

**Projet Worldwide Weather Watcher**

1. *Analyse du système*

**DATE :** Vendredi 2 Octobre 2020

**EMETTEUR(s) :** Eswann GIREAUD, Mattéo COFFIN, Théo SEROUSSI, Julie GILS

**DESTINATAIRE(s) :** Agence Internationale pour la Vigilance Météorologique (AIVM)

**OBJET :** Architecture du programme

**INFORMATION GENERALE**

Dans ce document, vous trouverez la structure générale du programme qui va être exécuté dans votre système embarqué. Elle sera composée d’algorithmes expliquant les différentes fonctions utilisées ainsi que leurs variables employées.

En attente de votre validation,

***L’équipe CESI***

Tout d’abord, le cahier des charges nous indiquait que la station météo devait fonctionner selon quatre modes : standard, configuration, maintenance et économique. Chacun d’eux permet d’effectuer des actions différentes. Afin de comprendre comment accéder à ces modes, voici en premier lieu les différentes fonctions expliquant les fonctionnalités de notre système :

**La fonction** rgb(r,g,b) **:**

*Entrées* :

Caractère non signé r, g et b et des entiers ledR , ledbB et ledV.

*Initialisation* :

Affecter à r, g et b les valeurs souhaitées.

*Traitement* :

Intensité par rapport à r sur le pin ledR.

Intensité par rapport à g sur le pin ledG.

Intensité par rapport à b sur le pin ledB.

Elle permet d’allumer une led qui, selon sa couleur, indiquera le mode dans lequel se trouve le système ou des potentielles erreurs.

**Les fonctions (1)**boutonRouge() **et (2)**boutonVert()**:**

1. *Initialisation* :

Utilisation de « buttonPin1 » en pinMode « INPUT\_PULLUP »

Utilisation des variables « redtimer », « greentimer » et « startred » de type « unsigned long »

*Traitement* :

**SI** buttonPin1 est pressé **ALORS**

**|** Affecter millis() à startred

**SINON**

**|** **SI** startred est différent de 0 **ALORS**

**|** **|**Affecter millis()-startred à redtimer

**|FIN**

**FIN**

1. *Initialisation* :

Utilisation de « buttonPin2 » en pinMode « INPUT\_PULLUP »

Utilisation des variables « redtimer »,« greentimer » et « startgreen » de type « unsigned long »

*Traitement* :

**SI** buttonPin1 est pressé **ALORS**

**|** Affecter millis() à startgreen

**SINON**

**|** **SI** startgreen est différent de 0 **ALORS**

**|** |Affecter millis()-startgreen à greentimer

**|FIN**

**FIN**

Ces deux fonctions permettent de compter depuis combien de temps les boutons rouge et vert sont pressés. Elles servent notamment lors des changements de modes qui nécessite de rester appuyé sur un des boutons durant cinq secondes.

**La fonction** Tempo()**:**

*Initialisation* :

Déclarer la variable MillisEnreg de type unsigned long.

MillisEnreg prend comme valeur le temps écoulé depuis le lancement du programme.

Récupération de la variable Attente qui était en paramètre de fonction.

Utilisation de la variable CutInterrupt de type bollean.

Utilisation de la variable Millis de type long.

*Traitement :*

**TANT QUE (** Millis **<** MillisEnreg + Attente\*CutInterrupt **)**

**FAIRE**

**|**Rien

**FIN**

CutInterruptprend la valeur 1 ;

**La fonction** TchangMod() **:**

*Initialisation :*

Récupération de la variable Timer qui était en paramètre de fonction.

Récupération de la variable Rouge qui était en paramètre de fonction.

Utilisation de la variable CutInterrupt de type bollean.

Utilisation de la variable tps de type int.

*Traitement :*

**SI (** Timer > tps **OU (**mode = 1 **ET** Rouge = 1**) ) ALORS**

**|** CutInterruptprend la valeur 0 ;

**FIN**

**La fonction** interruption()**:**

*Traitement* :

Appel de la fonction attachInterrupt avec comme paramètre :

Interrutpion = digitalPinToInterrupt(buttonPin1),

Fonction = boutonRouge

Mode = CHANGE

Appel de la fonction attachInterrupt avec comme paramètre :

Interrutpion = digitalPinToInterrupt(buttonPin2),

Fonction = boutonVert

Mode = CHANGE

Une interruption stoppe momentanément un programme pour effectuer un autre travail puis retourne à l’exécution du programme lorsque ce travail est terminé. Elle est déclenchée par un évènement extérieur imprévisible : ici, il s’agit d’un appui de l’utilisateur sur un bouton.

**La fonction** resetTimer()**:**

*Initialisation* :

Utilisation des variables « redtimer » et « greentimer » de type int

*Traitement* :

Affecter 0 à redtimer

Affecter 0 à greentimer

Les timers servant à chronométrer le temps d’appui sur les boutons sont remis à zero pour une prochaine utilisation.

**La fonction** reinitialisation()**:**

*Entrées :*

Structure Terreur

*Initialisation :* Réinitialise les erreurs à 0

**La fonction** Erreur()**:**

Entrées :

Structure Terreur

Initialisation :

**SI** Erreur\_pression = 1 ou Erreur\_temp = 1ouErreur\_hygrométrie = 1  **ALORS**

**|** Affecter 5 à mode

**|** Appeler fonction rgb avec en paramètres : rouge et verte avec fréquence de 1Hz durée 2 fois plus longue pour le vert

**FIN**

**SI** ErreurHorloge = 1  **ALORS**

**|** Affecter 6 à mode

**|** Appeler fonction rgb avec en paramètres : rouge et bleue avec fréquence de 1Hz

**FIN**

**SI** stockage\_erreur = 1 **ALORS**

**|** Affecter 7 à mode

**FIN**

**|** Appeler fonction rgb avec en paramètres : rouge et blanche avec fréquence de 1Hz

**SI** erreur\_ecriture = 1 **ALORS**

**|** Affecter 8 à mode

**|** Appeler fonction rgb avec en paramètres : rouge et blanche avec fréquence de 1Hz durée 2 fois plus longue pour le blanc

**FIN**

**SI** erreur\_accés= 1 **ALORS**

**|** Affecter 9 à mode

**|** Appeler fonction rgb avec en paramètres : rouge et vert avec fréquence de 1Hz

**FIN**

Appeler fonction réinitialisation

Cette fonction permettra à la led de s’allumer selon le type d’erreur reconnu.

**La fonction** AddVal()**:**

Déclarer la variable data de type Tmesure

*Structure Tmesure :*

**|** Contient PressionAtm

**|** Contient TempAir

**|** Contient Hygrométrie

**|** Contient Luminosité

*Structure Terreur :*

**|** Contient Erreur\_pression

**|** Contient Erreur\_temp

**|** Contient Erreur\_hygrométrie

**|** Contient erreur\_accés

*Bibliothèque***:** librairie de Tmesure.h (structure permettant la mesure), Adafruit\_Sensor.h, Wire.h , SPI.h et Adafruit\_BME280.h (librairie permettant l’acquisition des données des capteurs) .

*Traitement***:**

**SI** PRESSURE= 1 **ALORS**

**|** data.PressionAtm **prend la valeur** Valeur pression du capteur.

**|SI** PRESSURE\_MIN > data.PressionAtm ou PRESSURE\_MAX < data.PressionAtm **ALORS**

**| |** affecter 1 àdata.erreur\_pression

**| |** data.PressionAtm **prend la valeur « na »**

**| FIN**

**FIN**

**SI** TEMP\_AIR = 1 **ALORS**

**|** data. TempAir **prend la valeur** Valeur température air du capteur.

**| SI** MIN\_TEMP\_AIR > data. TempAir ou MAX\_TEMP\_AIR < data. TempAir **ALORS**

**| |** affecter 1 àdata.erreur\_Temp

**| |** data. TempAir **prend la valeur « na »**

**| FIN**

**FIN**

**SI** HYGR = 1 **ALORS**

**|** data. Hygrométrie **prend la valeur** Valeur Hygrométrie du capteur.

**| SI** HYGR\_MINT> data. Hygrométrie ou HYGR\_MAXT < data. Hygrométrie **ALORS**

**| |** affecter 1 àdata.Erreur\_hygrométrie

**| |** data. Hygrométrie **prend la valeur « na »**

**| FIN**

**|** data. Luminosité **prend la valeur** Valeur Luminosité du capteur (fonction light\_Sensor).

**FIN**

**SI** valeur égale NULL **ALORS**

**|**  ‘ = 1

**FIN**

Nous avons une structure contenant la valeur de la Pression atmosphérique, la température de l’air, l’hygrométrie et la luminosité. Grâce à elle, nous pourrons écrire sur la carte SD les valeurs récupérées par les capteurs

**La fonction** Light\_Sensor**()** :

*Bibliothèque :*Librairie <Grove\_LED\_bar.h> (Permettant l’utilisation du capteur de luminosité).

*Entrée :*

« Valeur » entier en local

*Initialisation :*

Initialisation du capteur (paramètre)

*Traitement :*

**Si** LUMIN=1 **alors**

**|** Lire la valeur du capteur

**|** Affecter la valeur du capteur à valeur

**| Si** Valeur du capteur > LUMIN\_LOW **alors**

**| |** luminosité est considérée comme “forte”

**| FIN**

**| Sinon**

**| |** luminosité est considérée comme “faible”

**FIN**

**Sortie :** Retourne valeur

En sortie de cette fonction, nous trouvons l’intensité captée par le capteur de luminosité

**La fonction** RTC()**:**

*Bibliothèque :* Librairie Wire.h ( Cette bibliothèque vous permet de communiquer avec l’horloge RTC) et DS1307.h (le type d’horloge RTC).

*Entrée :*

ErreurHorloge en booléen

*Initialisation :*

Affecter ErreurHorloge à 0

*Traitement :*

Définir la date du jour et l’heure

Intégrer la date à l’horloge RTC

**Si** erreur horloge

**|** Affecter ErreurHorloge à 1

**FIN**

*Sortie :* Retourne la date

Parvenir à récupérer la date du jour nous servira pour nommer les fichiers contenant les valeurs des capteurs.

**La fonction** nom\_fichier**:**

*Entrées :*

Jour, mois, année en entier et date en char.

*Traitement :*

Prend les valeurs de temps de la fonction horloge RTC.

Les converties en caractère.

Intègre cela dans une variable date avec .txt à la fin.

*Sortie :*  Retourne le nom du fichier

Comme dit précédemment, le nom du fichier sera composé de la date du jour auquel l’acquisition des données a été faite : cette fonction sert donc a mettre en forme le nome du fichier avec son extension.

**La fonction** SD() **:**

*Bibliothèque***:** Librairie de cartes SD et Librairie de <avr/pgmspace.h> :

*Entrées :*

La constante « donne » en int, la date, le pinSD en int, le stockage\_erreur et erreur\_ecriture en booléen.

***Initialisation****:*

Affecter à pinSD la valeur 10

Affecter à Carte SD la valeur myFile

*Traitement :*

**SI** la carte SD est branchée **ALORS**

**|** afficher « la carte SD est prête »

**|** Définir le fichier d’écriture par la date ou créer un fichier date (utiliser fonction nom\_fichier)

**| Si** mode\_SD != 2

**| | Pour** n allant de n donne **faire**

**| | |** Ecrire dans le fichier la donne n (de la fonction addVal)

**| | FIN**

**| Sinon**

**| | Pour** n allant de n donne alternance avec n allant de donne **faire**

**| | |** Ecrire dans le fichier la donne n (de la fonction addVal)

**| | FIN**

**| FIN**

**FIN**

**Sinon**

**|** afficher problème inconnu

**|**  erreur\_ecriture = 1

**|** Return

**FIN**

**Si** Stockage plein

**|** affecter stockage\_erreur à 1

**FIN**

Cette fonction est appelée pour inscrire les données sur la carte SD et détecter un stockage plein ou une autre erreur inconnue.

**Les fonctions modes**

Ici, nous verrons donc les quatre fonctions correspondant aux quatre modes possibles. Dans chacune d’elles, vous pourrez trouver toutes les fonctionnalités attendues, et les divers appels aux fonctions citées plus haut.

**La fonction** Standard()**:**

*Initialisation* :

Utilisation de la variable « mode » de type int

*Traitement* :

Affecter 1 à mode

Appeler fonction rgb avec en paramètres : (255,0,255)

Appeler fonction AddVal

Appeler fonction SD

Afficher « mode standard »

Appeler fonction resetTimer()

Appeler fonction erreur()

**La fonction** Maintenance()**:**

*Initialisation* :

Utilisation de la variable « mode » de type int

*Traitement* :

Affecter 2 à mode

Appeler fonction rgb avec en paramètres : (100,220,255)

Appeler fonction AddVal

Appeler fonction Affiche

Afficher « mode maintenance »

Appeler fonction resetTimer()

Appeler fonction erreur()

**La fonction** Economique()**:**

*Initialisation :*

Utilisation de la variable « mode » de type int

*Traitement :*

Affecter 3 à mode

Appeler fonction rgb avec en paramètres : (255,255,0)

Multiplication de 2 à LOG\_INTERVAL tant que le système est dans ce mode

Affecter 2 à mode\_SD

Afficher « mode economique »

Appeler fonction resetTimer()

Appeler fonction erreur()

**La fonction** Configuration()**:**

*Entrées :*

Toutes les options en int

*Initialisation :*

Utilisation de la variable « mode » de type int

*Traitement :*

Affecter 4 à mode

Appeler fonction rgb avec en paramètres : (0,0,255)

Affecter le choix d’utilisateur aux options

Afficher « mode configuration »

Appeler fonction resetTimer()

Appeler fonction erreur()

**Pour finir, voici la boucle exécutée par le système :**

Initialisation :

Utilisation des variables « mode » et « etat\_precedent » de type « int »

Utilisation des variables « redtimer », « greentimer » et « startred » de type « unsigned long »

Utilisation de la constante « tps » égale à 5000

Traitement :

**SI** mode est égal à 0 **ALORS**

**|** Appeler fonction Standard()

**FIN**

**SI** mode est égal à 1 **ALORS**

**|SI** redtimer est supérieur ou égal à tps **ALORS**

**| |**Appeler fonction Maintenance()

**| |**Affecter 1 à etat\_precedent

**|FIN**

**|SINON SI** greentimer est supérieur ou égal à tps **ALORS**

**| |**Appeler fonction Economique()

**|FIN**

**|SINON SI**  redtimer est compris entre 1 et tps **ALORS**

**| |**Appeler fonction Configuration()

**|FIN**

**|SINON** Appeler fonction AddVal()

**|**Appeler fonction Tempo() avec le temps à attendre en paramètre

**|FIN**

**FIN**

**SI** mode est égal à 2 **ALORS**

**|SI** redtimer est supérieur ou égal à tps **ALORS**

**| |SI** etat\_precedent est égal à 1 **ALORS**

**| | |**Appeler la fonction Standard()

**| |FIN**

**| |SINON SI** etat\_precedent est égal à 3 **ALORS**

**| | |**Appeler la fonction Economique()

**| |FIN**

**|FIN**

**FIN**

**SI** mode est égal à 3 **ALORS**

**| SI** greentimer est supérieur ou égal à tps **ALORS**

**| |**Appeler la fonction Standard()

**|FIN**

**|SINON**  **SI** redtimer est supérieur ou égal à tps **ALORS**

**|** **|**Appeler la fonction Maintenance()

**|** **|**Affecter 3 à etat\_precedent

**|FIN**

**|SINON** Appeler fonction AddVal()

**|**Appeler fonction Tempo() avec le temps à attendre en paramètre

**|FIN**

**FIN**

**SI** mode est égal à 4 **ALORS**

**|SI** greentimer > tps **ALORS**

**| |** Appeler la fonction Standard()

**|FIN**

**|SINON SI** TimerConfig = 0 **ALORS**

**| |** TimerConfig = Millis

**|FIN**

**|SINON SI** Millis > TimerConfig + 10sec **ALORS**

**| |** Appeler la fonction Standard()

**| |** TimerConfig prend la valeur 0

**|FIN**

**FIN**

**SI** mode est égal à 5 / 6 / 7 / 8 / 9 **ALORS**

**|**Appeler la fonction Erreur()

**|**Attendre 15 min

**|**Appeler la fonction Configuration()

**FIN**

FIN DU DOCUMENT