Annexe T.3

```
1   //  CommunicationServeur.ino
2   //
3   //  Description : fonctions liées à la communication entre l'ESP et le serveur
4   //  Auteur : Perret Mélissa
5   //  Création :  22/09/2024
6   //  Modifications :  --
7
8   //  Version     :   V1.0
9   /*-------------------------------------------------------*/
10
11
12  #include "CommunicationServeur.h"
13  #include <Arduino_JSON.h>  // Librairie pour manipuler et traiter les données JSON
    reçues par le serveur
14  #include <HTTPClient.h>    // pour HTTPClient
15
16
17  //// Fonction ReceptionServeur: logique pour recevoir les données de la part du STM
18  //// Description: envoie d'une requête HTTP Get au serveur
19  ////              réception de la réponse, analyse de la réponse pour extraire les
    données reçues (seuils min/max et écarts pour température et humidité)
20  ////              appel de la fonction TraiterValeurServeur pour traiter les données
    reçues
21  //// Entrées: Pointeur:HTTPClient http (client http utilisé pour effectuer les
    requêtes auprès du serveur)
22  //// Sorties: -
23  void ReceptionServeur(HTTPClient* http) {
24
25    if (MODE_DEBUG) {
26      printf("[HTTP] Requete GET...\n");
27    }
28
29    http->setTimeout(HTTP_TIMEOUT_MS);
30
31    int httpCode;
32    int nombreTentativesHTTP = 0;
33
34    do {
35
36      if(nombreTentativesHTTP > 0)
37      {
38        delay(DELAI_NOUVELLE_TENTATIVE_HTTP_MS); // S'il s'agit d'une nouvelle
          tentative, attendre un peu avant de réessayer
39      }
40
41      nombreTentativesHTTP++;
42
43      // Démarrer la connexion et envoyer la requête GET
44      int httpCode = http->GET();
45
46      // HttpCode est négatif en cas d'erreur
47      if (httpCode > 0) {
48        // Le header HTTP à été envoyé et une réponse du serveur à été reçue
49        if (MODE_DEBUG) {
50          printf("[HTTP] Requete GET... code: %d\n", httpCode);
51        }
52
53        if (httpCode == HTTP_CODE_OK) {
54          String reponseServeur = http->getString(); // Obtenir la réponse du serveur
55
56          if (MODE_DEBUG) {
57            printf("[HTTP] Requete GET reponse serveur: %s\n", reponseServeur.c_str());
58          }
59
60          // Parsing JSON
61          JSONVar myObject = JSON.parse(reponseServeur);
62          JSONVar keys = myObject.keys();
63
64          // Traitement des valeurs
65          for (int i = 0; i < keys.length(); i++) {
66            String nom = keys[i];
67            double valeur = std::atof(myObject[keys[i]]);  // utilisation de la
              fonction atof pour convertir de string en double
```

```
68              TraiterValeurServeur(nom, valeur);
69            }
70          }
71        } else {
72          printf("[HTTP] GET... echec, erreur: %s\n", http->errorToString(httpCode).c_str
            ());
73        }
74      } while (httpCode <= 0 && nombreTentativesHTTP < NOMBRE_TENTATIVES_REQUETES);
75    }
76
77    //// Fonction TraiterValeurServeur: logique pour traiter une donnée reçue par le
      serveur
78    //// Description: vérifier si la valeur à changé
79    ////              mettre à jour la valeur stockée dans la RTC
80    ////              construire et transmettre la trame UART au STM
81    //// Entrées: String nom (nom du paramètre modifié)
82    ////          double valeur (valeur du paramètre modifié)
83    //// Sorties: -
84    void TraiterValeurServeur(String nom, double valeur) {
85      int8_t index = IndexValeurServeur(nom);  // Récupérer index du paramètre modifié
        (-1 si non trouvé)
86      if (index == -1) {
87        printf("Nouvelle valeur provenant du serveur. Impossible de trouver le nom %s
          dans nomValeursServeur\n", nom.c_str());
88      } else if (valeur != valeursServeur[index]) {  // Si la valeur n'a pas changé,
        inutile d'en informer le STM
89
90        if (MODE_DEBUG) {
91          printf("Nouvelle valeur provenant du serveur. Nom:%s Avant:%lf Maintenant:%lf\n"
            , nomValeursServeur[index].c_str(), valeursServeur[index], valeur);
92        }
93
94        valeursServeur[index] = valeur;  // Mise à jour de la valeur stockée dans la RTC
95
96        // Rappel format des trames :
97        // double trame[] = {0,0,0,0,0} // 40 octets
98        // trame[0] = OCTET_DEBUT;
99        // trame[1] = index
100       // trame[2] = ':';
101       // trame[3] à trame[10] = valeur;
102       // trame[11] = OCTET_FIN;
103
104       // Explications memset : https://cplusplus.com/reference/cstring/memset/
105       memset(trame, 0, sizeof(trame));  // Réinitialisation de la trame en mettant tous
          les bits à 0 (par précautions)
106
107       // Explications memcpy : https://en.cppreference.com/w/cpp/string/byte/memcpy
108       memcpy(&trame[0], &OCTET_DEBUT, sizeof(OCTET_DEBUT));
                                                                                //
          OCTET_DEBUT
109       memcpy(&trame[sizeof(OCTET_DEBUT)], &index, sizeof(index));
                                                                    // index
110       memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index)], &separateur, sizeof(separateur
          ));                                          // ':'
111       memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index) + sizeof(separateur)], &valeur,
          sizeof(double));                             // valeur
112       memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index) + sizeof(separateur) + sizeof(
          valeur)], &OCTET_FIN, sizeof(OCTET_FIN));  // OCTET_FIN
113
114       int writeResult = uart_write_bytes(uart_num, trame, sizeof(trame));  //
          Transmission de la trame au STM
115
116       if (MODE_DEBUG) {
117         printf("\nEnvoie de %d octets en UART\n", writeResult);
118       }
119     }
120   }
121
122   //// Fonction IndexValeurServeur: logique pour retrouver depuis le nom d'un paramètre
      son index dans le tableau des valeurs
123   //// Description: parcourir le tableau nomValeursServeur pour trouver l'index où la
      valeur correspond au nom indiqué en paramètre d'entrée
124   //// Entrées: String nom (nom du paramètre recherché)
```

```
//// Sorties: iont8_t (index du paramètre dans le tableau des valeurs stockées, -1 si non trouvé)
int8_t IndexValeurServeur(String nom) {
  int tailleTableau = sizeof(valeursServeur) / sizeof(valeursServeur[0]);  // On recalcule la taille du tableau (pour plus d'informations sur la fonction sizeof : https://www.arduino.cc/reference/en/language/variables/utilities/sizeof/)
  for (int i = 0; i < tailleTableau; i++) {
    if (nomValeursServeur[i] == nom) {
      return i;  // Nom trouvé
    }
  }

  return -1;  // Nom introuvable
}
```