2021

Documentation technique

Luc BARRETO, Benoit BLÉE, Clémence GIROMAGNY, Jules ROGÉ

THS

01/10/2021

Table des matières :

Introduction

Ce document a pour but de définir l'architecture du programme et d'expliquer les fonctions majeures de celui-ci.

Architecture du programme :

Bibliothèques :

Pour notre projet, nous utilisons les bibliothèques suivantes :

- Arduino.h
- EEPROM.h
- SparkFunBME280.h (permet d'utiliser les fonctions du capteur BME280)
- Wire.h
- DS1307.h (permet le fonctionnement de l'horloge RTC)
- SPI.h
- string.h
- ChainableLED
- SoftwareSerial.h
- math.h
- stdlib.h
- stdio.h
- SD.h
- BME280.h
- BME280I2C_BRZO.h
- BME280I2C.h
- BME280Spi.h
- BME280SpiSw.h
- EnvironmentCalculations.h

Nous incluons également des headers dans notre programme qui sont les suivants :

- configurationParametre.h
- decalage.h
- main.h
- moyenne.h
- structure.h

Variables globales :

configurationParametre.h :

- `#define LUMIN 1`
- `#define TEMP_AIR 2`
- `#define HYGR 3`
- `#define PRESSURE 4`

- `#define LOG_INTERVALL 5`
- `#define TIMEOUT 6`
- `#define VERSION 7`

- `#define MIN_TEMP_AIR 8`
- `#define MAX_TEMP_AIR 9`
- `#define HYGR_MINT 10`
- `#define HYGR_MAXT 11`

- `#define FILE_MAX_SIZE 100`
- `#define LUMIN_LOW 102`
- `#define LUMIN_HIGH 104`
- `#define PRESSURE_MIN 106`
- `#define PRESSURE_MAX 108`

- `uint8_t JOUR_SEMAINE`
- `unsigned char sortie,HEURE,MINUTE,SECONDE,JOUR,MOIS,lectureParametre`
- `unsigned int ANNEE`
- `signed int valeurParametre`

enregistrer.cpp

- `const int chipSelect = 10`
- `String date` Emplacement de la date
- `String myFileName`
- `unsigned char numrev`
- `File myFile`
- `DS1307 clock`

main.cpp

- `BME280I2C bme;`
- `SoftwareSerial SoftSerial(2, 3)`
- `unsigned char interMoyenne,interMoyenneInstante`
- `unsigned char mode` Indique le mode actuel
- `unsigned long logTemp`
- `unsigned long tempsPasse`
- `unsigned long varTemps=0`

- `struct capteur lum[n]` Emplacement des mesures du capteur de luminosité
- `struct capteur hygro[n]` Emplacement des mesures du capteur hygrométrique
- `struct capteur temp[n]` Emplacement des mesures du capteur de température
- `struct capteur pression[n]` Emplacement des mesures du capteur de pression
- `signed int valMoy` Valeur moyenne
- `bool t,erreur` Indique s'il y a une erreur
- `String gpsData` Emplacement des données GPS
- `unsigned int delayX`

main.h

- `#define bpRougePin 3`

- `#define bpVertPin 2`

- `extern unsigned long logTemp`
- `extern unsigned char mode`
- `extern unsigned long tempsPasse`
- `extern unsigned long varTemps`
- `String dataString`
- `ChainableLED leds(7, 8, 1)` Définition des pins de la LED

structure.h

- `const char n = 5;` Taille de tableau de mesures.

Fonctions :

- `void parametreParDefaut(void)`
- `void eeprom_reset(void)`
- `void eeprom_write_int(int addr, int n)`
- `int eeprom_read_int(int addr, bool s)`
- `void configurationParametre()`
- `void decalageValeur(struct capteur *Tab, int info, char n)`
- `void decalageErreur(struct capteur *Tab, int info, char n)`
- `void decalageMoyenne(struct capteur *Tab, int *info, char n)`
- `void enregistrer(String dataString)`
- `String getTime()`
- `String getTimeShort()`
- `void interruption()`
- `void bpRouge()`
- `void bpVert()`
- `void couleurled(unsigned char Couleur)`
- `void lectureCapteur(String *dataString, unsigned char mode)`
- `void moyenne(struct capteur *Tab, signed int *valMoy, char n)`
- `void moyenneMoyenne(struct capteur *Tab, char n)`
- `void printBME280Data (Stream* client)`

Explication fonctions/mécanismes :

Fonctionnement changement de mode :

La partie du programme qui permet de changer de mode est un « switch case » placé dans la fonction loop().

Suivant la variable « mode », on rentre dans un cas différent, soit un mode différent.

Voici la traduction de la valeur de mode.

mode = 1 -> Standard

mode = 2 -> Configuration

mode = 3 -> Economique

mode = 4 -> Maintenance

Pour rappel, nous devons pouvoir changer de mode en appuyant sur des boutons poussoirs (rouges et vert). Pour cela nous avons utilisé des fonctions d'interruptions (une pour le bouton vert et une pour le bouton rouge).

Ces fonctions sont « `void bpRouge()` » et « `void bpVert()` ». De ce fait lorsque l'on appuie sur un bouton une des interruptions s'active et change la valeur de la variable « mode ».

Cette méthode permet, seulement lorsque l'on rentre dans le « switch case », de valider quel mode on souhaite utiliser. Alors, on assure de ne jamais corrompre les données, car le mode ne change que lorsque toutes les actions sont terminées.

Fonctionnement lectureCapteur() :

La fonction « `void lectureCapteur(String *dataString, unsigned char mode)` »
permet premièrement de lire toutes les valeurs des capteurs (horloge, luminosité, température, …), de les stocker dans une le pointeur de la variable `dataString` (toutes les valeurs des capteurs sont mis bout à bout pour former une seule grande ligne de valeur), qui s'enregistrera enfin dans un fichier.

Cette première partie de la fonction permet de stocker les valeurs des capteurs dans la carte SD.

En parallèle de la première partie, on va venir insérer chaque valeur de capteur dans la fonction `void decalageValeur.`

decalageValeur()

La fonction « `void decalageValeur(struct capteur *Tab, int info, char n)` » permet de rentrer une valeur d'un capteur dans un tableau de valeur (en vue d'une future utilisation).

Le mécanisme de la fonction est de décaler toutes les valeurs du tableau (ici on manipule les valeurs du tableau via un pointeur, qui a une taille `n`) vers leur adresse précédente, ce qui à pour effet de d'écraser une valeur du tableau. Suite à ce décalage, la dernière adresse du tableau est de nouveau libre, alors on insère la nouvelle valeur (`int info`) obtenu via le capteur dans cette adresse libre.

La fonction peut être appelé autant de fois qu'il y a de nouvelle valeur à entrer, mais autant de d'anciennes valeurs seront retirées du tableau.

Les fonctions « `void decalageErreur(struct capteur *Tab, int info, char n)` » et « `void decalageMoyenne(struct capteur *Tab, int *info, char n)` » fonctionnent pareil, mais prennent quand à elles avec un tableau d'erreur et un tableau de moyenne.

Fonctionnement de couleurled() :

Dans notre projet, nous utilisons une LED un peu particulière : la « Grove - Chainable RGB LED ». C'est une LED RGB qui est gérée avec les paramètres RGB (de 0 à 255).

LED Utilisée dans notre projet

Pour pouvoir utiliser la LED RGB, nous avons besoin de la bibliothèque « ChainableLED ». Nous l'incluons en mettant les fichiers ChainableLED.cpp ChainableLED.h dans le répertoire du projet et en inscrivant « `#include "ChainableLED.h"` ».

Il y a ensuite une phase d'initialisation avec la définition des pins avec « `ChainableLED leds(7, 8, 1)` » puis avec « `leds.init()` » dans le `void setup()`.

Pour finir, il y a la déclaration d'une fonction nommée couleurled(). Cette fonction prend en entrée une variable de type char nommée Couleur.

```
void couleurled(unsigned char Couleur)
{
    switch(Couleur){
      case 'R' :
         leds.setColorRGB(0, 255, 0, 0);
         break;
      case 'V' :
         leds.setColorRGB(0, 0, 255, 0);
         break;
      case 'B' :
         leds.setColorRGB(0, 0, 0,255);
         break;
      case 'O' :
         leds.setColorRGB(0, 255, 80, 0);
         break;
      case 'J' :
         leds.setColorRGB(0, 255, 255, 0);
         break;
```

```
      case 'b' :
         leds.setColorRGB(0, 255, 255, 255);
         break;
   }
}
```

Fonction couleurled()

Cette fonction contient une condition `switch` qui prend en entrée la variable char Couleur. À l'intérieur de cette condition plusieurs cas sont définis. Chaque cas correspond à une couleur. Par exemple, pour la couleur rouge, si `Couleur` est égale à `'R'`, la fonction couleurled() exécute la commande `leds.setColorRGB(0, 255, 0, 0)` qui fait allumer la LED en rouge. La fonction `leds.setColorRGB(0, r, v, b)` a pour paramètre le numéro de la LED (ici 0 car il y en a qu'une), et les paramètres de couleur r, v et b qui vont de 0 à 255. Pour appeler la fonction, on utilise la commande `couleurled('Couleur')`, avec `'Couleur'` qui est remplacé par la couleur demandée. Par exemple, pour appeler la fonction et qu'elle allume la LED en rouge, la commande utilisée sera `couleurled('R')`.