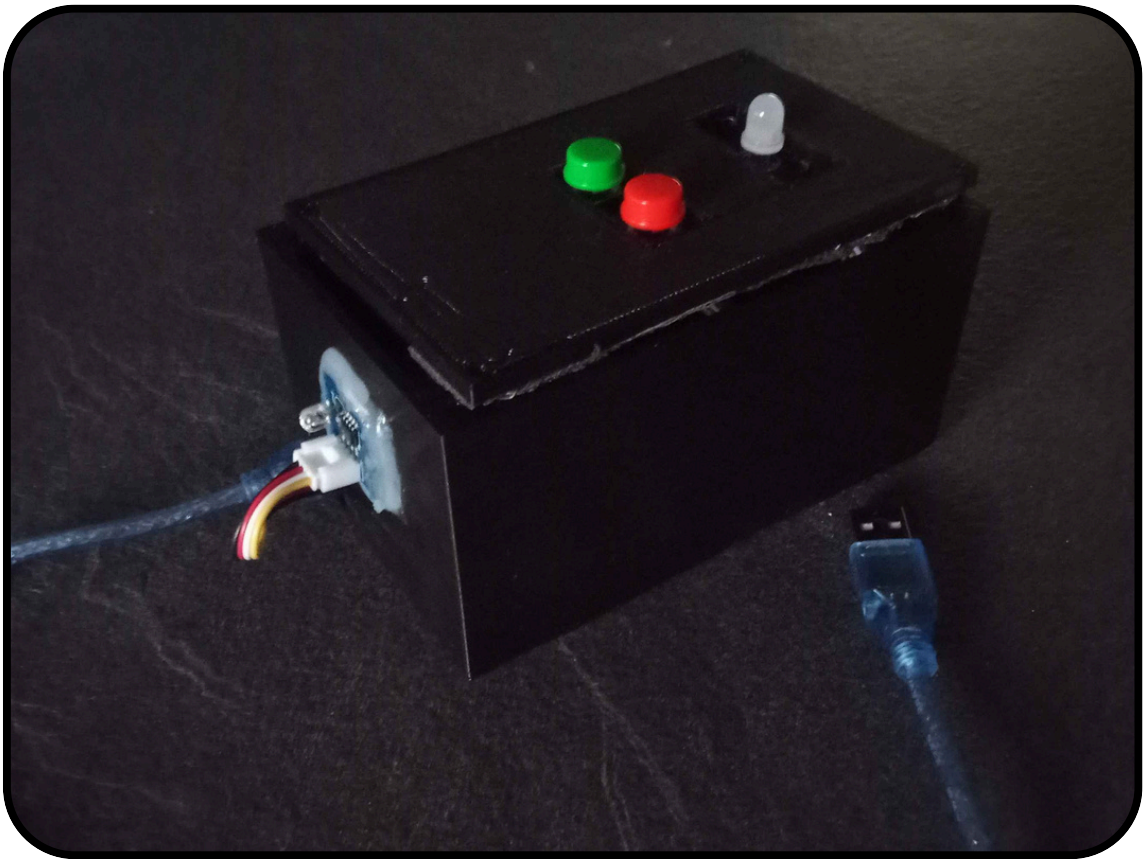
Documentation Technique Station T.H.S



Novembre 2024

1. Introduction

Le projet **Worldwide Weather Watcher** vise à surveiller les conditions environnementales en temps réel à l'aide de capteurs intégrés dans une station connectée. Cette station permet de recueillir des données météorologiques essentielles telles que la température, la pression atmosphérique, l'humidité et la luminosité, qui sont ensuite stockées et accessibles pour **analyse**.

1.2 Portée :

Le système est conçu pour des environnements nécessitant **une surveillance continue** des conditions atmosphériques. Les principales applications incluent les stations météorologiques locales, les installations sensibles aux conditions climatiques, ou les projets de recherche environnementale. Il est particulièrement adapté aux situations où une configuration et une maintenance minimales sont souhaitées, ainsi qu'aux contextes où les économies d'énergie sont importantes grâce à des **modes d'acquisition** de données optimisés.

1.3 Aperçu :

Ce document technique fournit une **description complète** du système Worldwide Weather Watcher. Il se divise en plusieurs sections :

1. **Fonctionnement global :** Explication du fonctionnement en différents modes (standard, économique, configuration, maintenance).
2. **Architecture du système :** Présentation des flux d'informations et de l'architecture matérielle, incluant les composants principaux.
3. **Composants matériels :** Détails des différents composants utilisés, comme l'Arduino Uno, les capteurs et le module de stockage.
4. **Conception logicielle :** Analyse des principales fonctions et de la gestion des modes dans le code source.
5. **Gestion des données :** Informations sur l'acquisition, le stockage et la structure des données enregistrées sur la carte SD.
6. **Gestion des LED et boutons :** Explication de l'utilisation des LED et des boutons pour signaler les états du système.
7. **Paramètres de configuration :** Liste des paramètres ajustables pour répondre à des besoins spécifiques.
8. **Maintenance et dépannage :** Indicateurs d'erreurs et solutions pour résoudre les problèmes rencontrés.
9. **Annexes :** Extraits de code, spécifications techniques, et références externes pour une consultation rapide.

2.Fonctionnement Global

Le système **Worldwide Weather Watcher** est une station connectée conçue pour surveiller et enregistrer des données environnementales essentielles. Son fonctionnement repose sur **quatre modes principaux** : le mode Standard, le mode Économique, le mode Configuration, et le mode Maintenance. Chaque mode offre des fonctionnalités spécifiques pour adapter le comportement de la station aux besoins opérationnels, qu'il s'agisse d'acquérir des données, d'économiser l'énergie, de configurer le système ou de **gérer les données** en toute sécurité.

2.1 Modes de fonctionnement

2.1.1 Mode Standard

- Le mode standard est le mode de fonctionnement **par défaut**, activé automatiquement au démarrage du système. Dans ce mode, le système acquiert les données des capteurs à intervalles réguliers (10 minutes par défaut). Les données recueillies sont enregistrées sur une carte SD sous forme de lignes horodatées, assurant ainsi une traçabilité continue des mesures. Si un capteur ne répond pas dans un délai défini (30 secondes par défaut), la valeur "NA" est enregistrée pour ce capteur. Le fichier de log sur la carte SD est **automatiquement segmenté** lorsque la taille limite (2 Ko par défaut) est atteinte, garantissant une gestion efficace de l'espace de stockage.

2.1.2 Mode Économique

- Accessible uniquement depuis le mode Standard, le mode Économique permet **d'optimiser la consommation énergétique** en réduisant la fréquence d'acquisition des données. Pour activer ce mode, l'utilisateur maintient le bouton vert pendant 5 secondes. Dans ce mode, l'intervalle entre deux mesures **est doublé**, ce qui prolonge l'autonomie du système en économisant la batterie. Le retour en mode Standard s'effectue également en maintenant le bouton vert pendant 5 secondes.

2.1.3 Mode Configuration

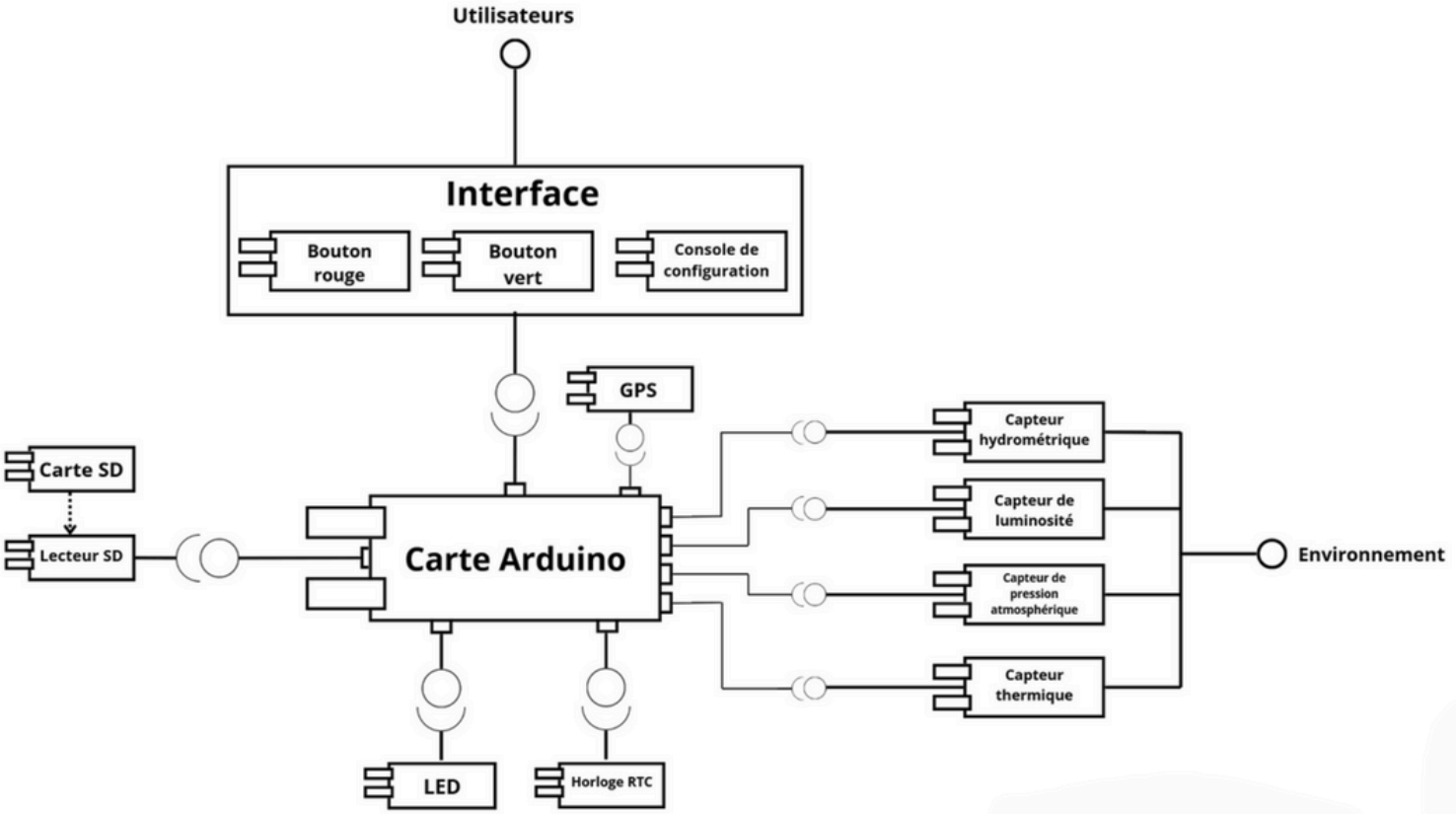
- Ce mode permet de **paramétrer le système** pour répondre aux besoins spécifiques de l'utilisateur. Il est accessible en maintenant le bouton rouge enfoncé lors du démarrage. Pendant ce mode, l'acquisition de données **est suspendue** et des commandes peuvent être envoyées via l'interface série pour modifier des paramètres tels que LOG_INTERVAL, FILE_MAX_SIZE, et TIMEOUT. Les paramètres sont stockés **dans l'EEPROM** pour une conservation **même en cas de redémarrage**. En l'absence d'activité pendant 30 minutes, le système revient automatiquement en mode Standard.

2.1.4 Mode Maintenance

- Le mode Maintenance est accessible depuis les modes Standard et Économique en maintenant le bouton rouge enfoncé pendant 5 secondes. Il permet de **consulter directement les données** des capteurs via une interface série et de **remplacer la carte SD** en toute sécurité, sans risquer de corrompre les données. Ce mode est particulièrement utile pour des interventions rapides, comme le diagnostic ou le changement de carte SD, avec un retour automatique au mode précédent en maintenant de nouveau le bouton rouge pendant 5 secondes.

3.Architecture du système

3.1 Diagramme de Composant :



3.2 Flux d'information :

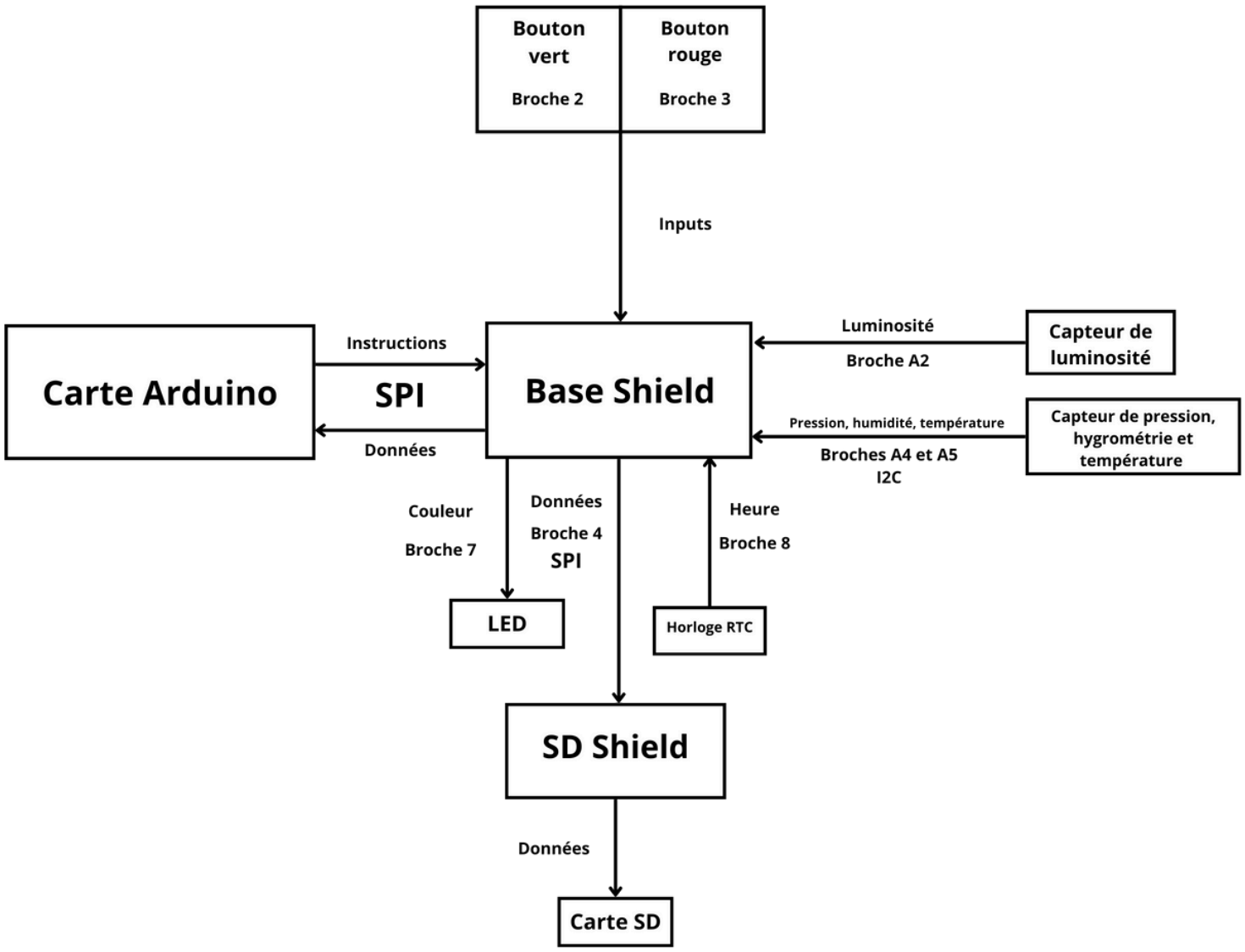
3.2.1 Acquisition des données : Les capteurs mesurent en continu les paramètres environnementaux (ex. température, humidité) et envoient ces valeurs au microcontrôleur.

3.2.2 Traitement des données : L'Arduino vérifie la cohérence des données et applique des règles prédéfinies (par ex : attendre TIMEOUT avant de considérer un capteur comme non disponible).

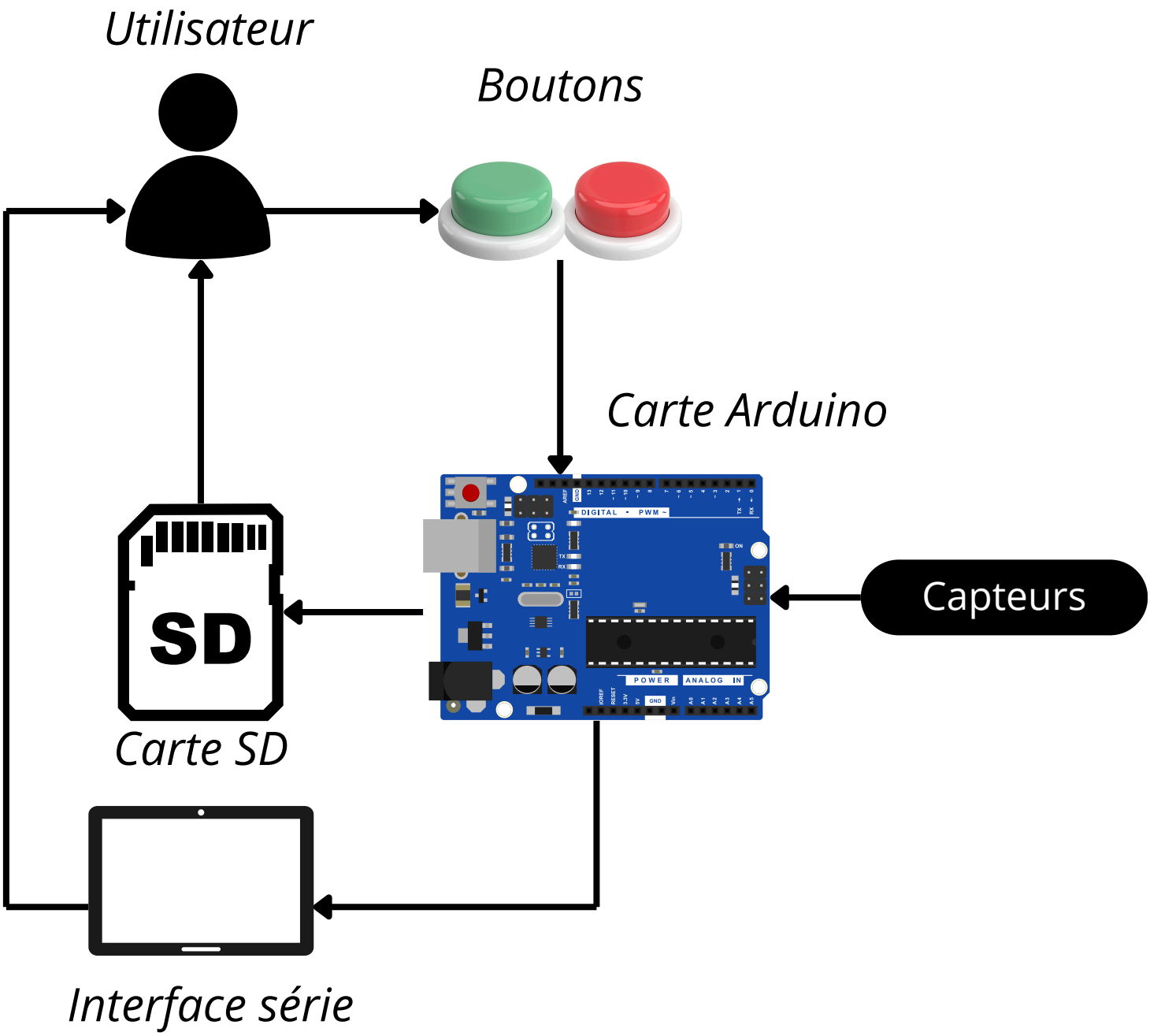
3.2.3 Enregistrement des données : Si les données sont valides, elles sont horodatées et enregistrées sur la carte SD dans un log. Lorsque la taille limite du fichier est atteinte, un nouveau fichier est créé.

3.2.4 Interaction avec l'utilisateur : En fonction du mode actif et des états d'erreur, l'Arduino utilise les LEDs pour informer l'utilisateur de l'état du système. Les boutons poussoirs permettent également de basculer entre les modes, influençant ainsi le flux de données par ex.

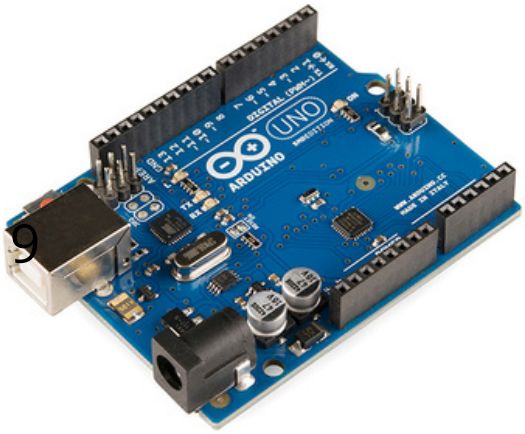
3.2.5 Diagrammes des flux d'information



3.3 Configuration matérielle :

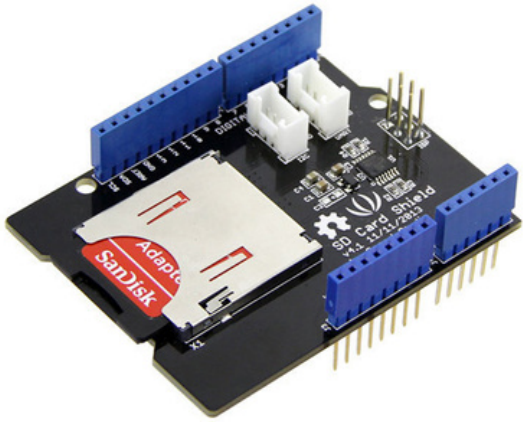


4. Composant Matériels

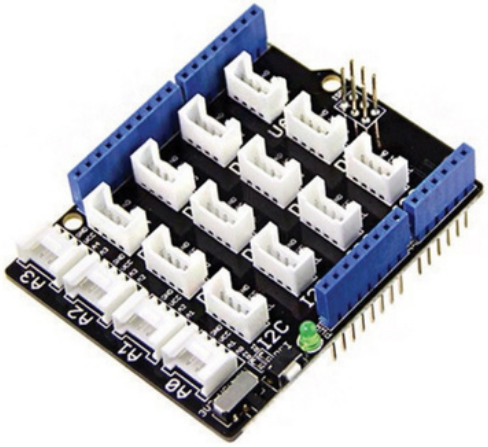


9

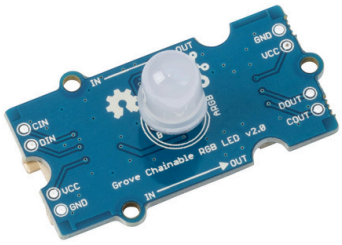
La carte Arduino Uno est basée sur un ATmega328 cadencé à 16 MHz. C'est la plus simple et la plus économique carte au microcontrôleur d'Arduino.



Le shield carte SD est une carte d'interface compatible Arduino permettant d'ajouter un espace de stockage. Il supporte les cartes SD et les cartes micro-SD (via adaptateur inclus).

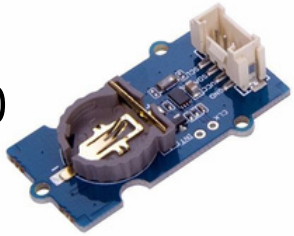


Le module Grove Base Shield est une carte d'interface permettant de raccorder facilement, rapidement et sans soudure les capteurs et les actionneurs Grove de Seeedstudio sur une carte compatible Arduino.



La LED RVB, permettant d’avoir plusieurs couleurs sur une seule LED

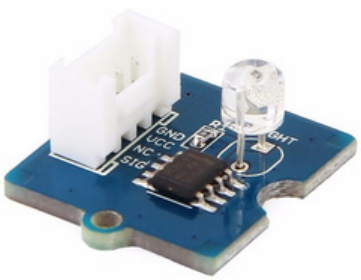
10



Module RTC de précision compatible Grove basé sur un PCF85063TP à faible consommation. Il donne la date et l'heure au format 12h et 24h, en tenant compte des années bissextiles.

Capteur Grove à double bouton poussoir permettant de contrôler deux canaux de signaux avec un seul module Grove.

Ce capteur est basé sur le circuit BME280 et mesure la température, l'humidité et la pression atmosphérique. Il communique avec un microcontrôleur type Arduino ou compatible via le bus I2C ou SPI.



Le Capteur de Lumière v1.2 Grove permet de mesurer le niveau de lumière

5.Conception Logicielle

5.1 Initialisation (Setup ()) :

La fonction setup() **initialise le système** et configure tous les composants. Elle commence par lancer la communication série à **9600 bauds**, puis initialise les LEDs (*initLED()*) et les boutons (*initButton()*). Elle récupère ensuite les paramètres enregistrés dans l'EEPROM, tels que HYGR et LOG_INTERVALL, pour les capteurs et la fréquence d'enregistrement. En **fonction de l'état** du bouton rouge, le système démarre soit en mode Configuration, soit en mode Standard.

Ensuite, la fonction teste l'initialisation des composants matériels :

- **Carte SD :** Si l'initialisation échoue, l'erreur **SD_WRITE_ERROR** est enregistrée.
- **Capteur BME280 :** Si le capteur est introuvable, l'erreur **SENSOR_DATA_ERROR** est signalée.
- **RTC :** Si le module RTC est introuvable, l'erreur **RTC_ERROR** est activée. Si le RTC n'est pas en cours d'exécution, il est ajusté à la date et l'heure actuelles.

5.2 Boucle principale du programme (loop())

La fonction loop() **exécute la logique continue** du système. Elle commence par vérifier et traiter les appuis des boutons via la fonction *handleButton()*. Selon le mode actif (*Standard, Économique, ou Maintenance*), le système effectue les opérations suivantes :

- **Modes Standard et Économique :** Si l'intervalle défini (*LOG_INTERVALL*) est écoulé, le système récupère les données des capteurs avec *recupDonnees()*, les affiche sur la console avec *afficherDonneesConsole()*, et les enregistre sur la carte SD via *writeDataToSD()*.
- **Mode Maintenance :** Les données sont affichées toutes les 5 secondes pour fournir une visibilité plus fréquente.

La fonction **met à jour les LEDs** en fonction de l'état et des erreurs avec *updateLEDs()*. Si le système est en mode Configuration, il écoute et interprète les commandes série via *handleSerialCommands()*.

5.3 Fonctions de modes

5.3.1 modStandard() : Active le mode Standard, dans lequel le système acquiert et enregistre les données des capteurs selon l'intervalle *LOG_INTERVALL*. Les paramètres personnalisés sont conservés sans réinitialisation de l'intervalle de journalisation.

5.3.2 modEco() : Active le mode Économique, doublant l'intervalle de collecte des données. Ce mode est particulièrement utile lorsque le système fonctionne sur batterie.

5.3.3 modMaintenance() : Change le mode en Maintenance, permettant d'accéder aux données en temps réel via l'interface série. Dans ce mode, la carte SD peut être retirée et insérée en toute sécurité, sans risque de corruption des données. La sortie de ce mode se fait en restaurant le mode précédent (Standard ou Économique).

5.3.4 modConfiguration() : Active le mode Configuration, où le système peut être configuré via des commandes série. Les paramètres modifiables incluent *LOG_INTERVALL*, *FILE_MAX_SIZE*, *TIMEOUT*, et plusieurs autres seuils de capteur. En cas d'inactivité prolongée (30 minutes), le système retourne automatiquement au mode Standard.

5.4 Gestion des Erreurs

Le système surveille en permanence les erreurs et utilise des LEDs pour signaler les problèmes. Ces erreurs sont codées par couleur et fréquence de clignotement de la LED, permettant **une identification rapide**. La gestion des erreurs inclut des messages sur l'interface série et des clignotements LED pour informer l'utilisateur et faciliter la résolution des problèmes.

Les erreurs possibles incluent :

- **RTC_ERROR :** Problème d'accès à l'horloge RTC, signalé par un clignotement spécifique de la LED.
- **SD_WRITE_ERROR et SD_CARD_FULL :** Problèmes d'accès ou de stockage sur la carte SD.
- **SENSOR_DATA_ERROR :** Erreur liée au capteur BME280.
- **GPS_ERROR :** Problème d'accès aux données GPS.
- **SENSOR_DATA_INCORRECT** : Données de capteurs incohérentes

6.Gestion des données et stockage

6.1 Acquisition des données

Les données sont collectées à **intervalles réguliers** à partir de plusieurs capteurs intégrés, en fonction du mode de fonctionnement actif. La fonction *recupDonnees()* gère cette acquisition en créant une nouvelle structure SensorData pour chaque ensemble de mesures, comprenant un horodatage (timestamp), la température, la pression, l'humidité et la luminosité. Selon les paramètres actifs, **chaque capteur est interrogé**, et si un capteur est désactivé ou ne répond pas dans le délai défini (TIMEOUT), sa valeur est remplacée par "NA" (Non Disponible). Les données acquises sont ensuite ajoutées à une liste chaînée pour un traitement ultérieur.

6.2 Stockage des données

Le stockage des données sur la carte SD est géré par la fonction *writeDataToSD()*. Le système enregistre **chaque ligne de données** dans un fichier au format .LOG, dont la taille est limitée par le paramètre *FILE_MAX_SIZE (2 Ko par défaut)*. Le nom du fichier comprend une date et un numéro de révision, par exemple, *200531_0.LOG* (format : année, mois, jour, numéro de révision). Lorsque **la limite de taille est atteinte**, le fichier est sauvegardé et un nouveau fichier avec un numéro de révision incrémenté est créé. En cas d'erreur d'écriture, un indicateur d'erreur (SD_WRITE_ERROR) est activé et le processus d'enregistrement est interrompu pour éviter toute corruption des données.

6.3 Structure des données

Chaque enregistrement de données **est structuré** comme suit :

- **Horodatage (timestamp)** : En secondes depuis l'époque Unix, permettant une traçabilité temporelle précise.
- **Température :** Valeur en degrés Celsius, captée par le capteur BME280.
- **Pression :** Mesurée en hectopascals (hPa) via le capteur de pression BME280.
- **Humidité :** Exprimée en pourcentage, elle provient également du capteur BME280.
- **Luminosité :** Valeur brute lue depuis le capteur de lumière analogique.

6.4 Communication série

Le système utilise **l'interface série** pour la configuration, le débogage et la visualisation des données. Lorsqu'il est en mode Configuration, l'utilisateur peut entrer des **commandes spécifiques** via la console série pour modifier des paramètres, tels que LOG_INTERVALL, FILE_MAX_SIZE, TIMEOUT, etc. Les messages de confirmation ou d'erreur sont renvoyés pour chaque commande, et les valeurs ajustées sont enregistrées **dans l'EEPROM** pour persister après un redémarrage. En mode Standard ou Maintenance, les données sont également affichées en temps réel sur la console série avec *afficherDonneesConsole()*, facilitant le suivi et le débogage sans nécessiter d'accès direct aux fichiers de la carte SD.

7. Gestion des LEDs et Boutons

7.1 Initialisation et mise à jour des LEDs

La gestion des LEDs dans le système est assurée par deux fonctions principales : **initLED()** et **updateLEDs()**.

- **initLED() :** Cette fonction initialise les LEDs pour indiquer les états du système. Elle configure les LEDs en affichant successivement différentes couleurs (rouge, vert, bleu, jaune) pour confirmer leur **bon fonctionnement au démarrage**. À la fin de l'initialisation, les LEDs sont éteintes, prêtes à afficher les couleurs correspondant aux états du système.

- **updateLEDs() :** Cette fonction met à jour les LEDs en fonction du mode actif ou des erreurs détectées. Si une erreur est présente (ex. RTC, GPS, écriture SD, capteur), des clignotements spécifiques de couleur sont utilisés pour indiquer le type de problème. En l'absence d'erreurs, les LEDs signalent le mode actif avec les couleurs suivantes :
  - **Vert continu :** Mode Standard.
  - **Bleu continu :** Mode Économique.
  - **Jaune continu :** Mode Configuration.
  - **Orange continu :** Mode Maintenance.

7.2 Interruptions des boutons

Les boutons vert et rouge permettent de **changer les modes du système**, et leur gestion est optimisée par des interruptions. Ces interruptions sont configurées dans la fonction setup() pour détecter les changements d'état de chaque bouton, et chaque bouton est **associé à une fonction d'interruption** dédiée (*greenButtonISR()* pour le bouton vert et *redButtonISR()* pour le bouton rouge).

- **Fonctions d'interruption :** Lorsqu'un bouton est pressé ou relâché, l'état est mis à jour dans la variable *buttonStates*. Cela permet de capturer les appuis longs ou courts, chaque appui étant traité en fonction de la durée (au moins 3 secondes pour changer de mode). Par exemple :

    - **Bouton vert :** Maintenir ce bouton pendant 3 secondes bascule entre le mode Standard et le mode Économique.
    - **Bouton rouge :** En maintenant ce bouton pendant 3 secondes, on passe au mode Maintenance ou Configuration selon le contexte.

- **Gestion des boutons (handleButton()) :** Cette fonction vérifie régulièrement l'état de *buttonStates* pour interpréter les appuis des boutons. Elle enregistre la durée d'appui et exécute la fonction de mode appropriée (ex. *modEco(), modMaintenance(), modStandard(), etc*.) selon le bouton et la durée d'appui.

8. Paramètres de configuration

8.1 Paramètres configurables par l'utilisateur

Le système WorldwidLe système Worldwide Weather Watcher propose plusieurs paramètres configurables qui permettent d'ajuster son fonctionnement en fonction des besoins spécifiques de l'utilisateur. Voici les paramètres principaux, leurs valeurs par défaut, ainsi que les plages de valeurs possibles le Worldwide Weather Watcher propose **plusieurs paramètres configurables** permettant d'ajuster le comportement du système en fonction des besoins de l'utilisateur :

LOG_INTERVALL -> Définit l'intervalle de temps entre deux acquisitions de données.

- *Valeur par défaut :* 600000 ms (10 minutes).
- *Plage de valeurs :* 60000 ms (1 minute) à 3600000 ms (1 heure).
- *Commande :* LOG_INTERVALL=<valeur en minutes> (ex. LOG_INTERVALL=5 pour un intervalle de 5 minutes).

TIMEOUT -> Durée maximale d'attente pour qu'un capteur réponde avant de considérer la donnée comme indisponible.

- *Valeur par défaut :* 30000 ms (30 secondes).
- *Plage de valeurs :* 1000 ms (1 seconde) à 60000 ms (1 minute).
- *Commande :* TIMEOUT=<valeur en secondes> (ex. TIMEOUT=20 pour un délai de 20 secondes).

FILE_MAX_SIZE -> Taille maximale (en octets) d'un fichier de journalisation sur la carte SD. Lorsqu'un fichier atteint cette taille, un nouveau fichier est créé.

- *Valeur par défaut :* 2048 octets (2 Ko).
- *Plage de valeurs :* 512 octets à 32768 octets (32 Ko).
- *Commande :* FILE_MAX_SIZE=<valeur en octets> (ex. FILE_MAX_SIZE=4096 pour une limite de 4 Ko).

LUMIN -> Active (1) ou désactive (0) le capteur de luminosité.

- *Valeur par défaut :* 1 (activé).
- *Valeurs possibles :* 0 ou 1.
- *Commande :* LUMIN=<0 ou 1>.

LUMIN_LOW et LUMIN_HIGH -> Définissent les seuils de luminosité pour considérer les niveaux de lumière comme faibles ou élevés.

- Valeur par défaut : LUMIN_LOW=255, LUMIN_HIGH=768.
- Plage de valeurs : 0 à 1023.
- Commande : LUMIN_LOW=<valeur> et LUMIN_HIGH=<valeur> (ex. LUMIN_LOW=200, LUMIN_HIGH=700).

TEMP_AIR -> Active (1) ou désactive (0) le capteur de température de l'air.

- Valeur par défaut : 1 (activé).
- Valeurs possibles : 0 ou 1.
- Commande : TEMP_AIR=<0 ou 1>.

MIN_TEMP_AIR et MAX_TEMP_AIR -> Seuils de température de l'air en dessous ou au-dessus desquels le capteur se met en erreur.

- *Valeurs par défaut :* MIN_TEMP_AIR=-10 °C, MAX_TEMP_AIR=60 °C.
- *Plage de valeurs :* -40 °C à 85 °C.
- *Commande :* MIN_TEMP_AIR=<valeur> et MAX_TEMP_AIR=<valeur>.

HYGR -> Active (1) ou désactive (0) le capteur d'hygrométrie.

- *Valeur par défaut :* 1 (activé).
- *Valeurs possibles :* 0 ou 1.
- *Commande :* HYGR=<0 ou 1>.

HYGR_MINT et HYGR_MAXT -> Plages de température au-delà desquelles les mesures d'hygrométrie ne sont pas prises en compte.

- *Valeurs par défaut :* HYGR_MINT=0 °C, HYGR_MAXT=50 °C.
- *Plage de valeurs :* -40 °C à 85 °C.
- *Commande :* HYGR_MINT=<valeur> et HYGR_MAXT=<valeur>.

PRESSURE -> Active (1) ou désactive (0) le capteur de pression atmosphérique.

- *Valeur par défaut :* 1 (activé).
- *Valeurs possibles :* 0 ou 1.
- *Commande :* PRESSURE=<0 ou 1>.

PRESSURE_MIN et PRESSURE_MAX -> Seuils de pression atmosphérique en dessous ou au-dessus desquels le capteur se met en erreur.

- *Valeurs par défaut :* PRESSURE_MIN=850 hPa, PRESSURE_MAX=1080 hPa.
- *Plage de valeurs :* 300 hPa à 1100 hPa.
- *Commande :* PRESSURE_MIN=<valeur> et PRESSURE_MAX=<valeur>.

DATE -> Permet de changer le jour le mois et l'année du système

- Valeur par défaut : date actuelle.
- Plage de valeurs : MOIS{1-12},JOUR{1-31},ANNEE{2000-2099}.
- Commande : DATE=<mois, jour, année choisis> (ex. DATE=10,23,2024).

DAY -> Permet de changer le jour de la semaine

- Valeur par défaut : jour de la semaine actuelle .
- Plage de valeurs : {MON,TUE,WED,THU,FRI,SAT,SUN}.
- Commande : DAY=<jour de la semaine> (ex: DAY=MON).

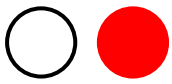
CLOCK -> Permet de changer l'heure, les minutes et les secondes du système

- Valeur par défaut : heure actuelle.
- Plage de valeurs : HEURE{0-23}:MINUTE{0-59}:SECONDE{0-59}.
- Commande : CLOCK=<heure, minutes, secondes choisie> (ex. CLOCK=12,59,23).

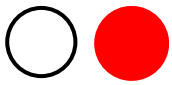
9. Maintenance et dépannage

9.1 Codes et indicateurs d'erreurs

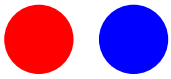
Le système **Worldwide Weather Watcher** utilise des LEDs pour signaler les erreurs et l'état général. Voici les codes d'erreurs principaux et leurs indicateurs lumineux, ainsi que les étapes de dépannage correspondantes :



Carte SD pleine

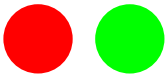


(blanc 2 fois plus long que le rouge) Erreur d'accès ou d'écriture sur la carte SD

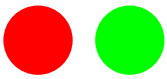


Erreur d'accès à l'horloge

Erreur d'accès au GPS



Erreur d'accès à un capteur



(vert 2 fois plus long que le rouge) Données reçues d'un capteur incohérentes

9.2 Réinitialisation des paramètres

La fonction *resetParameters()* est utilisée pour restaurer tous les paramètres du système à leurs valeurs par défaut.

Cette opération est utile **dans les situations suivantes** :

- Lorsqu'un mauvais paramétrage affecte le bon fonctionnement du système.
- Après des ajustements successifs de paramètres pour revenir aux réglages initiaux.
- En cas de problèmes de configuration non résolus par d'autres moyens.

Cette fonction **réinitialise des paramètres** tels que LOG_INTERVALL, TIMEOUT, FILE_MAX_SIZE, et les seuils de capteurs. Elle met également à jour l'EEPROM, garantissant que les valeurs par défaut **sont conservées** après un redémarrage.

9.3 Journaux système

Le système enregistre les données des capteurs dans des fichiers de journalisation (.LOG) sur la carte SD. Ces fichiers contiennent des **mesures horodatées** et sont segmentés en fonction de la taille maximale (FILE_MAX_SIZE) configurée, permettant de conserver un historique précis des mesures.

- Les journaux système peuvent être consultés :
  - **Via la console série** en mode Maintenance pour un accès direct aux données en temps réel.
  - **En récupérant** les fichiers de la carte SD pour une analyse hors ligne.

10. Annexes

10.1 Extraits de code

10.1.1 Initialisation du système (*setup()*) : Cette fonction initialise les composants et configure les paramètres du système.

```
void setup() {
  Serial.begin(9600);
  delay(1000);
  initLED();
  initButton();
  HYGR = EEPROM.read(HYGR_ADDR);
  isHygrActive = (HYGR != 0);
  EEPROM.get(LOG_INTERVALL_ADDR, LOG_INTERVALL);
  if (LOG_INTERVALL == 0xFFFFFFFF) { LOG_INTERVALL = 5000UL; }
  if (digitalRead(RED_BUTTON_PIN) == LOW) { modConfiguration(); } else { modStandard();
}
  // Vérification des composants
  if (!SD.begin(SD_CS_PIN)) { errorFlags |= SD_WRITE_ERROR; }
  if (!bme.begin(0x76)) { errorFlags |= SENSOR_DATA_ERROR; }
  if (!rtc.begin()) { errorFlags |= RTC_ERROR; } else if (!rtc.isrunning()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  attachInterrupt(digitalPinToInterrupt(GREEN_BUTTON_PIN), greenButtonISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(RED_BUTTON_PIN), redButtonISR, CHANGE);
}
```

10.1.2 Fonction d'acquisition des données (*recupDonnees()*)

- Fonction d'acquisition des données (*recupDonnees()*) : Récupère les données des capteurs et les structure pour l'enregistrement.

```
void recupDonnees() {
  SensorData* newData = (SensorData*)malloc(sizeof(SensorData));
  if (!newData) { Serial.println(F("Allocation mémoire échouée")); return; }
  newData->next = NULL;
  DateTime now = rtc.now();
  newData->timestamp = now.unixtime();
  newData->temperature = isTempAirActive ? bme.readTemperature() : NAN;
  newData->pressure = bme.readPressure() / 100.0F;
  newData->humidity = isHygrActive ? bme.readHumidity() : NAN;
  newData->luminosity = (isLuminActive && LUMIN == 1) ? analogRead(LIGHT_SENSOR_PIN) : -1;
  // Ajout aux données chaînées
  SensorData** ptr = &dataHead;
  while (*ptr) { ptr = &((*ptr)->next); }
  *ptr = newData;
}
```

10.1.3 Mise à jour des LEDs (*updateLEDs()*) : Gère les couleurs et clignotements des LEDs pour indiquer l'état du système et les erreurs.

```
void updateLEDs() {
  if (errorFlags & RTC_ERROR) {
    leds.setColorRGB(0, 255, 0, 0);
    delay(500);
    leds.setColorRGB(0, 0, 0, 255);
    delay(500);
  } else if (errorFlags & GPS_ERROR) {
    leds.setColorRGB(0, 255, 0, 0);
    delay(500);
    leds.setColorRGB(0, 255, 255, 0);
    delay(500);
  } else {
    switch (currentMode) {
      case STANDARD: leds.setColorRGB(0, 0, 255, 0); break;
      case ECO: leds.setColorRGB(0, 0, 0, 255); break;
      case MAINTENANCE: leds.setColorRGB(0, 148, 0, 211); break;
      case CONFIGURATION: leds.setColorRGB(0, 255, 255, 0); break;
    }
  }
}
```

10.2 Spécifications techniques

- **Arduino Uno :** Microcontrôleur ATmega328P fonctionnant à 5V et 16 MHz, consommant environ 50 mA.
- **Capteur BME280 :** Mesure la température (-40 à 85 °C), l'humidité (0 à 100%), et la pression (300 à 1100 hPa) avec une précision élevée.
- **Module RTC DS1307 :** Horloge temps réel pour la gestion de l'horodatage des données, fonctionnant avec une pile de secours.
- **Carte SD :** Utilisée pour stocker les données, supporte jusqu'à 2 Go en FAT32. Le paramètre FILE_MAX_SIZE est configuré pour créer des fichiers de journal segmentés.

10.3 Documentation de référence

- **Adafruit BME280 Library** : Documentation pour le capteur BME280.
- **RTClib** : Documentation pour l'horloge temps réel DS1307.
- **ChainableLED** : Documentation pour contrôler les LEDs RVB en chaîne.
- **SD Library** : Documentation pour l'utilisation de la carte SD avec Arduino.