```
/** Broche "DATA" du DHT11 */
const byte DHT11_PIN = 5;

/** Code d'erreur de la fonction readDHT11() */
enum {
  DHT11_SUCCESS, //!< Pas d'erreur
  DHT11_TIMEOUT_ERROR, //!< Temps d'attente dépassé
  DHT11_CHECKSUM_ERROR //!< Données reçues erronées
};

/** Fonction setup() */
void setup() {

  /* Initialisation du port série */
  Serial.begin(115200);
  Serial.println(F("DHT11 DEMO"));
}

/** Fonction loop() */
void loop() {

  /* Variables d'usage */
  byte temperature, humidity;

  /* Lecture de la température et de l'humidité + gestion des erreurs */
  switch (readDHT11(DHT11_PIN, &temperature, &humidity)) {
  case DHT11_SUCCESS:

    /* Affichage de la température et du taux d'humidité */
    Serial.print(F("Humidite (%): "));
    Serial.println((int) humidity);
    Serial.print(F("Temperature (^C): "));
    Serial.println((int) temperature);
    break;

  case DHT11_TIMEOUT_ERROR:
    Serial.println(F("Temps d'attente depasse !"));
    break;

  case DHT11_CHECKSUM_ERROR:
    Serial.println(F("Erreur de checksum !"));
    break;
  }

  /* Pas besoin de rafraichir l'affichage très souvent */
  delay(2000);
}

/**
 * Lit la température et le taux d'humidité capté par un capteur DHT11
 *
 * @param pin Broche sur laquelle est câblé le capteur
 * @param temperature Pointeur vers la variable stockant la température
 * @param humidity Pointeur vers la variable stockant le taux d'humidité
 * @return DHT11_SUCCESS si aucune erreur, DHT11_TIMEOUT_ERROR en cas de timeout, ou
DHT11_CHECKSUM_ERROR en cas d'erreur de checksum
 */
byte readDHT11(byte pin, byte* temperature, byte* humidity) {

  /* data[] -> buffer contenant les données du cpateur
   * counter -> compteur permettant de savoir quel bit est reçu (bitwise)
   * index -> compteur permettant de savoir quel octet est reçu (bitwise)
   * timeout -> compteur pour le timeout
   */
  byte data[5] = { 0 }, counter = 7, index = 0;
  unsigned int timeout;

  /* Conversion du numéro de broche Arduino en ports/masque binaire "bas niveau" */
  /* Utiliser les registres du microcontrôleur est bien plus rapide que digitalWrite() */
  uint8_t bit = digitalPinToBitMask(pin);
  uint8_t port = digitalPinToPort(pin);
  volatile uint8_t *ddr = portModeRegister(port); // Registre MODE (INPUT / OUTPUT)
```

```
  volatile uint8_t *out = portOutputRegister(port); // Registre OUT (écriture)
  volatile uint8_t *in = portInputRegister(port); // Registre IN (lecture)

  /* Réveil du capteur */
  *ddr |= bit; // OUTPUT
  *out &= ~bit; // LOW
  delay(18); // Temps d'attente à LOW causant le réveil du capteur
  *out |= bit; // HIGH
  delayMicroseconds(40);
  *ddr &= ~bit; // INPUT

  /* Attente de la réponse du capteur */
  timeout = 0;
  while(!(*in & bit)) /* Attente d'un état LOW */
    if (++timeout == 10000)
      return DHT11_TIMEOUT_ERROR;
  timeout = 0;
  while(*in & bit) /* Attente d'un état HIGH */
    if (++timeout == 10000)
      return DHT11_TIMEOUT_ERROR;

  /* Lecture des données du capteur (40 bits) */
  for (byte i = 0; i < 40; ++i) {

    /* Attente d'un état LOW */
    timeout = 0;
    while(!(*in & bit))
      if (++timeout == 10000)
        return DHT11_TIMEOUT_ERROR;

    /* Mise en mémoire du temps courant */
    unsigned long t = micros();

    /* Attente d'un état HIGH */
    timeout = 0;
    while(*in & bit)
      if (++timeout == 10000)
        return DHT11_TIMEOUT_ERROR;

    /* Si le delta Temps est supérieur à 40µS c'est un "1", sinon c'est un "0" */
    if ((micros() - t) > 40)
      data[index] |= (1 << counter); // "1"
    // Le tableau data[] est initialisé à "0" par défaut <span class="wp-smiley wp-emoji wp-emoji-
wink" title=";)">;)</span>

    /* Si le compteur de bits atteint zéro */
    if (counter-- == 0) {
      counter = 7; /* On passe à l'octet suivant */
      ++index;
    }
  }

  /* Format des données :
   * [0] = humidité en %
   * [1] = zéro
   * [2] = température en degrés Celsius
   * [3] = zéro
   * [4] = checksum (humidité + température)
   */
  *humidity = data[0];
  *temperature = data[2];

  /* Vérifie la checksum */
  if (data[4] != (data[0] + data[2]))
    return DHT11_CHECKSUM_ERROR; /* Erreur de checksum */
  else
    return DHT11_SUCCESS; /* Pas d'erreur */
}
```