La station Météo de l'ASAP Lille

Le 3 octobre 2023

La station météo est composée d'un module central et de plusieurs capteurs. Actuellement il existe 6 types de capteurs qui peuvent être connectés à la station météo. Cette station propose un point d'accès WIFI qui pourra être utilisé pour suivre les remontées d'information des capteurs dans une page web. Il peut optionnellement être connecté à internet pour initialiser la date et l'heure du jour de sa mise sous tension.
Dans le cas où la station n'est pas connectée à internet, la date et l'heure courante sera initialisée en chargeant la page web à cette adresse IP : 192.168.4.1

L'aspect matériel

Le module central de la station météo est composé d'une carte ESP32-CAM qui est enfichée dans une autre carte qui est utilisée pour la connexion USB (POPESQ® 1 pcs. x ESP32-CAM-MB Programmateur #A4444) pour y injecter le programme et pour l'alimenter. A noter que la fonction CAM de la carte n'est pas utile mais il a été choisi car il embarque une carte SD.
Ce module central est lui est même organisé en plusieurs modules logiciels ayant chacun leur utilité:

- Module FILE : il regroupe les fonctions qui accèdent aux fichiers de l'application
- Module WIFI : il regroupe les fonctions qui permettent la connexion au réseau WiFi et qui gérent la date et l'heure
- Module ESP_NOW : il regroupe les fonctions qui communiquent avec les capteurs
- Module BOARD : regroupe les fonctions qui gèrent les données d'un capteur
- Module MESURES : il regroupe les fonctions qui gèrent les mesures envoyées par les capteurs
- Module WEBSERVER_MESURES : Il est chargé de répondre à la requête « /getMesures » qui permet d'envoyer une suite de mesures à un client HTTP

Il faudra prévoir un chargeur de téléphone avec prise USB pour alimenter le module central de la station météo.

Le module central prend en charge la gestion des requêtes HTTP envoyées par les navigateurs.
Liste des requêtes définies dans la version actuelle :

- « / » : Retourne le tableau de bord (html et javascript)
- « /sendEvents » : Demande à la carte de renvoyer pour chaque capteur les dernières mesures reçues
- « /getNbMesures » : Renvoie le nombre de mesures contenues dans la carte SD
- « /reset » : Réinitialise le serveur web
- « /getEpochTime » : Renvoie la date et l'heure
- « /setEpochTime?epochtime=xxx » : Définit la date et l'heure du serveur à la valeur du paramètre epochtime
- « /getMesures?from=xxx&to=yyy » : renvoie la suite des mesures à partir de la position définie par la paramètre from (début du fichier si absent)jusqu'à celle définie par le paramètre to (fin du fichier si absent)

L'affichage de la page web est paramétré par deux fichiers présents sur la carte SD :

- **index.html** qui contient le cœur de la page web avec le tableau et les javascripts qui vont rendre cette page dynamique en fonction des données reçues par le serveur web. Le container «%boards% » est remplacé par le code html de chacune des mesures définie dans la configuration des capteurs.
- **board.html** qui contient le fragment html qui décrit comment s'affiche une mesure envoyée par un capteur. Dans ce fichier, apparaissent les variables %id%, %owner %, %typemesure%, %unite% et %title% propres à chaque type de mesure (voir module mesure plus loin dans ce document). Bien entendu, entre les caractères % nous retrouvons le nom de chaque variable définie dans le fichier **board.csv**

Le module FILE

Ce module regroupe les fonctions qui accèdent aux fichiers de l'application. Dans sa version actuelle, ce module gère les fichiers stockés sur une carte SD.
Il offre une interface composée des fonctions :

- bool setup_File(); // initialise l'acès à la carte SD
- void print_File(char * path);
- String readHtml_File(char * filename); // Retourne le contenu du fichier défini par l'argument filename
- bool testSDCard_File(); // Retourne l'état actuel de la carte SD (pour savoir si elle est opérationnelle)
- bool getSDstatus_File(); // Retounr le dernier état connu de la carte SD
- void append_File(const char * message) ; // ajoute le contenu de message à la fin du fichier mesures.csv
- void writeLog_File(String message); // érit le message à la fin du fichier log.csv

La présence des fichiers /config.txt et /mesures.csv est indispensable au bon fonctionnement de ce module.
Le fichier /mesures.csv contient une suite de lignes et chacune des lignes contient la description d'une mesure au format CSV (le caractère ; est utilisé comme séparateur de champs)

Le module WIFI

Ce module regroupe les fonctions qui permettent d'initialiser la configuration de l'accès WiFi et celles qui gèrent la date et l'heure. Les données utilisées pour la configuration de l'accès WiFi sont stockées sur la carte SD.
Elles sont définis dans les 4 premières lignes du fichier « /**config.txt** » qui est préparé pour un fonctionnement sans connexion à internet :

1. Nom du réseau de la station : valeur initiale dans le fichier **stationMeteo**
2. **SSID** de la box WIFI du domicile : ligne vide si pas de connexion internet ou ssid de la box
3. **Mot de passe** de la box WIFI : ligne vide si pas de connexion internet
4. **Nom du serveur NTP** utilisé pour récupérer la date et l'heure courante : ligne vide si pas de connexion internet ou **pool.ntp.org**

En voici la liste des fonctions :

- bool setup_Wifi(); // initialise la connexion WiFi
- bool isConnected_Wifi(); // Indique si le WiFi est connecté
- String getBoxSsid_Wifi(); // retourne le ssid de la box qui donne l'accès à internet
- int getAlarmLedPin_Wifi(); // Retourne le numéo de la pin qui indique l'éatt du

réseau WiFi
- int getChannel_Wifi(); // Retourne le numéro du canal utilisé par le WiFi
- bool shouldBeConnected_Wifi(); // Indique si la carte devrait connectéeau WiFi
- // interface pour la date et l'heure
- unsigned long get_EpochTime_Wifi(); // Retourne la valeur de la date et de l'heure
- void set_EpochTime_Wifi(unsigned long ep); // positionne la valeur de la date et de l'heure
- unsigned long getCurrentTime_Wifi(); // Retourne la date et l'heure actuelle
- String timeToString_Wifi(unsigned long _t); // Retourne sous forme de chaine de caractères lisible par un humain la date et l'heure passée par l'argument t

Le module ESP_NOW

Ce module définit la fonction qui gère à l'aide de plusieurs autres fonctions privées la connexion de Type ESP-NOW entre la carte centrale et les capteurs :

- bool setup_ESP_NOW(); // initialise le protocole de communication ESP-NOW de la carte

Détails utiles pour coder un capteur

Fonctionnement interne du module ESP_NOW

Le module ESP-NOW collecte tous les messages envoyés par chacun des capteurs.
Les capteurs envoient 2 types de message :

- un message de demande d'apairage sur le canal WIFI utilisé par le module central (type de message = 0)
- un message contenant des mesures (type de message = 1)

Le module central envoie un seul de type de message pour répondre positivement à la demande d'appairage
La structure de chacun de ces messages est ainsi définie :
Message de demande d'apairage sur le canal WIFI :

```
typedef struct struct_pairing {
    uint8_t msgType;
    uint8_t id;
    uint8_t macAddr[6];
    uint8_t channel; // a été rendu non obligatoire !
} struct_pairing;
```

Pour répondre positivement à une demande d'apairage, le module central envoie la même structure de message avec msgType=1 et indique son adresse mac et le numéro de canal qu'il utilise.
Voici la structure du message envoyé par les capteurs pour envoyer des mesures :

```
// taille maximale = 250
// ici taille = 18 octets
typedef struct struct_message {
  uint8_t msgType; // 1 octet
  uint8_t id; // 1 octet
  unsigned int seqNum; // 4 octets → n'est pas exploitée dans cette version (mais elle est stockée sur la carte SD)
  float valeurs[10]; // taille = 10 * 4 = 40 octets  -> on pourrait avoir float valeurs[60]
} struct_message;
```

Une fois un message de mesures reçu, l'application, crée la ou les mesures. Une mesure est créée pour chaque valeur envoyée.

Ensuite chaque mesure est stockée sur la carte SD et la page web est mise à jour par exécution d'un code javascript déclenché par le module central.
Le module central peut actuellement gérer jusqu'à 10 mesures par capteur, mais les tests ont jusqu'ici été limité à 3 mesures.

Le module BOARD

Chacun des capteurs est représenté par une variable de type BOARD. La configuration des l'ensemble des capteurs est stockée dans le fichier au format CSV (dont le caractère ; est le séparateur de champ) **/boards.csv**.
Chaque ligne du fichier **boards.csv** contient la description d'un capteur. Ce fichier est modifiable soit par un éditeur de texte (comme Notepad++) soit par calc ou excel. Les différents champs d'une ligne sont dans cet ordre :

- id : l'identifiant du capteur (une valeur numérique de 1 à …)
- owner : le nom du responsable de ce capteur
- typemesures : une chaîne de caractères composée d'une suite d'un à trois types de mesure (typemesure) séparé par le caractère |. Chaque typemesure est identifié par un seul caractère.
- Unites : une chaîne de caractères composée d'une à trois unités (unite) séparées par le caractère |. Chaque unité est elle-même une chaîne de caractères qui ne doit pas contenir ni le caractère | ni le caractère ;
- titles : une chaîne de caractères composée d'un à trois « titles » (title) séparés par le caractère |. Chaque unité est elle-même une chaîne de caractères qui ne doit pas contenir ni le caractère | ni le caractère ;

Note : ces noms de champs sont utilisés dans le fichier board.html (lors de l'affichage d'une mesure dans une page web)

Voici la liste des caractères qui ont été choisis arbitrairement pour définir chacun un type de mesure :

- a : vitesse du vent
- b : pression
- g : sens du vent
- h : hygrométrie
- n : niveau d'eau de pluie
- t : température
- c : niveau charge batterie

Par exemple, la carte du Pluviomètre gérée par le capteur de Claude qui envoie une seule mesure de niveau d'eau recueillie par le capteur est représentée par cette ligne :

3;Claude;n;mm;Pluviometre

Pour la carte gérée par Philippe qui envoie la température, la pression et l'hygrométrie, la ligne est :

1;Philippe;t|p|h;C|b|%;temperature|pression|hygro

Le module BOARD prépare, lors de son initialisation, le contenu de la page HTML utilisée par le module principal.
Il regroupe les fonctions suivantes :

- bool setup_Board(); // initialise la description des capteurs à partir du fichier boards.csv
- char getBoardValueAsChar(board *_board, int index); // Retourne le type de mesure à la position définie par l'argument "index" du capteur board
- int getMaxBoards(); // Retourne le nombre de capteurs qui ont été définis dans le fichier boards.csv
- board * getBoardById(int boardId); // Retourne le capteur qui possède l'id défini par l'argument boardId
- board * getBoardByIndex(int index); // Retourne le capteur à la position définie par l'argument index
- String getIndex_html(); // Retourne la page html principale du tableau de bors des capteurs
- String getBoardString(String myString, int index);
- String boardToString (board _board); // Retourne une chaine de caractère représentant la description du capteur correspondant à l'argument _board
- int getNumberOfValues(String data); // Retourne le nombre de valeurs définies dans le descripteur data

Le module MESURES

Ce module regroupe les fonctions qui permettent de gérer la visualisation et le stockage des mesures envoyées par les capteurs :

- bool setup_Mesures(); // initialise la liste des mesures qui seront manipulées par la carte
- unsigned long getNextIdMesure(); // Retourn l'id de la prochaine mesure
- mesure* creerMesure(board *_board, unsigned int seqNum, float value, int index); // crée et renvoie une mesure de valeur value envoyée par une carte à la position index
- ListeMesures * getListMesures(); // Retourne la liste des mesures
- void ajouterMesure(mesure * p_mesure); // Ajoute une mesure à la liste
- void storeMesure(); // stocke la première mesure de la liste et la retire de cette liste

Le module WEBSERVER_MESURES

Il est chargé de répondre à la requête **/getMesures?from=xxxx&to=yyyy**
Si from est absent, il est défini à 0
Si to est absent, toutes les mesures depuis celle définie par **from** jusqu'à la fin du fichier sont envoyées au navigateur

- void resetServerBusy(); // Force l'état du serveur à prêt
- AsyncWebServerResponse * fillMesures(AsyncWebServerRequest *request); // renvoie les mesures comprises entre la position définie par le paramètre "from" et celle définie par le paramètre "to" tous les portés par la requêtes /getMesures?from=xxx&to=yyy