Annexe S.10

```c
//  communication.c
//
//  Description : fonctions liées à la communication ESP/STM
//  Auteur : Perret Mélissa
//  Création : 16/09/2024
//  Modifications : --

//  Version   : V1.0
/*-----------------------------------------------------------*/


#include "communication.h"
#include <string.h> // pour memset et memcpy


uint8_t receptionTramesUART[TAILLE_BUFFER]; // Tableau pour stocker les trames UART reçues

// Important: sizeof ne renvoie pas les bonnes valeurs quand utilisé pour définir la valeurs de variables déclarées en const
uint8_t OCTET_DEBUT = 0x02; // STX (Start of Text)
uint8_t OCTET_FIN = 0x03; // ETX (End of Text)
uint8_t TAILLE_TRAME = sizeof(OCTET_DEBUT) + sizeof(int8_t) + sizeof(char) + sizeof(double) + sizeof(OCTET_FIN);


//// Fonction ReceptionnerTrameUART (réception analyse et traitement des trames UART provenant de l'ESP)
//// Description: réception des données UART
////              analyse des données pour localiser les trames (caractères de début et de fin)
////              analyse des trames reçues pour extraire et stocker les données reçues
//// Entrées: Pointeur:UART_HandleTypeDef huart, Tableau:DefinitionValeur valeursServeur (tableau pour identifier et stocker les valeurs reçues)
//// Sorties: -
void ReceptionnerTrameUART(UART_HandleTypeDef* huart, DefinitionValeur valeursServeur[])
{
  int uartStatut; // Déclaration variable locale (valeur de retour de la lecture UART)

  do
  {
    // Réception données UART
    uartStatut = HAL_UART_Receive(huart, (uint8_t*)receptionTramesUART, TAILLE_TRAME, UART_RECEPTION_TIMEOUT_MS);
    if(uartStatut == HAL_OK)
    {
      // On parcourt le buffer pour trouver toutes les trames (identifiées par OCTET_DEBUT)
      for (int indexDebutTrame = 0; indexDebutTrame < TAILLE_BUFFER; indexDebutTrame++)
      {
        if (receptionTramesUART[indexDebutTrame] == OCTET_DEBUT) // Début de trame trouvé
        {
          int indexFinTrame = indexDebutTrame + TAILLE_TRAME - 1;
          if(indexFinTrame < TAILLE_BUFFER && receptionTramesUART[indexFinTrame] == OCTET_FIN) // Si on a bien une fin de trame comme attendu
          {
            uint8_t index = receptionTramesUART[indexDebutTrame + 1]; // Deuxième octet de la trame correspond à l'identifiant de la
trame
            // Troisème octet correspond à un séparateur ':' pour faciltier la lecture des trames lors du debuggage

            // Exemple de trame
            // 0:12
```

```
53                  // Effet de la trame: mettre la valeur de SEUIL_TEMPERATURE_MIN à 12
54                  // valeursServeur[0].valeur = 12
55
56                  // Explications pour memcpy : https://en.cppreference.com/w/cpp/string/byte/memcpy
57                  // Les 8 octets suivants (octets 4 à 11) correspondent à la valeur de la trame
58                  // On copie la valeur de la trame dans la variable qu'on utilise pour stocker les valeurs du serveur
59                  memcpy(&valeursServeur[index].valeur, &receptionTramesUART[indexDebutTrame + 3], sizeof(double));
60
61                  rafraichirEPaper = true; // Une nouvelle valeur reçue implique qu'un rafraichissement de l'écran sera nécessaire
62              }
63          }
64      }
65  }
66    } while (uartStatut == HAL_OK); // Continuer tant qu'on reçoit quelque chose
67  }
68
69  //// Fonction EnvoyerTrameUART (construire et transmettre une trame UART à l'ESP)
70  //// Description: construire et transmettre la trame UART à l'ESP
71  //// Entrées: Pointeur:UART_HandleTypeDef huart, uint8_t index, double valeur
72  //// Sorties: -
73  void EnvoyerTrameUART(UART_HandleTypeDef* huart, uint8_t index, double valeur)
74  {
75      HAL_GPIO_WritePin(ALARME_GPIO_Port, ALARME_Pin, GPIO_PIN_SET); // Réveiller l'ESP
76      HAL_Delay(DELAI_ENVOIE_TRAME_MS); // Délai pour être sûr que l'ESP soit réveillé et prêt à recevoir les trames
77
78      uint8_t trame[TAILLE_TRAME];  // Buffer pour stocket la trame à envoyer
79      char separateur = ':'; // Séparateur entre l'index et la valeur dans les trames, pour faciliter le debuggage
80
81      // Explications memset : https://cplusplus.com/reference/cstring/memset/
82      memset(trame, 0, sizeof(trame)); // Réinitialisation de la trame en mettant tous les bits à 0 (par précautions)
83
84      // Explications memcpy : https://en.cppreference.com/w/cpp/string/byte/memcpy
85      memcpy(&trame[0], &OCTET_DEBUT, sizeof(OCTET_DEBUT));                                                                  // OCTET_DEBUT
86      memcpy(&trame[sizeof(OCTET_DEBUT)], &index, sizeof(index));                                                            // index
87      memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index)], &separateur, sizeof(separateur));                                  // ':'
88      memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index) + sizeof(separateur)], &valeur, sizeof(double));                     // valeur
89      memcpy(&trame[sizeof(OCTET_DEBUT) + sizeof(index) + sizeof(separateur) + sizeof(valeur)], &OCTET_FIN, sizeof(OCTET_FIN));   // OCTET_FIN
90
91      HAL_UART_Transmit(huart, trame, sizeof(trame), 1000); // Transmission de la trame à l'ESP
92  }
93
```