```
1  bool sleep = false;
2  /**
3   * e-p-433-v2
4   * 1 - Introduction
5   *Dans le cadre du concept culinaire Quiet cook http://quiet-cook.com
6   *et de son Système de Cuisson Assistée par Ordinateur (SCAO) http://
    fablabo.net/wiki/SCAO.
7   *Le prototypage (prototype N°3) de la e-poignée (433MHZ en version 2)
8   *est réalisé par Régis LERUSTE http://fablabo.net/wiki/
    Utilisateur:LERUSTE_REGIS
9   *et Olivier MARAIS http://fablabo.net/wiki/Cahier_de_recettes#Les_recettes_d.
    27Olivier
10 *ce programe e-p-433-v2.ino constitue le code source qui permet l'édition,
11 la compilation et le téléversement du firmware à destination du micro-
    contrôleur.
12 *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".
13 *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
14 *Ce firmware est dévelloppé sous licence creative commons CC-BY-SA.
15 *Son exécution par le microcontôleur est systématique dès son téléversement
    et ensuite à chaque mise sous tension. Il est organisé selon 2 fonctions
    principales :
16 *- la capture des températures délivrées par 2 thermomètres digitaux.
17 *- la transmission périodique de ces valeurs au e-rupteur (e-r-433).
18 *Ses fonctions principales sont complétées de fonctions secondaires (voir ci-
    dessous).
19 *Il utile des ressources extérieures (librairies, codes sources et exemples)
    développées par des informaticiens.
20 *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.
21  *
22  * 1a - Acquisition des températures
23  * L'acquisition de la température est réalisée à l'aide d'un DS18B20
    digital thermometer provides 9-bit to 12-bit Celsius temperature
    measurements.
24  * The DS18B20 communicates over a 1-Wire bus.
25  * La librarie OneWire.h met à disposition un ensemble de fonction permettant
    de gérer l'échange de données entre le DS18B20 et le microcntrôleur.
26  * Le développement de cette librarie est assuré par Paul Stoffregem,
    accessibles par les liens :
27  *
28  *- https://www.pjrc.com/teensy/td_libs_OneWire.html
29  *- https://github.com/PaulStoffregen/OneWire
30  *Il utile le code source dévelloppé par :
31  *- 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.
32  *- selon l'article https://www.carnetdumaker.net/articles/mesurer-une-
    temperature-avec-un-capteur-1-wire-ds18b20-et-une-carte-arduino-genuino/
33  *- en tenant compte que pour des raisons de sécurité inhérant au projet SCAO
    qu'il est indispensable que la capture de données en provenance
34  * des 2 thermomètres digitaux DB18B20 se réalise toujours dans le même
    ordre, en particulier après un changement de l'un ou des deux thermomètres.
35  *
36  * 1b - Mesure des tensions
37  *
38  * 1c - BITE
```

```
39    *
40    * 1d - Transmission
41    * - Mike McCauley, accesibles par les liens :
42    *- https://www.pjrc.com/teensy/td_libs_VirtualWire.html
43    *- http://www.airspayce.com/mikem/arduino/VirtualWire.pdf
44    *- https://github.com/manashmndl/VirtualWire
45    *
46    * 1e - Horodatage & Chronomètre
47    * Horodatage
48    * - Michael Margolis, accesibles par les liens :
49    *- https://www.pjrc.com/teensy/td_libs_Time.html
50    *- https://github.com/PaulStoffregen/Time
51    *- https://github.com/PaulStoffregen/Time/blob/master/examples/TimeTeensy3/
      TimeTeensy3.ino (pour la fonction RTC),
52    *  accessible également dans l'Arduino IDE (Fichier - Exemples - Time -
      TimeTeensy3)
53    *  TimeRTC.pde
54    *  example code illustrating Time library with Real Time Clock.
55    *
56    *  Chronomètre
57    * Chrono library for Arduino or Wiring by Sofian Audry and Thomas Ouellet
      Fredericks
58    * - http://github.com/SofaPirate/Chrono
59    *
60    * 1f - Visualisation des résultats
61    *
62    * 1g - Mode Sleep
63    * - Colin Duffy, pour le "mode sleep" (snooze) avec la gestion de la liaison
      série, accesible par le lien :
64    *- https://github.com/duff2013/Snooze/blob/master/examples/deepsleep/
      deepSleep_usb_serial/deepSleep_usb_serial.ino
65    *- https://github.com/duff2013/Snooze
66    *
67    * 1h - Bilan énergétique de la batterie
68    * Calculs de l'énegie électrique consommée et du ratio par rapport à la
      capacité nominale de 400 mAh
69    *
70    * 1i - Start Stop (StSp) Interrupt Service Routine (ISR)
71    */
72    /**
73    * 2 - Initialisation des paramètres
74    * Paramètres communs
75    */
76    unsigned long tt1=0;//temps de travail 1
77    unsigned long tt2=0;//temps de travail 2
78    unsigned long ti=30000000;//temps itératif
79    unsigned long ts=0;//temps de sleep
80    /**
81    * 2a - Acquisition des températures
82    * Code pour lire un thermomètre digital DS18B20 sur un bus 1-Wire.
83    */
84
85    /* Dépendance pour le bus 1-Wire */
86    #include <OneWire.h> //Chargement de la librairie OneWire.h
87    /* Broches des 2 bus 1-Wire */
88    const byte BROCHE_ONEWIRE_1 = 16;
89    const byte BROCHE_ONEWIRE_2 = 17;
90
91    /* Code de retour des 2 fonctions getT1() et getT2() */
92    enum DS18B20_RCODES {
93      READ_OK,  // Lecture ok
94      NO_SENSOR_FOUND,  // Pas de capteur
95      INVALID_ADDRESS,  // Adresse reçue invalide
96      INVALID_SENSOR  // Capteur invalide (pas un DS18B20)
97    };
98
```

```
 99
100  /* Création des 2 objets OneWire pour manipuler les 2 bus 1-Wire */
101  OneWire ds1(BROCHE_ONEWIRE_1);
102  OneWire ds2(BROCHE_ONEWIRE_2);
103  /**
104   * 2b - Mesures des tensions
105   */
106  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)
107  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)
108  float Vbat_1; //Initialisation de la variable Vbat_1 (valeur de la tension
     Vbat avant la résistance de 1 Ohm)
109  float Vbat_2; //Initialisation de la variable Vbat_2 (valeur de la tension
     Vbat après la résistance de 1 Ohm)
110  float ibat; //Initialisation de la variable ibat (valeur du courant qui
     traverse la résistance de 1 Ohm)
111  const float Vbat_limite = 4300;
112  const float Vbat_nominal = 3700;
113  const float Vbat_min = 3600;
114  const float Vbat_cut_off = 2800;
115  const int Vusb_demie = A8; //Mesure de la moitié de la tension Vusb
116  float Vusb;
117  const int V33_demie = A9; //Mesure de la moitié de la tension 3.3V
118  float V33;
119  const int MaxConv = 8192;
120  const int MaxVolt = 3272;
121  /**
122   * 2c - Built In Test Equipment (BITE)
123   * 2d - Transmission
124   * Chargement de la librairie VirtualWire - Gestion de l'émetteur 433 MHZ
125   */
126  #include <VirtualWire.h>
127  const int transmit_pin = 18;//Pin de sortie de l'émetteur
128  byte count = 0;//Initialisation du numéro du message
129  /**
130   * 2e - Horodatage & Chronomètre
131   * Horodatage
132   */
133  #include "TimeLib.h" //Include IimeLib.h library
134  /**
135   * Chronomètre
136    */
137  #include <Chrono.h> //Include Chrono.h Library
138  Chrono Chrono(Chrono::MICROS);//Instanciate a Chrono object
139
140  /**
141   * 2f - Visualisation des résultats
142   *  Les résultats du BITE sont visalisés par 3 leds :
143  verte : l'allumage témoigne d'un bon fonctionnement
144          l'extinction peut être envisagée en mode sleep
145  jaune : allumée quand l'émission commence
146          éteinte quand l'émission s'arrête
147  rouge : l'allumage témoigne d'une ou plusieurs anomalies
148  **/
149  const int led_pin_v = 13;//Led verte
150  const int led_pin_j = 14;//Led jaune
151  const int led_pin_r = 15;//Led rouge
152  /**
153   * 2g - Mode sleep
154   * Chargement de la librairie
155   */
156  #include <Snooze.h>
157  // Load timer and USBSerial drivers
```

```
158  SnoozeTimer timer;
159  SnoozeUSBSerial usb;
160
161  /***************************************************
162   Install drivers to a SnoozeBlock, timer to wake and USB Serial Driver to
163   fix printing to serial monitor after sleeping.
164   ***************************************************/
165  SnoozeBlock config_teensy32(usb, timer);
166  /**
167   * 2h - Bilan énergétique de la batterie
168   * Chargement de la librairie
169   */
170  unsigned long Etot=0; //Energie électrique totale consommée en joule
171  unsigned long Et=0; //Energie électrique consommée pendant le travail
172  unsigned long Es=0; //Energie électrique consommée pendant le sommeil (sleep)
173  unsigned long Ec=0; //Energie électrique consommée cumulée
174  /** 2h - Bilan énergétique de la batterie
175  *2i - Start Stop (StSp) Interrupt Service Routine (ISR)
176   *
177   */
178  const byte interruptPin = 11;
179  volatile byte stsp = LOW;
180  /**
181   * 3 - Fonctions spécifiques
182   * 3a-1 - Fonction d'acquisition de la température via le 1er thermomètre
     digital DS18B20 (ds1).
183   * Fonction de lecture de la température via un thermomètre digital DS18B20
     câblé sur le 1er bus ds1.
184   */
185  byte getT1(float *T1, byte reset_search) {
186    byte data[9], addr[8];
187    // data[] : Données lues depuis le scratchpad
188    // addr[] : Adresse du module 1-Wire détecté
189
190    /* Reset le bus 1-Wire si nécessaire (requis pour la lecture du premier
     capteur) */
191    if (reset_search) {
192      ds1.reset_search();
193    }
194
195    /* Recherche le prochain capteur 1-Wire disponible */
196    if (!ds1.search(addr)) {
197      // Pas de capteur
198      return NO_SENSOR_FOUND;
199    }
200
201    /* Vérifie que l'adresse a été correctement reçue */
202    if (OneWire::crc8(addr, 7) != addr[7]) {
203      // Adresse invalide
204      return INVALID_ADDRESS;
205    }
206
207    /* Vérifie qu'il s'agit bien d'un DS18B20 */
208    if (addr[0] != 0x28) {
209      // Mauvais type de capteur
210      return INVALID_SENSOR;
211    }
212
213    /* Reset le bus 1-Wire et sélectionne le capteur */
214    ds1.reset();
215    ds1.select(addr);
216
217    /* Lance une prise de mesure de température et attend la fin de la mesure */
218    ds1.write(0x44, 1);
219    delay(800);
220
```

```
221     /* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de
      lecture du scratchpad */
222     ds1.reset();
223     ds1.select(addr);
224     ds1.write(0xBE);
225 
226    /* Lecture du scratchpad */
227     for (byte i = 0; i < 9; i++) {
228       data[i] = ds1.read();
229     }
230 
231     /* Calcul de la température en degré Celsius */
232     *T1 = (int16_t) ((data[1] << 8) | data[0]) * 0.0625;
233 
234     // Pas d'erreur
235     return READ_OK;
236   }
237    /** 3a-2 - Fonction d'acquisition de la température via le 2ème thermomètre
      digital DS18B20 (ds2).
238    */
239   byte getT2(float *T2, byte reset_search) {
240     byte data[9], addr[8];
241     // data[] : Données lues depuis le scratchpad
242     // addr[] : Adresse du module 1-Wire détecté
243 
244     /* Reset le bus 1-Wire ci nécessaire (requis pour la lecture du premier
      capteur) */
245     if (reset_search) {
246       ds2.reset_search();
247     }
248 
249     /* Recherche le prochain capteur 1-Wire disponible */
250     if (!ds2.search(addr)) {
251       // Pas de capteur
252       return NO_SENSOR_FOUND;
253     }
254 
255     /* Vérifie que l'adresse a été correctement reçue */
256     if (OneWire::crc8(addr, 7) != addr[7]) {
257       // Adresse invalide
258       return INVALID_ADDRESS;
259     }
260 
261     /* Vérifie qu'il s'agit bien d'un DS18B20 */
262     if (addr[0] != 0x28) {
263       // Mauvais type de capteur
264       return INVALID_SENSOR;
265     }
266 
267     /* Reset le bus 1-Wire et sélectionne le capteur */
268     ds2.reset();
269     ds2.select(addr);
270 
271     /* Lance une prise de mesure de température et attend la fin de la mesure */
272     ds2.write(0x44, 1);
273     delay(800);
274 
275     /* Reset le bus 1-Wire, sélectionne le capteur et envoie une demande de
      lecture du scratchpad */
276     ds2.reset();
277     ds2.select(addr);
278     ds2.write(0xBE);
279 
280    /* Lecture du scratchpad */
281     for (byte i = 0; i < 9; i++) {
282       data[i] = ds2.read();
```

```
283     }
284
285     /* Calcul de la température en degré Celsius */
286     *T2 = (int16_t) ((data[1] << 8) | data[0]) * 0.0625;
287
288     // Pas d'erreur
289     return READ_OK;
290   }
291   /**
292    * 3b - Fonction mesure des tensions et calibration
293    * 3c - BITE
294    * 3d - Tranmission
295    * 3e - Horodatage & Chronomètre
296    */
297    void digitalClockDisplay() {
298     // digital clock display of the time
299     Serial.print(hour());
300     printDigits(minute());
301     printDigits(second());
302     Serial.print(" ");
303     Serial.print(day());
304     Serial.print(" ");
305     Serial.print(month());
306     Serial.print(" ");
307     Serial.print(year());
308     Serial.println();
309   }
310
311   time_t getTeensy3Time()
312   {
313     return Teensy3Clock.get();
314   }
315
316   /*  code to process time sync messages from the serial port   */
317   #define TIME_HEADER  "T"   // Header tag for serial time sync message
318
319   unsigned long processSyncMessage() {
320     unsigned long pctime = 0L;
321     const unsigned long DEFAULT_TIME = 1357041600; // Jan 1 2013
322
323     if(Serial.find(TIME_HEADER)) {
324        pctime = Serial.parseInt();
325        return pctime;
326        if( pctime < DEFAULT_TIME) { // check the value is a valid time (greater
    than Jan 1 2013)
327          pctime = 0L; // return 0 to indicate that the time is not valid
328        }
329     }
330     return pctime;
331   }
332
333   void printDigits(int digits){
334     // utility function for digital clock display: prints preceding colon and
    leading 0
335     Serial.print(":");
336     if(digits < 10)
337       Serial.print('0');
338     Serial.print(digits);
339   }
340
341    /** 3f - Visualisation des résultats
342    * 3g - Mode sleep
343    * 3h - Bilan énergétique de la batterie
344    * 3i - Start Stop (StSp) Interrupt Service Routine (ISR)
345    */
346    void StSp(){
```

```
347     stsp = !stsp;
348     }
349
350   /** 4 - Fonction setup() **/
351   void setup() {
352     /* Initialisation du port série */
353     Serial.begin(9600);
354     delay(10000);
355     Serial.println("N°;date;heure;T1 (degrés C);T2 (degrés C);Vusb (mV);Vbat
    (mV);ibat (mA);V33 (mV);Ec (joules)");
356     //4a - Acquisition des températures
357     //4b - Mesure des tensions et calibration
358     analogReadResolution(13);
359     //4c - BITE
360
361     //4d - Transmission
362       // Initialise the IO and ISR
363     vw_set_tx_pin(transmit_pin);
364     vw_setup(2000);   // Bits per sec
365     //4e - Horodatage & Chronomètre
366     //setTime(15, 17, 00, 21, 01, 2019);
367     // set the Time library to use Teensy 3.0's RTC to keep time
368     setSyncProvider(getTeensy3Time);
369     while (!Serial);  // Wait for Arduino Serial Monitor to open
370     delay(100);
371     if (timeStatus()!= timeSet) {
372       Serial.println("Unable to sync with the RTC");
373     } else {
374       Serial.println("RTC has set the system time");
375     }
376     //4f - Visualisation des résultats
377     pinMode(led_pin_v, OUTPUT);
378     pinMode(led_pin_j, OUTPUT);
379     pinMode(led_pin_r, OUTPUT);
380
381    //4g - Mode sleep
382     digitalWrite(led_pin_r, HIGH);//USB serial connection is not establihed -
    Clic on serial monitor icon (right top icon)
383     while (!Serial);
384       delay(100);
385       digitalWrite(led_pin_r, LOW);
386       //Serial.println("Starting...");
387       delay(100);
388    //4h - Bilan énergétique de la batterie
389    //4i - Start Stop (StSp) Interrupt Service Routine (ISR)
390    pinMode(interruptPin, INPUT_PULLDOWN);
391    attachInterrupt(interruptPin, StSp, RISING);
392       }
393    /** 5 - Fonction loop() **/
394   void loop() {
395     Chrono.restart();  // restart the Chrono
396     //5a - Acquisition des températures
397     float T1;
398     /* Lit la température T1 */
399     if (getT1(&T1, true) != READ_OK) {
400       Serial.println(F("Erreur de lecture du capteur"));
401       return;
402     }
403     float T2;
404     /* Lit la température ambiante à ~1Hz */
405     if (getT2(&T2, true) != READ_OK) {
406       Serial.println(F("Erreur de lecture du capteur"));
407       return;
408     }
409
410     //5b - Mesure des tensions et calibration
```

```
411     Vusb=map (2.0038*analogRead(Vusb_demie), 0, MaxConv, 0, MaxVolt);
412     Vbat_1=map (2.0038*analogRead(Vbat_demie_1), 0, MaxConv, 0, MaxVolt);
413     Vbat_2=map (2.0038*analogRead(Vbat_demie_2), 0, MaxConv, 0, MaxVolt);
414     ibat=Vbat_1 - Vbat_2;
415     V33=map (2.0038*analogRead(V33_demie), 0, MaxConv, 0, MaxVolt);
416
417     //5c - BITE
418     int tmax=25;
419     if (T2 >= tmax){digitalWrite(led_pin_r, HIGH);} else
      {digitalWrite(led_pin_r, LOW);}
420     if (Vbat_2 <= Vbat_limite && Vbat_2 > Vbat_min){digitalWrite(led_pin_v,
      HIGH);} else {digitalWrite(led_pin_v, LOW);}
421     if (Vbat_2 > Vbat_limite || Vbat_2 <= Vbat_cut_off){digitalWrite(led_pin_r,
      HIGH);} else {digitalWrite(led_pin_r, LOW);}
422
423     //5d - Transmission
424     /* Transmission des données à l'e-r-433 */
425     /*La fonction de transmission vw_send(message, length) : transmit a
      message, "message" is an array of the bytes to send,
426     and "length" is the number of bytes stored in the array.*/
427     char msg[7] = {'h','e','l','l','o',' ','#'};
428     msg[6] = count;
429     time_t heure[1] = {now()};
430     float mesure[5]= {T1, T2, Vusb, Vbat_1, ibat};
431     unsigned long Energie[1] = {Ec};
432     digitalWrite(led_pin_j, HIGH); // Flash a light to show transmitting
433     vw_send((uint8_t *)msg, 7); //Transmission hello et count
434     vw_wait_tx(); // Wait until the whole message is gone
435     vw_send((uint8_t *)heure, 1); //Transmission de l'heure
436     vw_wait_tx(); // Wait until the whole message is gone
437     vw_send((uint8_t *)mesure, 5); //Transmission T1, T2, Vusb, Vbat, ibat
438     vw_wait_tx(); // Wait until the whole message is gone
439     vw_send((uint8_t *)Energie, 1); //Transmission de l'énergie consommée
440     vw_wait_tx(); // Wait until the whole message is gone
441     digitalWrite(led_pin_j, LOW);
442     count = count + 1;
443     delay(100);
444
445     //5e - Horodatage & Chronomètre
446     time_t t = now();
447       if (Serial.available()) {
448       time_t t = processSyncMessage();
449       if (t != 0) {
450         Teensy3Clock.set(t); // set the RTC
451         setTime(t);
452       }
453     }
454     //digitalClockDisplay();
455     delay(1000);
456     //5g - Mode sleep
457
458         /*******************************************************
459        Set Low Power Timer wake up in milliseconds.
460        *******************************************************/
461     tt1 = (Chrono.elapsed());
462     ts = ti - (tt1 + tt2);
463     timer.setTimer(ts/1000);// milliseconds
464     delay(200);
465     if (sleep) {Snooze.deepSleep( config_teensy32 );}// return module that woke
      processor
466     if (!sleep) {delay(ts/1000);}
467     Chrono.restart();  // restart the Chrono
468     // wait for serial monitor
469       elapsedMillis time = 0;
470       while (!Serial && time < 1000) {
471           Serial.write(0x00);// print out a bunch of NULLS to serial monitor
```

```
472           digitalWriteFast(led_pin_v, HIGH);
473           delay(30);
474           digitalWriteFast(led_pin_v, LOW);
475           delay(30);
476       }
477       // normal delay for Arduino Serial Monitor
478       delay(200);
479       delay(1000);
480
481     //5f - Visualisation des résultats
482     Serial.print(count);
483     Serial.print(";");
484     Serial.print(day (t));
485     Serial.print("/");
486     Serial.print(month (t));
487     Serial.print("/");
488     Serial.print(year (t));
489     Serial.print(";");
490     Serial.print(hour (t));
491     Serial.print(":");
492     Serial.print(minute (t));
493     Serial.print(":");
494     Serial.print(second (t));
495     Serial.print(";");
496     /* Affiche T1 et T2 */
497     //Serial.print(F("T1 : "));
498     Serial.print(T1, 2);
499     Serial.write(";"); // Caractère degré
500     //Serial.write('C');
501     //Serial.print(";");
502     //Serial.print(F("T2 : "));
503     Serial.print(T2, 2);
504     Serial.write(";"); // Caractère degré
505     //Serial.write('C');
506     //Serial.print(";");
507     /* Affiche les tensions et le courant consommé ibat*/
508     //Serial.print(" - Vusb : ");
509     Serial.print(Vusb);
510     Serial.print(";");
511     //Serial.print(" Vbat : ");
512     Serial.print(Vbat_1);
513     Serial.print(";");
514     //Serial.print(" ibat : ");
515     Serial.print(ibat);
516     Serial.print(";");
517     //Serial.print(" V33 : ");
518     Serial.print(V33);
519     Serial.print(";");
520     //Serial.println(";");
521     //Serial.print(" ti : ");
522     //Serial.print(ti);
523     //Serial.print(" tt1 : ");
524     //Serial.print(tt1);
525     tt2 = (Chrono.elapsed());
526     //Serial.print(" tt2 : ");
527     //Serial.print(tt2);
528     //Serial.print(" ts : ");
529     //Serial.println(ts);
530
531   //5h - Bilan énergétique de la batterie
532   Et = ((Vbat_1/1000) * ibat)*((tt1 + tt2)/1000000);
533   if (sleep) {Es = ((Vbat_1/1000) * 1.5)*(ts/1000000);}
534   if (!sleep) {Es = ((Vbat_1/1000) * ibat)*(ts/1000000);}
535   Etot = Et + Es;
536   Ec = Ec + Etot;
537   //Serial.print(" Energie électrique Ec (joules) : ");
```

```
538    Serial.println(Ec);
539    //5i - Start Stop (StSp) Interrupt Service Routine (ISR)
540    digitalWrite(led_pin_v, stsp);
541      }
```