```
bool sleep = true;
/**
 * e-p-433-v2
 * 1 - Introduction
 *Dans le cadre du concept culinaire Quiet cook http://quiet-cook.com/
 *et de son Système de Cuisson Assistée par Ordinateur (SCAO) http://fablabo.net/
wiki/SCAO.
 *Le prototypage (prototype N°3) de la e-poignée (433MHZ en version 2)
 *est réalisé par Régis LERUSTE http://fablabo.net/wiki/Utilisateur:LERUSTE_REGIS
 *et Olivier MARAIS http://fablabo.net/wiki/
Cahier_de_recettes#Les_recettes_d.27Olivier
 *Dans l'environnement de développement Arduino IDE,
 *le programe e-p-433-v2.ino constitue le code source qui permet l'édition,
 *la compilation et le transfert du firmware à destination du micro-contrôleur.
 *L'objet de ce programme est double :
 *- la capture des températures délivrées par 2 thermomètres digitaux.
 *- la transmission périodique de ces valeurs au e-rupteur (e-r-433).
 *Ce programme est dévelloppé sous licence creative commons CC-BY-SA.
 *Il utile des ressources extérieures (librairies et codes sources) développées
par des informaticiens.
 *Chacun, des sous paragraphes ci-dessous, dédié à une fonction, cite, le nom de
l'informaticien, indique les liens permettant d'accéder à la librairie et aux
codes sources.
 *
 * 1a - Acquisitions des températures
 * - Paul Stoffregem, accesibles par les liens :
 *- https://www.pjrc.com/teensy/td_libs_OneWire.html
 *- https://github.com/PaulStoffregen/OneWire
 *Il utile le code source dévelloppé par :
 *- Fabien Batteix (Skywood) qui est le rédacteur en chef et administrateur du
site Carnet du Maker, ainsi que le gérant de l'entreprise TamiaLab qui édite le
site.
 *- selon l'article https://www.carnetdumaker.net/articles/mesurer-une-temperature-
avec-un-capteur-1-wire-ds18b20-et-une-carte-arduino-genuino/
 *- en tenant compte que pour des raisons de sécurité inhérant au projet SCAO
qu'il est indispensable que la capture de données en provenance
 * des 2 thermomètres digitaux DB18B20 se réalise toujours dans le même ordre, en
particulier après un changement de l'un ou des deux thermomètres.
 *
 * 1b - Mesure des tensions
 *
 * 1c - BITE
 *
 * 1d - Transmission
 * - Mike McCauley, accesibles par les liens :
 *- https://www.pjrc.com/teensy/td_libs_VirtualWire.html
 *- http://www.airspayce.com/mikem/arduino/VirtualWire.pdf
 *
 * 1e - Horodatage - Chronomètre
 * Horodatage
 * - Michael Margolis, accesibles par les liens :
 *- https://www.pjrc.com/teensy/td_libs_Time.html
 *- https://github.com/PaulStoffregen/Time
 *
 * Chrono library for Arduino or Wiring by Sofian Audry and Thomas Ouellet
Fredericks
 * - http://github.com/SofaPirate/Chrono
 *
 * 1f - Visualisation des résultats
 *
 * 1g - Mode Sleep
 * - Colin Duffy, pour le "mode sleep" (snooze) avec la gestion de la liaison
série, accesible par le lien :
 *- https://github.com/duff2013/Snooze/blob/master/examples/deepsleep/
deepSleep_usb_serial/deepSleep_usb_serial.ino
 *
```

```
 * 1h - Bilan énergétique de la batterie
 * Calculs de l'énegie électrique consommée et du ratio par rapport à la capacité
nominale de 400 mAh
 */
/**
 * 2 - Initialisation des paramètres
 * Paramètres communs
 */
 unsigned long tt1=0;//temps de travail 1
 unsigned long tt2=0;//temps de travail 2
 unsigned long ti=10000000;//temps itératif
 unsigned long ts=0;//temps de sleep
/**
 * 2a - Acquisition des températures
 * Code pour lire un thermomètre digital DS18B20 sur un bus 1-Wire.
 */

/* Dépendance pour le bus 1-Wire */
#include <OneWire.h> //Chargement de la librairie OneWire.h
/* Broches des 2 bus 1-Wire */
const byte BROCHE_ONEWIRE_1 = 16;
const byte BROCHE_ONEWIRE_2 = 17;

/* Code de retour des 2 fonctions getT1() et getT2() */
enum DS18B20_RCODES {
  READ_OK,  // Lecture ok
  NO_SENSOR_FOUND,  // Pas de capteur
  INVALID_ADDRESS,  // Adresse reçue invalide
  INVALID_SENSOR  // Capteur invalide (pas un DS18B20)
};


/* Création des 2 objets OneWire pour manipuler les 2 bus 1-Wire */
OneWire ds1(BROCHE_ONEWIRE_1);
OneWire ds2(BROCHE_ONEWIRE_2);
/**
 * 2b - Mesures des tensions
 */
const int Vbat_demie_1 = A6; //Mesure de la moitié de la tension Vbat avant la
résistance de 1 Ohm
const int Vbat_demie_2 = A7; //Mesure de la moitié de la tension Vbat_2 après la
résistance de 1 Ohm
float Vbat_1;
float Vbat_2;
float ibat;
const float Vbat_limite = 4300;
const float Vbat_nominal = 3700;
const float Vbat_min = 3600;
const float Vbat_cut_off = 2800;
const int Vusb_demie = A8; //Mesure de la moitié de la tension Vusb
float Vusb;
const int V33_demie = A9; //Mesure de la moitié de la tension 3.3V
float V33;
const int MaxConv = 8192;
const int MaxVolt = 3272;
/**
 * 2c - Built In Test Equipment (BITE)
/**
 * 2d - Transmission
 * Chargement de la librairie VirtualWire - Gestion de l'émetteur 433 MHZ
 */
#include <VirtualWire.h>
const int transmit_pin = 18;//Pin de sortie de l'émetteur
byte count = 1;//Initialisation du numéro du message
/**
 * 2e - Horodatage - Chronomètre
```

```
 * Horodatage
 */
#include "TimeLib.h" //Include IimeLib.h library
/**
 * Chronomètre
  */
#include <Chrono.h> //Include Chrono.h Library
Chrono Chrono(Chrono::MICROS);//Instanciate a Chrono object

/**
 * 2f - Visualisation des résultats
 *  Les résultats du BITE sont visalisés par 3 leds :
verte : l'allumage témoigne d'un bon fonctionnement
        l'extinction peut être envisagée en mode sleep
jaune : allumée quand l'émission commence
        éteinte quand l'émission s'arrète
rouge : l'allumage témoigne d'une ou plusieurs anomalies
 */
const int led_pin_v = 13;//Led verte
const int led_pin_j = 14;//Led jaune
const int led_pin_r = 15;//Led rouge
/**
 * 2g - Mode sleep
 * Chargement de la librairie
 */
#include <Snooze.h>
// Load timer and USBSerial drivers
SnoozeTimer timer;
SnoozeUSBSerial usb;

/*********************************************************
 Install drivers to a SnoozeBlock, timer to wake and USB Serial Driver to
 fix printing to serial monitor after sleeping.
 *********************************************************/
SnoozeBlock config_teensy32(usb, timer);
/**
 * 2h - Bilan énergétique de la batterie
 * Chargement de la librairie
 */
unsigned long Etot=0; //Energie électrique totale consommée en joule
unsigned long Et=0; //Energie électrique consommée pendant le travail
unsigned long Es=0; //Energie électrique consommée pendant le sommeil (sleep)
unsigned long Ec=0; //Energie électrique consommée cumulée
/**
 * 3 - Fonctions spécifiques
 * 3a-1 - Fonction d'acquisition de la température via le 1er thermomètre digital
DS18B20 (ds1).
/**
 * Fonction de lecture de la température via un thermomètre digital DS18B20 câblé
sur le 1er bus ds1.
 */
byte getT1(float *T1, byte reset_search) {
  byte data[9], addr[8];
  // data[] : Données lues depuis le scratchpad
  // addr[] : Adresse du module 1-Wire détecté

  /* Reset le bus 1-Wire si nécessaire (requis pour la lecture du premier capteur)
*/
  if (reset_search) {
    ds1.reset_search();
  }

  /* Recherche le prochain capteur 1-Wire disponible */
  if (!ds1.search(addr)) {
    // Pas de capteur
    return NO_SENSOR_FOUND;
```

```
  }

  /* Vérifie que l'adresse a été correctement reçue */
  if (OneWire::crc8(addr, 7) != addr[7]) {
    // Adresse invalide
    return INVALID_ADDRESS;
  }

  /* Vérifie qu'il s'agit bien d'un DS18B20 */
  if (addr[0] != 0x28) {
    // Mauvais type de capteur
    return INVALID_SENSOR;
  }

  /* Reset le bus 1-Wire et sélectionne le capteur */
  ds1.reset();
  ds1.select(addr);

  /* Lance une prise de mesure de température et attend la fin de la mesure */
  ds1.write(0x44, 1);
  delay(800);

  /* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de lecture
du scratchpad */
  ds1.reset();
  ds1.select(addr);
  ds1.write(0xBE);

 /* Lecture du scratchpad */
  for (byte i = 0; i < 9; i++) {
    data[i] = ds1.read();
  }

  /* Calcul de la température en degré Celsius */
  *T1 = (int16_t) ((data[1] << 8) | data[0]) * 0.0625;

  // Pas d'erreur
  return READ_OK;
}
 /** 3a-2 - Fonction d'acquisition de la température via le 2ème thermomètre
digital DS18B20 (ds2).
 */
byte getT2(float *T2, byte reset_search) {
  byte data[9], addr[8];
  // data[] : Données lues depuis le scratchpad
  // addr[] : Adresse du module 1-Wire détecté

  /* Reset le bus 1-Wire ci nécessaire (requis pour la lecture du premier capteur)
*/
  if (reset_search) {
    ds2.reset_search();
  }

  /* Recherche le prochain capteur 1-Wire disponible */
  if (!ds2.search(addr)) {
    // Pas de capteur
    return NO_SENSOR_FOUND;
  }

  /* Vérifie que l'adresse a été correctement reçue */
  if (OneWire::crc8(addr, 7) != addr[7]) {
    // Adresse invalide
    return INVALID_ADDRESS;
  }

  /* Vérifie qu'il s'agit bien d'un DS18B20 */
```

```
  if (addr[0] != 0x28) {
    // Mauvais type de capteur
    return INVALID_SENSOR;
  }

  /* Reset le bus 1-Wire et sélectionne le capteur */
  ds2.reset();
  ds2.select(addr);

  /* Lance une prise de mesure de température et attend la fin de la mesure */
  ds2.write(0x44, 1);
  delay(800);

  /* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de lecture
du scratchpad */
  ds2.reset();
  ds2.select(addr);
  ds2.write(0xBE);

 /* Lecture du scratchpad */
  for (byte i = 0; i < 9; i++) {
    data[i] = ds2.read();
  }

  /* Calcul de la température en degré Celsius */
  *T2 = (int16_t) ((data[1] << 8) | data[0]) * 0.0625;

  // Pas d'erreur
  return READ_OK;
}
/**
 * 3b - Fonction mesure des tensions et calibration
 * 3c - BITE
 * 3d - Tranmission
 * 3e - Horodatage - Chronomètre
 * 3f - Visualisation des résultats
 * 3g - Mode sleep
 * 3h - Bilan énergétique de la batterie
 */

/** 4 - Fonction setup() **/
void setup() {
  /* Initialisation du port série */
  Serial.begin(9600);
  delay(1000);
  Serial.println("N°;date;heure;T1 (degrés C);T2 (degrés C);Vusb (mV);Vbat
(mV);ibat (mA);V33 (mV);Ec (joules)");
  //a) Acquisition des températures
  //b) Mesure des tensions
  analogReadResolution(13);
  //c) BITE

  //d) Transmission
    // Initialise the IO and ISR
  vw_set_tx_pin(transmit_pin);
  vw_setup(2000);   // Bits per sec
  //e) Horodatage - Chronomètre
  setTime(12, 05, 00, 29, 07, 2018);

  //f) Visualisation des résultats
  pinMode(led_pin_v, OUTPUT);
  pinMode(led_pin_j, OUTPUT);
  pinMode(led_pin_r, OUTPUT);

 //g) Mode sleep
  digitalWrite(led_pin_r, HIGH);//USB serial connection is not establihed - Clic
```

```
on serial monitor icon (right top icon)
  //while (!Serial);
    delay(100);
    digitalWrite(led_pin_r, LOW);
    //Serial.println("Starting...");
    delay(100);
    }
 //h) - Bilan énergétique de la batterie
 /** 5 - Fonction loop() **/
void loop() {
  Chrono.restart();  // restart the Chrono
  //a) Acquisition des températures
  float T1;
  /* Lit la température T1 */
  if (getT1(&T1, true) != READ_OK) {
    Serial.println(F("Erreur de lecture du capteur"));
    return;
  }
  float T2;
  /* Lit la température ambiante à ~1Hz */
  if (getT2(&T2, true) != READ_OK) {
    Serial.println(F("Erreur de lecture du capteur"));
    return;
  }

  //b) Mesure des tensions
  Vusb=map (2.0038*analogRead(Vusb_demie), 0, MaxConv, 0, MaxVolt);
  Vbat_1=map (2.0038*analogRead(Vbat_demie_1), 0, MaxConv, 0, MaxVolt);
  Vbat_2=map (2.0038*analogRead(Vbat_demie_2), 0, MaxConv, 0, MaxVolt);
  ibat=Vbat_1 - Vbat_2;
  V33=map (2.0038*analogRead(V33_demie), 0, MaxConv, 0, MaxVolt);

  //c) BITE
  int tmax=25;
  if (T2 >= tmax){digitalWrite(led_pin_r, HIGH);} else {digitalWrite(led_pin_r,
LOW);}
  //if (Vbat_2 <= Vbat_limite && Vbat_2 > Vbat_min){digitalWrite(led_pin_v,
HIGH);} else {digitalWrite(led_pin_v, LOW);}
  if (Vbat_2 > Vbat_limite || Vbat_2 <= Vbat_cut_off){digitalWrite(led_pin_r,
HIGH);} else {digitalWrite(led_pin_r, LOW);}

  //d) Transmission
  /* Transmission des données à l'e-r-433 */
  char msg[7] = {'h','e','l','l','o',' ','#'};

  msg[6] = count;
  digitalWrite(led_pin_j, HIGH); // Flash a light to show transmitting
  vw_send((uint8_t *)msg, 7);
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(led_pin_j, LOW);
  count = count + 1;
  delay(100);

  //e) Horodatage - Chronomètre
  time_t t = now();

  //g) Mode sleep

      /*****************************************************
     Set Low Power Timer wake up in milliseconds.
     *****************************************************/
  tt1 = (Chrono.elapsed());
  ts = ti - (tt1 + tt2);
  timer.setTimer(ts/1000);// milliseconds
  delay(200);
  if (sleep) {Snooze.deepSleep( config_teensy32 );}// return module that woke
```

```
processor
  if (!sleep) {delay(ts/1000);}
  Chrono.restart();  // restart the Chrono
  // wait for serial monitor
    elapsedMillis time = 0;
    while (!Serial && time < 1000) {
        Serial.write(0x00);// print out a bunch of NULLS to serial monitor
        digitalWriteFast(led_pin_v, HIGH);
        delay(30);
        digitalWriteFast(led_pin_v, LOW);
        delay(30);
    }
    // normal delay for Arduino Serial Monitor
    delay(200);
    delay(1000);

  //f) Visualisation des résultats
  Serial.print(count);
  Serial.print(";");
  Serial.print(day (t));
  Serial.print("/");
  Serial.print(month (t));
  Serial.print("/");
  Serial.print(year (t));
  Serial.print(";");
  Serial.print(hour (t));
  Serial.print(":");
  Serial.print(minute (t));
  Serial.print(":");
  Serial.print(second (t));
  Serial.print(";");
  /* Affiche T1 et T2 */
  //Serial.print(F("T1 : "));
  Serial.print(T1, 2);
  Serial.write(";"); // Caractère degré
  //Serial.write('C');
  //Serial.print(";");
  //Serial.print(F("T2 : "));
  Serial.print(T2, 2);
  Serial.write(";"); // Caractère degré
  //Serial.write('C');
  //Serial.print(";");
  /* Affiche les tensions et le courant consommé ibat*/
  //Serial.print(" - Vusb : ");
  Serial.print(Vusb);
  Serial.print(";");
  //Serial.print(" Vbat : ");
  Serial.print(Vbat_1);
  Serial.print(";");
  //Serial.print(" ibat : ");
  Serial.print(ibat);
  Serial.print(";");
  //Serial.print(" V33 : ");
  Serial.print(V33);
  Serial.print(";");
  //Serial.println(";");
  //Serial.print(" ti : ");
  //Serial.print(ti);
  //Serial.print(" tt1 : ");
  //Serial.print(tt1);
  tt2 = (Chrono.elapsed());
  //Serial.print(" tt2 : ");
  //Serial.print(tt2);
  //Serial.print(" ts : ");
  //Serial.println(ts);
```

```
//h) - Bilan énergétique de la batterie
Et = ((Vbat_1/1000) * ibat)*((tt1 + tt2)/1000000);
Es = ((Vbat_1/1000) * 1.5)*(ts/1000000);
Etot = Et + Es;
Ec = Ec + Etot;
//Serial.print(" Energie électrique Ec (joules) : ");
Serial.println(Ec);
  }
```