```
bool sleep = false;
/**
 * 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
*ce programe e-p-433-v2.ino constitue le code source qui permet l'édition,
la compilation et le téléversement du firmware à destination du micro-contrôleur.
*L'environnement de développement Arduino IDE est constitué du microcontrôleur
Teensy 3.2 relié à l'ordinateur à l'aide d'un câble USB. Ce câble permet
l'établissement d'une liaison série. De l'ordinateur vers le microcontrôleur pour
téléverser le firmware. Du microcontrôleur vers l'ordinateur pour l'envoi de
messages à l'aide du port /dev/ttyACM0, soit pour les afficher sur la console,
soit pour les mettre à disposition du programme BASH capture.sh qui édite un
fichier "journal".
*L'objet du firmware est l'administration du microcontrôleur et de ses composants
périphériques câblés selon le schéma électrique https://raw.githubusercontent.com/AIREL46/SCAO/master/kicad/e-p-433-v2/e-p-433-v2-1.png
*Ce firmware est dévelloppé sous licence creative commons CC-BY-SA.
*Son exécution par le microcontôleur est systématique dès son téléversement et
ensuite à chaque mise sous tension. Il est organisé selon 2 fonctions
principales :
*- la capture des températures délivrées par 2 thermomètres digitaux, le premier
concerne la température sur le couvercle de la casserole, le second la température
de la batterie, l'un des éléments qui permet d'assurer la sécurité de la e-
poignée. .
*- la transmission périodique de ces valeurs au e-rupteur (e-r-433).
*Ses fonctions principales sont complétées de fonctions secondaires (voir ci-
dessous).
*Il utile des ressources extérieures (librairies, codes sources et exemples)
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 ainsi
qu'aux codes sources ou aux exemples.
 *
 * 1a - Acquisition des températures
 * L'acquisition de la température est réalisée à l'aide d'un DS18B20  digital
thermometer  provides  9-bit  to  12-bit  Celsius  temperature  measurements.
 * The  DS18B20  communicates  over  a  1-Wire  bus.
 * La librarie OneWire.h met à disposition un ensemble de fonction permettant de
gérer l'échange de données entre le DS18B20 et le microcontrôleur.
 * Le développement de cette librarie est assuré par Paul Stoffregem, accessibles
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 et calcul du courant ibat
 * La mesure des tensions est l'un des éléments qui permet d'assurer la sécurité
de la e-poignée.
 * La mesure consiste en une conversion analogique digitale de la tension. Cette
```

```
conversion est réalisée par le microcontrôleur. La tension à mesurer est appliquée
sur l'une de ses entrées analogiques. Pour tenir compte de la tension
d'alimentation de 3.3V du microcontrôleur, la tension à mesurer est au préalable
divisée par 2 à l'aide d'un pont diviseur constitué de 2 résistances de valeurs
égales.
 * La valeur mesurée est ensuite multipliée par 2, plus exactement par une valeur
légérement supérieure à 2 pour compenser l'effet de l'impédance de l'entrée du
microcontrôleur.
 * 1c - BITE
 * L'objet du Build In Test Equipment (BITE) est d'assurer la sécurité de la e-
poignée.
 * Cette sécurité se concrétise par un réseau de surveillance qui détecte
d'éventuelles anomalies.
 * L'utilisateur est informé de l'état du fonctionnement par 3 leds (verte,
orange, rouge).
 * Le BITE surveille la température de la batterie, les tensions continues Vusb,
Vbat et V33.
 *
 * 1d - Transmission
 * L'objet est la transmission périodique (30s) au e-rupteur (e-r-433) des
échantillons.
 * Les échantillons ont pour objet le regroupement des données.
 * Chaque échantillon contient : la date et l'heure, la température mesurée sur le
couvercle de la casserole ainsi que les données culinaires et de sécurité.
 * Cette fonction utilise un shield AM Transmitter module QAM-TX1 en version 433
MHZ.
 * Ce module est connecté à une antenne 1/4 d'onde.
 * La librarie VirtualWire apporte les fonctions spécifiques qui permettent de
gérer cette fonction transmission.
 * Cette Librarie a été développée par Mike McCauley, elle est accesible par les
liens :
 *- https://www.pjrc.com/teensy/td_libs_VirtualWire.html
 *- http://www.airspayce.com/mikem/arduino/VirtualWire.pdf
 *- https://github.com/manashmndl/VirtualWire
 *
 * 1e - Horodatage & Chronomètre
 * Horodatage
 * L'objet de l'horodatage est la datation des échantillons.
 * Un cristal a été ajouté au Teensy 3.2.
 * La librarie Time apporte les fonctions spécifiques qui permettent de gérer
cette fonction d'horodatage.
 * Cette Librarie a été développée par Michael Margolis, accesibles par les liens :
 *- https://www.pjrc.com/teensy/td_libs_Time.html
 *- https://github.com/PaulStoffregen/Time
 *- https://github.com/PaulStoffregen/Time/blob/master/examples/TimeTeensy3/
TimeTeensy3.ino (pour la fonction RTC),
 *  accessible également dans l'Arduino IDE (Fichier - Exemples - Time -
TimeTeensy3)
 *  TimeRTC.pde
 *  example code illustrating Time library with Real Time Clock.
 *
 *  Chronomètre
 *  L'objet du chronomètre est la mesure du temps de travail du microcontrôleur.
 *  La librarie Chrono apporte les fonctions spécifiques qui permettent de gérer
cette fonction.
 *  Cette Librarie a été développée par Sofian Audry and Thomas Ouellet
Fredericks, elle est accesible par les liens :
 * - http://github.com/SofaPirate/Chrono
 *
 * 1f - Visualisation du contenu des échantillons
 * L'objet est la visualisation du contenu des échantillons.
 * Cette fonction utilise la liaison série (câble USB) entre le microcontrôleur et
l'ordinateur.
 * Elle se concrétise par une console sur l'écran de l'ordinateur.
 * 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
 *- https://github.com/duff2013/Snooze
 *
 * 1h - Bilan énergétique de la batterie
 * L'objet est l'établissement du bilan énergétique de la batterie
 * Cette fonction réalise les calculs de l'énegie électrique consommée et le ratio
par rapport à la capacité nominale de 400 mAh
 *
 * 1i - Start Stop (StSp) Interrupt Service Routine (ISR)
 * L'objet est l'exécution d'une routine d'interruption suite à l'appui sur un
bouton poussoir
 * Cette fonction peut être utilisée pour gérer le mode de fonctionnement Start
Stop ou tout autrevariante de mode de fonctionnement.
 * Une attention toute particulière doit être apportée au traitement de
l'antirebond du bouton poussoir.
 */
/**
 * 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=30000000;//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 - Mesure des tensions et calcul du courant ibat
 */
const int Vbat_demie_1 = A6; //Initialisation de la variable Vbat_demie_1 et
affectation à l'entrée analogique A6 (demie valeur de la tension Vbat avant la
résistance de 1 Ohm)
const int Vbat_demie_2 = A7; //Initialisation de la variable Vbat_demie_2 et
affectation à l'entrée analogique A7 (demie valeur de la tension Vbat après la
résistance de 1 Ohm)
float Vbat_1; //Initialisation de la variable Vbat_1 (valeur de la tension Vbat
avant la résistance de 1 Ohm)
float Vbat_2; //Initialisation de la variable Vbat_2 (valeur de la tension Vbat
après la résistance de 1 Ohm)
float ibat; //Initialisation de la variable ibat (valeur du courant qui traverse
la résistance de 1 Ohm)
const float Vbat_limite = 4300; //Limite supérieure de Vbat
const float Vbat_nominal = 3700; //Valeur nominale de Vbat
```

```
const float Vbat_min = 3600; //Valeur minimun de Vbat
const float Vbat_cut_off = 2800; //Cut off de Vbat
const int Vusb_demie = A8; //Initialisation de la variable Vusb_demie
float Vusb; //Initialisation de la variable Vusb
const int V33_demie = A9; //Initialisation de la variable V33_demie
float V33; //Initialisation de la variable V33
//Initialisation des valeurs utilisées par la fonction de changement d'échelle
(fonction mathématique map() de l'arduino)
const int MaxConv = 8192; //Valeur maximale lue
const int MaxVolt = 3272; //Valeur de la tension correspondante à la valeur
maximale
/**
 * 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 = 0;//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 du contenu des échantillons
 *  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'arrête
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
/** 2h - Bilan énergétique de la batterie
*2i - Start Stop (StSp) Interrupt Service Routine (ISR)
```

```
 *
 */
const byte interruptPin = 11;
volatile byte stsp = LOW;
/**
 * 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 - Mesure des tensions et calcul du courant ibat
 * 3c - BITE
 * 3d - Tranmission
 * 3e - Horodatage & Chronomètre
 */
 void digitalClockDisplay() {
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
```

```
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

time_t getTeensy3Time()
{
  return Teensy3Clock.get();
}

/*  code to process time sync messages from the serial port   */
#define TIME_HEADER  "T"   // Header tag for serial time sync message

unsigned long processSyncMessage() {
  unsigned long pctime = 0L;
  const unsigned long DEFAULT_TIME = 1357041600; // Jan 1 2013

  if(Serial.find(TIME_HEADER)) {
     pctime = Serial.parseInt();
     return pctime;
     if( pctime < DEFAULT_TIME) { // check the value is a valid time (greater than
Jan 1 2013)
       pctime = 0L; // return 0 to indicate that the time is not valid
     }
  }
  Serial.println(pctime);
  return pctime;
}

void printDigits(int digits){
  // utility function for digital clock display: prints preceding colon and
leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

 /** 3f - Visualisation du contenu des échantillons
 * 3g - Mode sleep
 * 3h - Bilan énergétique de la batterie
 * 3i - Start Stop (StSp) Interrupt Service Routine (ISR)
 */
 void StSp(){
  stsp = !stsp;
  }

/** 4 - Fonction setup() **/
void setup() {
  /* Initialisation du port série */
  Serial.begin(9600);
  delay(10000);
  Serial.println("N°;date;heure;T1 (degrés C);T2 (degrés C);Vusb (mV);Vbat
(mV);ibat (mA);V33 (mV);Ec (joules)");
  //4a - Acquisition des températures
  //4b - Mesure des tensions et calcul du courant ibat
  analogReadResolution(13);
  //4c - BITE

  //4d - Transmission
    // Initialise the IO and ISR
```

```
  vw_set_tx_pin(transmit_pin);
  vw_setup(2000);   // Bits per sec
  //4e - Horodatage & Chronomètre
  //setTime(16, 8, 00, 23, 02, 2019);
  // set the Time library to use Teensy 3.0's RTC to keep time
  setSyncProvider(getTeensy3Time); //Configure Time to automatically called the
getTimeFunction() regularly. This function should obtain the time from another
service and return a time_t number, or zero if the time is not known.
  while (!Serial);  // Wait for Arduino Serial Monitor to open
  delay(100);
  if (timeStatus()!= timeSet) {
    Serial.println("Unable to sync with the RTC");
  } else {
    Serial.println("RTC has set the system time");
  }
  //4f - Visualisation du contenu des échantillons
  pinMode(led_pin_v, OUTPUT);
  pinMode(led_pin_j, OUTPUT);
  pinMode(led_pin_r, OUTPUT);

 //4g - Mode sleep
  digitalWrite(led_pin_r, HIGH);//USB serial connection is not eablihed - Clic
on serial monitor icon (right top icon)
  while (!Serial);
    delay(100);
    digitalWrite(led_pin_r, LOW);
    //Serial.println("Starting...");
    delay(100);
 //4h - Bilan énergétique de la batterie
 //4i - Start Stop (StSp) Interrupt Service Routine (ISR)
 pinMode(interruptPin, INPUT_PULLDOWN);
 attachInterrupt(interruptPin, StSp, RISING);
    }
 /** 5 - Fonction loop() **/
void loop() {
  Chrono.restart();  // restart the Chrono
  //5a - 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;
  }

  //5b - Mesure des tensions et calcul du courant ibat
  //fonction de changement d'échelle (fonction mathématique map() de l'arduino)
  //La multiplication par 2.0038 compense la division par 2 (pont diviseur) et
l'impédance de l'entrée du microcontrôleur pour les chiffres après la virgule.
  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);

  //5c - 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);}

  //5d - Transmission
  /* Transmission des données à l'e-r-433 */
  /*La fonction de transmission vw_send(message, length) : transmit a message,
"message" is an array of the bytes to send,
  and "length" is the number of bytes stored in the array.*/
  char msg[7] = {'h','e','l','l','o',' ','#'};
  msg[6] = count;
  time_t heure[1] = {now()};
  float mesure[5]= {T1, T2, Vusb, Vbat_1, ibat};
  unsigned long Energie[1] = {Ec};
  digitalWrite(led_pin_j, HIGH); // Flash a light to show transmitting
  vw_send((uint8_t *)msg, 7); //Transmission hello et count
  vw_wait_tx(); // Wait until the whole message is gone
  vw_send((uint8_t *)heure, 1); //Transmission de l'heure
  vw_wait_tx(); // Wait until the whole message is gone
  vw_send((uint8_t *)mesure, 5); //Transmission T1, T2, Vusb, Vbat, ibat
  vw_wait_tx(); // Wait until the whole message is gone
  vw_send((uint8_t *)Energie, 1); //Transmission de l'énergie consommée
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(led_pin_j, LOW);
  count = count + 1;
  delay(100);

  //5e - Horodatage & Chronomètre
  //time_t t = now();
    if (Serial.available()) {
    time_t t = processSyncMessage();
    if (t != 0) {
      Teensy3Clock.set(t); // set the RTC
      setTime(t);
    }
  }
  digitalClockDisplay();
  delay(1000);
  //5g - 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);

  //5f - Visualisation du contenu des échantillons
  //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);

//5h - Bilan énergétique de la batterie
Et = ((Vbat_1/1000) * ibat)*((tt1 + tt2)/1000000);
if (sleep) {Es = ((Vbat_1/1000) * 1.5)*(ts/1000000);}
if (!sleep) {Es = ((Vbat_1/1000) * ibat)*(ts/1000000);}
Etot = Et + Es;
Ec = Ec + Etot;
//Serial.print(" Energie électrique Ec (joules) : ");
Serial.println(Ec);
//5i - Start Stop (StSp) Interrupt Service Routine (ISR)
digitalWrite(led_pin_v, stsp);
  }
```