# RADEXPO - Code Arduino

```
/*****************************************************************************************
RadExpo_IR   ==> sur cate N°1.    Carte n°2 gérée par prog RadExpo_SD
YLC - 10/04/2015
Affichage des mesures sur écran OLED 128x64 + envoi données sur port série vers carte N°2
+ affichage date/heure reçu de la carte n°2 et réglage date/heure retransmis ensuite à la carte n°2
Les comptages affichés au démarrage tiennent compte des données déjà enregistrée dans la carte SD.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Messages du port série :
format "initcompteurs" récept = inittot/nb visit./durée totale/durée mini/durée maxi/durée moy/n°appareil$
                        inittot/xxxxx/xxxxxxx/xxxxx/xxxxx/xxxxx/xx$
format "demdatheure"     émet = demheur$
                  récept = affheur/hhmmss/JJMMAAAA/j$
format "reglageheure"    émet = regheur/hhmmss/JJMMAAAA/j$
format "logvisite"       émet = radexpo/n°visiteur/durée de la visite en sec$
                        radexpo/xxxxx/xxxxx$
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NB : Utilisation d'un détecteur de mouvement infrarouge (PIR sensor).
   Ce détecteur envoie une tension sur la pin de sortie s'il détecte un mouvement.  Toutefois,
   il repasse à l'état bas (0v) de temps en temps même s'il détecte toujours un mouvement.
   Pour éviter d'interprêter ces passages à l'état bas de courte durée comme une absence de présence,
   on ignore les phases à 0v sur la pin de détection si elles font moins de 5 secondes. On réalise ainsi
   une détection approximative de présence à condition que le visiteur ne reste pas totalement immobile.
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ATTENTION si modif, ne pas augmenter la mémoire dynamique utilisée pour que le prog puisse fonctionner.
Actuellement :  1659 octets pour variables globales et 389 octets libres pour les variables locales.
*****************************************************************************************/

#include <SPI.h>                  // param écran OLED
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_SDA   9
#define OLED_SCL   10
#define OLED_DC    11
#define OLED_RST   13
#define OLED_CS    12          // non connecté
Adafruit_SSD1306 display(OLED_SDA, OLED_SCL, OLED_DC, OLED_RST, OLED_CS);
#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

#define bout2 3         // bouton 2 (bouton bleu) sur pin 3
#define bout3 4         // bouton 3 (bouton rouge) sur pin 4
#define pir 5             // pin réception états du PIR
unsigned long debpre = 0;          // temps de début présence
unsigned long debbas = 0;          // temps de début état bas
unsigned int durpres;         // duree de la présence actuelle en sec.
unsigned int durtot = 0;      // duree total des présences en sec.
unsigned int durmin = 9999;   // durée présence minimale en sec.
unsigned int durmax = 0;      // durée présence maximale en sec.
unsigned int durmoy = 0;      // durée présence moyenne en sec.
unsigned int nbtot = 0;       // nb total des visites
unsigned int durdet;          // duree de détection active par le PIR en sec.
boolean presence, indmod, etahigh = false;   // états présence,indic de modif heure et absence
String sec,minu,heu,jj,mm,aaaa,js;   // variables heure et date
String enreg, mess1, mess2, numapp;
char action = '0';             // indicateur de la phase active du programme
unsigned long timpre,timbou = 0;  // pour eviter le rebond sur appui bouton 1
int i, ztra = 0;               // variables de travail
```

```
void setup() {
  Serial.begin(9600);       // vitesse communication série avec carte 2
  enreg.reserve(42);     // réserver allocation memoire taille maxi du string
  mess1.reserve(17);     // idem
  mess2.reserve(8);      // idem
  numapp.reserve(2);     // idem
  display.begin(SSD1306_SWITCHCAPVCC);   // OLED avec alim 3,3v arduino
  display.setTextColor(WHITE);
  display.clearDisplay();         // affacer écran
  pinMode(pir, INPUT);            // detecteur présence en lecture
  attachInterrupt(0, interbout1, RISING );   // bouton 1 sur broche 2 (interruption #0) pour demande heure
  pinMode(bout2, INPUT);                  // bouton 3 sur pin 4  pour modifier date/heure
  pinMode(bout3, INPUT);                  // bouton 3 sur pin 4  pour modifier date/heure
  calibrage();                  // temps nécéssaire à la mise en route du PIR
  serialEvent();                // récupérer comptages initiaux depuis fichier de la carte 2
  initcompteurs();
  messnumapp();
}

void loop() {
  if (action == '0') {
    detection();
    display.clearDisplay();
    if (millis() - debbas < 60 000 || presence == true) affimesure();  // affichage mesures pendant 1 min après la fin de visite
    display.display();            // sinon aucun affichage (écran noir)
  }
  if (action == '1')  demdateheure();
  if (action == '3')  affdateheure();
  if (action == '4')  testbout2();
  if (action == '5')  moddateheure();
  if (action == '6')  envoireglage();
}
//_____________________________________________________________________________________
void initcompteurs() {      // réinit des compteurs au redémarrage éventuel
  if (enreg.substring(0,7) == F("inittot"))  {    // réception valeurs actuelle des totaux
    nbtot = enreg.substring(8,13).toInt();
    durtot = enreg.substring(14,21).toInt();
    durmin = enreg.substring(22,27).toInt();
    durmax = enreg.substring(28,33).toInt();
    durmoy = enreg.substring(34,39).toInt();
    numapp = enreg.substring(40,42);          // numéro d'appareil
  }
}
void affimesure() {
    display.setTextSize(1);
    display.setCursor(0,0);
    display.print(mess1);
    display.setCursor(0,16);
    display.print(F("Min. "));
    if (durmin < 99 999)  display.print(durmin);
    display.setCursor(0,26);
    display.print(F("Max. "));
    if (durmax > 0)  display.print(durmax);
    display.setCursor(0,36);
    display.print(F("Moy. "));
    if (durmoy > 0)  display.print(durmoy);
    display.setCursor(60,41);
    display.print(F("D.tot.(min)"));
    display.setCursor(70,16);
    display.print(F("Nb.visit."));
    display.setTextSize(2);
    display.setCursor(77,25);
    display.print(nbtot);
```

```
    display.setCursor(65,50);
    display.print(durtot/60);
    if (etahigh) display.fillCircle(120, 5, 5, 1);    // voyant de détection de mouvement en cours
    display.setCursor(0,50);
    if (presence)  display.print(durdet);
    else display.print(durpres);
}
void calibrage() {       // affichage du décompte du temps de calibrage (30 sec)
  i = 30;
  while (i > 0) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(5,0);
    display.print(F("Calibrage"));
    display.setTextSize(3);
    display.setCursor(50,30);
    display.print(i);
    display.display();
    delay(1000);            // attente 1 sec
    i--;
  }
  display.clearDisplay();
  display.display();
}
void interbout1() { // bouton jaune (interruption): affichage date/heure ou retour ou choix de variable à modifier
  timbou = millis();
  if (timbou - timpre > 300) {    // temporisation 1/2 sec anti-rebond du bouton
    if (action == '0')  action = '1';    // demande affichage date/heure
    if (action == '2' || action == '4') {    // quitter affichage date/heure (sans modif)
      action = '0';
      messnumapp();
    }
    if (action == '5') i++;
    timpre = timbou;
  }
}
void testbout2() {              // bouton bleu  (appel programmé)
  if (digitalRead(bout2) == HIGH) {    // si appuyé = demande de modif ou validation modif
    timbou = millis();
    if (timbou - timpre > 300) {    // temporisation 1/2 sec anti-rebond du bouton
      if (action == '4')  action = '5';     //  demande de modif date / heure affichée
      if (action == '5' && i == 7) action = '6';    // modif date/heure validée
      timpre = timbou;
    }
  }
}
void demdateheure() {   // envoi message demande date/heure à carte n°2
  Serial.flush();
  Serial.print(F("demheur$"));
  action = '2';
  delay(500);
}
void serialEvent() {     // interruption du port série quand réception message
  enreg = "";
  char car;
  if(Serial.available() != 0)  delay(500);  // attendre de recevoir tout le message
  while (Serial.available() != 0) {
      car = (Serial.read());       // lit et vide le buffer du port série
      if (car == '$')   break;     // car $ marque la fin d'enregistrement
      enreg = enreg + car;         // reconstitution du message reçu
  }
  if (enreg.substring(0,7) == F("affheur"))  action = '3';     // réponse à la demande date/heure
}
```

```
void affdateheure() {     // affichage date /heure reçue
  action = '4';
  i = 0;
  indmod = false;
  heu = enreg.substring(8,10);
  minu = enreg.substring(10,12);
  sec = enreg.substring(12,14);
  jj = enreg.substring(15,17);
  mm = enreg.substring(17,19);
  aaaa = enreg.substring(19,23);
  js = enreg.substring(24,25);
  mess1 = F("Date/heure");
  mess2 = F("actuelle");
  affichdatheu();
  delay(200);
}
void moddateheure() {      // modification date, heure et jour de la semaine
  mess1 = F("Modif");
  if (i > 7)  i = 0;
  if (i == 0) {
    ztra = js.toInt();
    incredatheur();
    if (ztra > 7)  ztra = 1;
    mess2 = F("JOUR SEM");
  }
  if (i == 1) {
    ztra = heu.toInt();
    incredatheur();
    if (ztra > 23)  ztra = 0;
    mess2 = F("HEURES");
  }
  if (i == 2) {
    ztra = minu.toInt();
    incredatheur();
    if (ztra > 59)  ztra = 0;
    mess2 = F("MINUTES");
  }
  if (i == 3) {
    ztra = sec.toInt();
    incredatheur();
    if (ztra > 59)  ztra = 0;
    mess2 = F("SECONDES");
  }
  if (i == 4) {
    ztra = jj.toInt();
    incredatheur();
    if (ztra > 31)  ztra = 1;
    mess2 = F("JOUR");
  }
  if (i == 5) {
    ztra = mm.toInt();
    incredatheur();
    if (ztra > 12)  ztra = 1;
    mess2 = F("MOIS");
  }
  if (i == 6) {
    ztra = aaaa.toInt();
    incredatheur();
    if (ztra > 2030)  ztra = 2015;
    mess2 = F("ANNEE");
  }
  if (i == 0)  js = String(ztra);
  if (i == 1)  heu = zerogauch2(ztra);
```

```
  if (i == 2)  minu = zerogauch2(ztra);
  if (i == 3)  sec = zerogauch2(ztra);
  if (i == 4)  jj = zerogauch2(ztra);
  if (i == 5)  mm = zerogauch2(ztra);
  if (i == 6)  aaaa = String(ztra);
  if (i == 7) {           // si tout a été modifié on peut valider (ou sortir sans modif)
    delay(300);
    mess1 = (F("SI OK "));
    mess2 = (F("VALIDER"));
    testbout2();          // test si validation des modifs demandé
  }
  affichdatheu();       // on réaffiche date/heure au fur et à mesure des modification
}
void incredatheur() {     // routine d'incrémentation de la variable en cours de modif
  if (digitalRead(bout3) == HIGH) {    // bouton rouge (appui pour faire +1 sur la variable en cours)
    timbou = millis();
    if (timbou - timpre > 300) {    // temporisation 1/3 sec anti-rebond du bouton
      ztra++;
      indmod = true;
      timpre = timbou;
    }
  }
}
void affichdatheu()  {      // affichage date, heure et jour de la semaine en clair
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print(mess1);
  display.print(F(" "));
  display.print(mess2);
  display.setTextSize(2);
  display.setCursor(0,16);
  if (js == "1") display.print(F("Dimanche"));
  if (js == "2") display.print(F("Lundi"));
  if (js == "3") display.print(F("Mardi"));
  if (js == "4") display.print(F("Mercredi"));
  if (js == "5") display.print(F("Jeudi"));
  if (js == "6") display.print(F("Vendredi"));
  if (js == "7") display.print(F("Samedi"));
  display.setCursor(0,32);
  display.print(heu);
  display.print(F(":"));
  display.print(minu);
  display.print(F(":"));
  display.print(sec);
  display.setCursor(0,50);
  display.print(jj);
  display.print(F("/"));
  display.print(mm);
  display.print(F("/"));
  display.print(aaaa);
  display.display();
}
void detection()  {       // routine de détection de présence
  if (presence)  durdet = (millis() - debpre) / 1000;   // actualiser duree de la présence effective
  if(digitalRead(pir) == HIGH)  {   // détection active
    etahigh = true;
    if (presence == false) {    // démarrage de la détection
      presence = true;
      mess1 = F("Presence visit.");
      debpre = millis();    // stockage début détection présence
    }
  }
```

```
  else {
    if (etahigh) {
      debbas = millis();    // fin de détection
      etahigh = false;
    }
    if (presence) {
      if (millis() - debbas > 5000) {  // on attend 5000 millisec pour être sur de la fin de la visite
        durpres = (debbas - debpre) / 1000;   //calcul duree de la presence effective en sec.
        presence = false;
        messnumapp();
        if (durpres > 2) {   // comptabiliser seulement présences de plus de 2 sec
          nbtot++;              // nombre de visites
          durtot = durtot + durpres;        // total cumulé temps de visite
          if (durpres > durmax)   durmax = durpres;   //  durée maxi en sec.
          if (durpres < durmin)   durmin = durpres;   //  durée mini en sec.
          durmoy = durtot / nbtot;                // calcul durée moyenne en sec.
          durdet = 0;
          char buf[5] = "";
          Serial.flush();        // envoi à carte n°2 pour enregistrement de la visite
          Serial.print(F("radexpo/"));
          sprintf(buf,"%05d", nbtot);
          Serial.print(buf);
          Serial.print(F("/"));
          sprintf(buf,"%05d", durpres);
          Serial.print(buf);
          Serial.print(F("$"));
        }
      }
    }
  }
}
void envoireglage() {    // Envoi à carte n°2 de date/heure si modifiée
  action = '0';
  if (indmod == true) {      // si modif date, heure ou jours semaine envoi à carte n°2
    Serial.flush();
    Serial.print(F("regheur/"));
    Serial.print(heu);
    Serial.print(minu);
    Serial.print(sec);
    Serial.print(F("/"));
    Serial.print(jj);
    Serial.print(mm);
    Serial.print(aaaa);
    Serial.print(F("/"));
    Serial.print(js);
    Serial.println(F(";$"));
    mess1 = F("Modif enregistree");
  }
  else mess1 = F("Pas de modif");
}
String zerogauch2(int nombre) {   // ajouter zéro non significatif si nombre nn d'un 1 seul chiffre
  String valnch = "";
  if (nombre < 10)   { valnch = F("0"); }
  valnch += String(nombre, DEC);
  return valnch;              // renvoie le nombre en string de 2 chiffres
}
void messnumapp() {
  mess1 = F("Appareil ");
  mess1 = mess1 + numapp;
  mess2 = "";
}
//*************************************************************************************
```

```
/*********************************************************************************************
RasExpo_SD   ==> sur carte n°2.    Carte n°1 gérée par prog RadExpo_IR
YLC - 10/04/2015
programme de la carte n°2 pour stockage dans SD card des visites recues de la carte n°1
avec horodatage par DS1307 (sans la lib RTC)+ réglages date/heure mis à jour par carte n°1
Lecture au démarrage des données deja enregistrée pour initialiser les compteurs de la carte n°1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Messages du port série :
format "initcompteurs"   émet = inittot/nb visit./durée totale/durée mini/durée maxi/durée moy/n°appareil$
                         inittot/xxxxx/xxxxxxx/xxxxx/xxxxx/xxxxx/xx$
format "demdatheure" :  récept = demheur$
                    émet = affheur/hhmmss/JJMMAAAA/j$
format "reglageheure" : récept = regheur/hhmmss/JJMMAAAA/j$
format "logvisite" :    récept = radexpo/n°visiteur/durée de la visite en sec$
                        radexpo/xxxxx/xxxxx$
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
format enregt fichier radexpo :  (format .CSV pour tableur)     long 44 fixe
  radexpo;n°appareil;n°visiteur;durée(sec);AAAA;MM;JJ;hh;mm;ss;jour de la semaine (Dim=1)
  radexpo;xx;xxxxx;xxxxx;AAAA;MM;JJ;hh;mm;ss;x
*********************************************************************************************/

#include <SPI.h>
#include <SD.h>
File fichiersd;
#include "Wire.h"
#include <DS1307.h>

#define ledalarm 7
String timbre(14);        //  timbre pour horodatage des enregistrements
char car;                 // caractère lu sur port série
String enrfi = "";        // buffer des données a écrire sur fichier
String enreg = "";        // buffer des données reçues sur port série
boolean pd = false;       // ind présence de données
boolean testpf = false;       // test présence fichier apres écriture
boolean indalarm = false;    // ind anomalie de fontionnement
int sec,minu,heu,jj,mm,aa,js;   // variables horloge
unsigned long durtot = 0;      // duree total des présences en sec.
unsigned long durmin = 9999;   // durée présence minimale en sec.
unsigned long durmax = 0;      // durée présence maximale en sec.
unsigned long durmoy = 0;      // durée présence maximale en sec.
unsigned int nbtot = 0;        // nb total des visites
char buf[7] = "";
String numapp = "";           // numéro d'appareil (cf fichier param)

void setup() {
  pinMode(53, OUTPUT);   // pour SD card (obligatoire)
  pinMode(ledalarm, OUTPUT);    // led anomalie de fonctionnement
  digitalWrite(ledalarm,LOW);   // éteinte par défaut   :-))
  Serial.begin(9600);
  if (!SD.begin(4)) { indalarm = true; }   // Alarme si pas de SD card sur pin 4
  lectparam();            // lire le n° d'appareil dans le fichier param
  delay(20 000);           // attendre 20 sec avant envoi récap comptage vers carte 1
  recupcompta();          // lire et compter données déjà enregistrée sur SD card au démarrage
  Serial.flush();        // vider buffer en sortie du port série
  Serial.print(F("inittot/"));
  sprintf(buf,"%05d/", nbtot);
  Serial.print(buf);
  sprintf(buf,"%07d/", durtot);
  Serial.print(buf);
  sprintf(buf,"%05d/", durmin);
  Serial.print(buf);
  sprintf(buf,"%05d/", durmax);
```

```
  Serial.print(buf);
  sprintf(buf,"%05d/", durmoy);
  Serial.print(buf);
  Serial.print(numapp);
  Serial.println(F("$"));
}

void loop() {
  testalarme();
  enreg = "";
  if(Serial.available() != 0) delay(500);  // attendre de recevoir tout le message
  while (Serial.available()!= 0) {
    car = (Serial.read());   // on lit en vidant le buffer d'entrée du port série
    if (car == '$') { pd = true;  break; }    // car $ marque la fin d'enregistrement
    enreg = enreg + car;      // concaténer les caratères lus
  }
  if (pd) {       // si données reçues
    if (enreg.substring(0,7) == F("radexpo"))   logvisite();
    if (enreg.substring(0,7) == F("regheur"))   reglageheure();
    if (enreg.substring(0,7) == F("demheur"))   affdatheure();
  pd = false;
  }
}

void logvisite() {    // enregistrement d'une visite
  enrfi = enreg.substring(0,7);    // identif enregt.  "radexpo"
  enrfi += ";";      // séparateur
  enrfi += numapp;       // n° d'appareil lu sur fichier param
  enrfi += ";";      // séparateur
  enrfi += enreg.substring(8,13);    // n° visiteur
  enrfi += ";";      // séparateur
  enrfi += enreg.substring(14,19);    // durée de visite
  enrfi += ";";      // séparateur
  // ajouter AAAA;MM;JJ;hh;mm;ss et jour semaine à l'enregistrement
  enrfi += (RTC.get(DS1307_YR,true));                  // annee
  enrfi += ";";      // séparateur
  enrfi += zerogauch2(int(RTC.get(DS1307_MTH,false)));  // mois
  enrfi += ";";      // séparateur
  enrfi += zerogauch2(int(RTC.get(DS1307_DATE,false))); // jour
  enrfi += ";";      // séparateur
  enrfi += zerogauch2(int(RTC.get(DS1307_HR,false)));   // heure
  enrfi += ";";      // séparateur
  enrfi += zerogauch2(int(RTC.get(DS1307_MIN,false)));  // minutes
  enrfi += ";";      // séparateur
  enrfi += zerogauch2(int(RTC.get(DS1307_SEC,false)));  // secondes
  enrfi += ";";      // séparateur
  enrfi += int(RTC.get(DS1307_DOW,false));              // jour de la semaine
  fichiersd = SD.open("radexpo.csv", FILE_WRITE);
  if (fichiersd) {
    fichiersd.seek(fichiersd.size());   // positionnement à la fin du fichier
    fichiersd.println(enrfi);
    fichiersd.close();
  }
  testpf = SD.exists("radexpo.csv");       // contrôle présence du fichier
  if (testpf != true)  indalarm = true;    // alarme si erreur  fichier
}
String zerogauch2(int nombre) {   // ajouter zéros non significatifs si nombre nn < taille voulue
  String valnch = "";
  if (nombre < 10)   { valnch = F("0"); }
  valnch += String(nombre, DEC);
  return valnch;      // renvoie le nombre en string de 2 chiffres
}
void reglageheure() {      // mettre à jour date et heure reçus
```

```
  heu = enreg.substring(8,10).toInt();
  minu = enreg.substring(10,12).toInt();
  sec = enreg.substring(12,14).toInt();
  jj = enreg.substring(15,17).toInt();
  mm = enreg.substring(17,19).toInt();
  aa = enreg.substring(21,23).toInt();   // on ne prend que les 2 derniers chiffres de l'année pour la mise à jour
  js = enreg.substring(24,25).toInt();
  RTC.stop();
  RTC.set(DS1307_SEC,sec);
  RTC.set(DS1307_MIN,minu);
  RTC.set(DS1307_HR,heu);
  RTC.set(DS1307_DATE,jj);
  RTC.set(DS1307_MTH,mm);
  RTC.set(DS1307_YR,aa);    // année en 2 chiffres a la mise à jour
  RTC.set(DS1307_DOW,js);     // jour de la semaine
  RTC.start();
}
void affdatheure() {       // renvoi du message date et heure actuelle
  Serial.flush();          // vider buffer en sortie du port série
  Serial.print(F("affheur/"));
  Serial.print(zerogauch2(int(RTC.get(DS1307_HR,true))));
  Serial.print(zerogauch2(int(RTC.get(DS1307_MIN,false))));
  Serial.print(zerogauch2(int(RTC.get(DS1307_SEC,false))));
  Serial.print(F("/"));
  Serial.print(zerogauch2(int(RTC.get(DS1307_DATE,false))));
  Serial.print(zerogauch2(int(RTC.get(DS1307_MTH,false))));
  Serial.print(RTC.get(DS1307_YR,false));      // année en 4 chiffres à la lecture
  Serial.print(F("/"));
  Serial.print(RTC.get(DS1307_DOW,false));
  Serial.println(F("$"));
}
void testalarme() {  // on fait clignoter la led d'alarme
  if(indalarm == true) {
    digitalWrite(ledalarm,HIGH);
    delay(500);
    digitalWrite(ledalarm,LOW);
    delay(500);
  }
}
void recupcompta() {
  testpf = SD.exists("radexpo.csv");       // contrôle présence du fichier
  if (testpf == true)  {
    fichiersd = SD.open("radexpo.csv",FILE_READ);
    while (fichiersd.available()) {
      String hdur = "";
      for (int i = 0; i < 46; i++) {      // enregt = 44 + cr/lf
        char c = fichiersd.read();
        if ((i>16) && (i<22))  hdur += c ;
      }
      unsigned int dur = hdur.toInt();
      if (dur > 0) {
        if (dur < durmin)   durmin = dur;
        if (dur > durmax)   durmax = dur;
        durtot = durtot + dur;
        nbtot++;
      }
    }
    fichiersd.close();
    durmoy = durtot / nbtot;
  }
}
void lectparam() {
  testpf = SD.exists("param.txt");       // contrôle présence du fichier
```

```
  if (testpf == true)  {
    fichiersd = SD.open("param.txt",FILE_READ);
    while (fichiersd.available()) {
      for (int i = 0; i < 2; i++) {      // enregt = 2 + cr/lf
        char c = fichiersd.read();
        numapp += c ;
      }
    }
    fichiersd.close();
  }
  else numapp = "01";   // en l'absence de fichier param on attribue le n° d'appareil 01
}
//**********************************************************************
```