Annexe T.4

```
/* ********************************************************************** */
/** Descriptive File Name

  @Company
    ETML-ES

  @File Name
    CommunicationSTM.h

  @Auteurs
    - Perret Mélissa

  @Description
    Fonctions liées à la communication entre l'ESP et le STM
 */
/* ********************************************************************** */


#ifndef _COMMUNICATION_STM_H
#define _COMMUNICATION_STM_H


#include "driver/uart.h"  // pour UART_NUM_0
#include <HTTPClient.h>   // pour HTTPClient


#define TXD_PIN (GPIO_NUM_7)  // PIN utilisée pour écrire les trames UART
#define RXD_PIN (GPIO_NUM_6)  // PIN utilisée pour lire les trames UART

#define PIN_POUR_REVEILLER_STM GPIO_NUM_5  // PIN utilisée pour réveiller le STM (état haut) et pour indiquer quand l'ESP à fini de transmettre ses trames UART (état bas)

#define UART_RECEPTION_TIMEOUT_MS 100  // Timeout lors de la réception UART

const uart_port_t uart_num = UART_NUM_0;      // UART utilisé
const size_t TAILLE_BUFFER = 50;              // Taille du tableau utilisé pour stocker les données UART reçues
uint8_t receptionTramesUART[TAILLE_BUFFER];  // Tableau pour stocker les données UART reçues

// Enumération pour représenter le type de valeur reçu par le STM et améliorer la lisibilité du code
typedef enum {
  ETAT_SEUILS,
  ETAT_BATTERIE,
} ValeursSTM;

// Enumération pour représenter les états reçus par le STM (alarme seuils ou batterie) et améliorer la lisibilité du code
typedef enum {
  FONCTIONEL,  // 0
  ALARME,      // 1
  IDENTIQUE,   // 2
} Etat;

// Structure pour stocker les valeurs liées au STM (alarme seuils et etat batterie)
typedef struct {
  String nom;
  double valeur;
} ValeurSTM;

// Valeurs liées au STM (alarme seuils et etat batterie)
ValeurSTM valeursSTM[] = {
  { "alarme", -9999 },
  { "pilesFaibles", -9999 },
};


#endif /* _COMMUNICATION_STM_H */
```