres de seuils et d'intervalles stockés dans l'EEPROM. Elle permet également de définir les options d'activation des capteurs en fonction des préférences sauvegardées.

Fonctions de gestion des capteurs :

- **lireLuminosite()** : Mesure la luminosité actuelle avec le capteur de lumière et vérifie si elle se trouve dans les seuils définis par LUMIN_LOW et LUMIN_HIGH.
- **lireTemperature()** : Lit la température de l'air à partir du capteur BME280 et vérifie si elle respecte les seuils de MIN_TEMP_AIR et MAX_TEMP_AIR.
- **lireHumidite()** : Lit l'humidité relative et vérifie si elle se situe dans les limites HYGR_MINT et HYGR_MAXT.
- **lirePression()** : Mesure la pression atmosphérique et vérifie si elle se trouve entre PRESSURE_MIN et PRESSURE_MAX.
- **collecterDonneesCapteurs()** : Regroupe les valeurs de chaque capteur et vérifie s'il est temps de les enregistrer en fonction de l'intervalle d'acquisition. Les données sont ensuite enregistrées sur la carte SD, avec une vérification de l'état de la carte.

Fonctions de gestion de l'affichage:

- **afficherDonneesCapteurs()** : Affiche les données en temps réel de chaque capteur sur le moniteur série.
- **afficherMaintenance()** : Appelée en mode maintenance pour afficher les données des capteurs à un intervalle défini (MAINTENANCE_INTERVAL).

Fonctions de gestion des boutons :

- **lireBoutons()** : Lit l'état actuel des boutons rouge et vert, calcule la durée d'appui, et détecte si un appui court ou long a été effectué. Cette fonction utilise les variables de temps et d'état (dernierTempsRouge, dernierTempsVert, etc.) pour distinguer les différentes interactions.
- **changerMode()** : Change le mode du système selon le type d'appui sur les boutons. Par exemple, un appui court pourrait basculer entre les modes de base, tandis qu'un appui long pourrait passer au mode configuration ou maintenance.

Fonctions de gestion des LEDs :

- **clignoter_LED()** : Gère le clignotement de la LED RGB en alternant les couleurs à un intervalle défini. Utilisée pour signaler des états (ex. mode économie, erreurs) en fonction de la couleur.
- **afficherErreur()** : Allume les LEDs dans une couleur spécifique pour indiquer une erreur. Par exemple, le rouge pourrait indiquer un problème avec la carte SD ou un capteur défaillant.

Fonctions de gestion de la carte SD:

- **initialiserCarteSD()** : Initialise la carte SD et crée un nouveau fichier si nécessaire. Vérifie également la compatibilité du système de fichiers et gère les erreurs potentielles.
- **enregistrerDonneesSurSD()** : Enregistre les lectures de capteurs dans un fichier sur la carte SD, avec un horodatage provenant de l'horloge RTC.

Fonctions de gestion de l'EEPROM :

- **sauvegarderParametres()** : Enregistre les paramètres des capteurs et des seuils dans l'EEPROM. Cette fonction est utilisée en mode configuration pour préserver les changements.
- **chargerParametres()** : Lit les paramètres stockés dans l'EEPROM pour appliquer les configurations des capteurs et des seuils au démarrage.

Fonction de boucle principale :

- **loop()** : Fonction qui s'exécute en continu, gérant les différentes tâches du système selon le mode actif. Elle appelle les fonctions pour lire les capteurs, gérer les boutons, enregistrer les données, et afficher les informations.