```
/*
 * DHTLib_GPA788.h
 * Mise à jour du code de Rob Tillaart en utilisant le C++ moderne.
 *
 * GPA788 - ETS
 * T. Wong
 * 10-2019
 * 07-2020
 *
 * Le code est la création de Rob Tillaart voir les commentaires
 * ci-dessous.
 */
//
//    FILE: dht.h
//  AUTHOR: Rob Tillaart
// VERSION: 0.1.14
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// HISTORY:
// see dht.cpp file
//

#ifndef dht_h
#define dht_h

#include <Arduino.h>

/* ------------------------------------------------------------------------
 * Code d'erreur de cette bibliothèque
 * Note: Utiliser un enum class pour réduire les conflits potentiels de nom et pour
 *       permettre le "type check" par le compilateur.
 * ------------------------------------------------------------------------
*/
enum class DHTLIB_ErrorCode : int16_t { DHTLIB_OK = 0, DHTLIB_ERROR_CHECKSUM = -1,
               DHTLIB_ERROR_TIMEOUT = -2, DHTLIB_INVALID_VALUE = -999 };

/* ------------------------------------------------------------------------
 * Constantes et variables globales
 * ------------------------------------------------------------------------
*/
const char DHT_LIB_VERSION[] = "0.1.14";
const uint8_t DHTLIB_DHT11_WAKEUP{18};  // 18 ms pour DHT11
const uint8_t DHTLIB_DHT_WAKEUP  {1};   // 1 ms pour les autres

// max timeout is 100 usec.
// For a 16 Mhz proc 100 usec is 1600 clock cycles
// loops using DHTLIB_TIMEOUT use at least 4 clock cycli
// so 100 us takes max 400 loops
// so by dividing F_CPU by 40000 we "fail" as fast as possible
const uint16_t DHTLIB_TIMEOUT = (F_CPU/40000);

class dhtlib_gpa788
{
public:
  /* ------------------------------------------------------------------------
--
   * Constructeurs, assignation et destructeur
   * Note: seul le constructeur par défaut est utilisable.
```

```
57    * ------------------------------------------------------------------------
-- */
58   // Constructeur par défaut
59   dhtlib_gpa788(uint8_t pin) {
60     // Initialiser la température et l'humidité relative à des valeurs" connues"
61     humidity = temperature = static_cast<double>
(DHTLIB_ErrorCode::DHTLIB_INVALID_VALUE);
62     setPin(pin);
63   }
64
65   dhtlib_gpa788() {
66     // initialiser la temp et l'humidité avec des valeurs par défait et passer en
paramètre un numéro de pin
67     humidity = temperature = static_cast<double>
(DHTLIB_ErrorCode::DHTLIB_INVALID_VALUE);
68   }
69   // Empêcher l'utilisation du constructeur de copie
70   dhtlib_gpa788(const dhtlib_gpa788& other) = delete;
71   // Empêcher l'utilisation de l'opérateur d'assignation
72   dhtlib_gpa788& operator=(const dhtlib_gpa788& other) = delete;
73   // Empêcher l'utilisation du constructeur par déplacement
74   dhtlib_gpa788(dhtlib_gpa788&& other) = delete;
75   // Empêcher l'utilisation de l'opérateur de déplacement
76   dhtlib_gpa788& operator=(dhtlib_gpa788&& other) = delete;
77   // Destructeur
78   ~dhtlib_gpa788() { }
79
80   /* ------------------------------------------------------------------------
--
81    * Lire l'humidité et la température par le capteur
82    * Code de retour: DHTLIB_OK, DHTLIB_ERROR_CHECKSUM, DHTLIB_ERROR_TIMEOUT.
83    * ------------------------------------------------------------------------
-- */
84   DHTLIB_ErrorCode read11(uint8_t pin);
85   inline DHTLIB_ErrorCode read21(uint8_t pin) { return read(pin); };
86   inline DHTLIB_ErrorCode read22(uint8_t pin) { return read(pin); };
87   inline DHTLIB_ErrorCode read33(uint8_t pin) { return read(pin); };
88   inline DHTLIB_ErrorCode read44(uint8_t pin) { return read(pin); };
89
90   /* ------------------------------------------------------------------------
--
91    * Accesseurs
92    * ------------------------------------------------------------------------
-- */
93   double getHumidity() const { return humidity; }
94   double getTemperature() const { return temperature; }
95   uint8_t getPin() const {return mPin;}
96
97 /* ------------------------------------------------------------------------
98    * Accesseurs
99    * ------------------------------------------------------------------------
-- */
100  void setPin(uint8_t pin) {mPin = pin;}
101
102 private:
103   uint8_t bits[5];  // buffer to receive data
104   // "Driver" pour les capteurs autre que le DHT11
105   DHTLIB_ErrorCode read(uint8_t pin);
106   // Réalise la séquence de communication avec le capteur
107   DHTLIB_ErrorCode _readSensor(uint8_t pin, uint8_t wakeupDelay);
```

```
  // Données lues du capteur
  double humidity;
  double temperature;
  uint8_t mPin;
};
#endif
//
// END OF FILE
//
```