RADIO FM YLC - code Arduino

```
/*********************************************************************
 * Radio FM Stereo
 * auth: Yves Le Chevalier  -  Maj le 30/04/2017
 *
 * CONNEXIONS : Arduino Mega > Full graphic smart controller
 *  pin 6  < detection présence carte SD     => EXP 2  pin 7
 *  pin 50 < MISO SD card                    => EXP 2  pin 1
 *  pin 51 < MOSI SD card                    => EXP 2  pin 6
 *  pin 52 < CLK SD card                     => EXP 2  pin 2
 *  pin 53 < SS : CS : SD card sélection     => EXP 2  pin 4
 *
 *  pin 2 + INPUT_PULLUP <  encodeur rot. vers droite (Inter 4)         => EXP 2  pin 3
 *  pin 3 + INPUT_PULLUP <  encodeur rot. vers gauche (Inter 5)         => EXP 2  pin 5
 *  pin 18 + INPUT_PULLUP <  bouton raz frequences mémorisées (Inter 3) => EXP 2  pin 8
 *  pin 4 + INPUT_PULLUP <  Push button de l'encodeur rotatif           => EXP 1  pin 2
 *
 *  pin 13 > LCD : EN (CLK)   => EXP 1  pin 5
 *  pin 11 > LCD : RW (MOSI)  => EXP 1  pin 3
 *  pin 10 > LCD : RS (MISO)  => EXP 1  pin 4
 *  pin 7 > Beeper            => EXP 1  pin 1
 *
 * CONNEXIONS : Arduino Mega > RDA5807M (connexion I2C)
 *  pin 20 SDA >  RDA5807 SDA
 *  pin 21 SCL >  RDA5807 SCL
 *
 * CONNEXIONS : Arduino Mega <> autres composants
 *  pin A0 < potentiomètre volume son
 */
#define TRACES 0          // 1 pour activer trace sur port série, 0 pour la désactiver

#include "U8g2lib.h"
#include <SD.h>
#include <radio.h>
#include <RDA5807M.h>
#include <RDSParser.h>
#include <pushbuttoncommand.h>

// Ecran full graphic 128x64 SPI
U8G2_ST7920_128X64_F_SW_SPI u8g2(U8G2_R0, 13, 11, 10);  // full screen mode (Clk=13, MOSI=11, CS=10)
#define pol_5an  u8g2_font_baby_tf       // police 5 pix alpha-num
#define pol_9an  u8g2_font_7x13B_tf      // police 9 pix alpha-num
#define pol_16n  u8g2_font_10x20_tn      // police 16 pix numerique

// Radio
RDA5807M radio;    // Create an instance of a RDA5807 chip radio
RDSParser rds;
static unsigned long nextFreqTime = 0;
int lastf = 0;
int frequence = 0;
static unsigned long lastfretime;
String station;            // nom de la station
String radio_text;         // RDS : RT = radio text     64 car
String message;            // message à afficher (RDS ou message du logiciel)
int volume = 1;            // volume 1 par défaut  (0 à 15)
int vol_old;               // mémo volume
unsigned int etat[6];
boolean stereo;
float freq_min = 8760;     // frequence FM minimale en France 87.6
float freq_max = 10790;    // frequence FM maximale en France 107.9
float freq_memo[10];       // frequences mémorisees
```

```
float freq_allu = freq_min; // frequence a allumage du poste
int nb_memo = 0;            // Nb de frequences mémorisees
int der_memo = 0;           // indice derniere frequence mémorisee
int der_lue = 0;            // indice derniere frequence affichée
int mode_nav = 0;           // mode fonctionnement : 1= navigation sur station préréglées (défaut)  2= scan manuel
boolean mute = false;       // son actif ou coupé
int frestation[30];         // tableau des fréquences connues sur [Rennes]
String nomstation[30];      // nom des station des fréquences connues

// carte SD
int sd_select = 53;         // pin 53 (SS : sélection carte SD)  (SS = select slave)
int sd_detect = 6;          // pin 6 (detection présence carte SD)
boolean sd_present = false; // indicateur présence carte SD
boolean sd_lec1 = true;     // indicateur erreur lecture fichier noms stations
boolean sd_lec2 = true;     // indicateur erreur lecture fichier mémo stations
File sd_file;               // objet fichier
File fichier;               // objet fichier des stations connues sur [Rennes]
String readString = "";     // chaine de lecture fichier

// encodeur rotatif
const byte rot_porta  = 2;  // pin 2 + INPUT_PULLUP   Rotary Encoder right (Interruption 4)
const byte rot_portb  = 3;  // pin 3 + INPUT_PULLUP   Rotary Encoder left  (Interruption 5)
int rot_val = 0;            // valeur incrémentale de rotation
int rot_val_old = 0;        // mémo valeur incrémentale de rotation précédente
byte rot_etat = 0;          // data lue sur rotatif

// bouton RAZ
const byte raz = 18;        // pin 18 + INPUT_PULLUP    bouton raz frequences mémorisées (Interruption 3)
unsigned long timboupre,timbouact,timraz = 0; // pour éviter le rebond sur appui bouton raz
int dem_raz = 0;            // indic de demande de RAZ
int beep = 7;               // pin 7 alerte beep si RAZ

// boutons
#define rot_bouton     4    // pin 4 + INPUT_PULLUP   Push button de l'encodeur rotatif
#define nb_boutons     1    // nbre de boutons utilisés
#define antirebond     2    // durée antirebond
#define doubleclic    20    // durée double clic
#define appuilong     40    // durée appui long
byte bouton_val = 0;        // valeur retournee par le bouton
byte bouton_ret = 0;        // code retour de l'appui bouton 1=clic, 2 = double clic, 3=appui long
byte bouton_ret_old = 0;    // mémo code retour appui bouton précédent
PushButtonCommand poussebouton;

void RDS_process(uint16_t block1, uint16_t block2, uint16_t block3, uint16_t block4) {
  rds.processData(block1, block2, block3, block4);
}

void setup() {
  #if TRACES
    Serial.begin(57600);
  #endif
  radio.init();
  radio.debugEnable();   // Enable information to the Serial port
  u8g2.begin();
  pinMode(beep, OUTPUT);
  pinMode(sd_select, OUTPUT);
  pinMode(sd_detect,INPUT_PULLUP);
  pinMode(rot_porta, INPUT_PULLUP);
  pinMode(rot_portb, INPUT_PULLUP);
  pinMode(rot_bouton, INPUT_PULLUP);
  pinMode(raz,INPUT_PULLUP);
  poussebouton.setup(nb_boutons, antirebond, doubleclic, appuilong);
  attachInterrupt(digitalPinToInterrupt(rot_porta), Rotation, CHANGE);
```

```
  attachInterrupt(digitalPinToInterrupt(rot_portb), Rotation, CHANGE);
  attachInterrupt(digitalPinToInterrupt(raz), Dem_raz_freq, CHANGE);

  for (int i=0; i<10; i++) {  // initialiser tableau = 0
    freq_memo[i] = 0;
  }
  Charger_freq();         // charger frequences enregistrées sur la carte SD
  Lecfistations();        // charger les noms des stations locales
  rot_val = frequence;
  rot_val_old = rot_val;
}

void initialisation() {
  radio.setBandFrequency(RADIO_BAND_FM, freq_allu);
  radio.setMono(false);
  stereo = true;
  radio.setMute(false);
  radio.setSoftMute(true);
  radio.setBassBoost(true);
  Regvolume();
  radio.attachReceiveRDS(RDS_process);
  rds.attachServicenNameCallback(DisplayServiceName);
  Affichage();
  lastfretime = millis();   // pour changer freq demarrage si ecoute > 5 min
}

void loop() {
  Regvolume();
  Appui_Boutons();
  if (bouton_ret != 0) {
    switch(bouton_ret) {
      case 1:                           // simple clic => Mute (couper ou remettre le son)
        if (mute == false) {
          mute = true;
          radio.setMute(true);
        }
        else {
          mute = false;
          radio.setMute(false);
        }
        #if TRACES
          Serial.println("Clic court");
        #endif
      break;
      case 2:                           // double clic
        if (mute == false) {                // opérationnel si en mode écoute seulement
          switch(mode_nav) {                // bascule du mode de navigation
            case 1:
              mode_nav = 2;                 // mise en mode manuel rapide (scan) ...
              rot_val = frequence;            // ... à partir de la fréquence actuelle
            break;
            case 2:
              mode_nav = 3;                 // mise en mode manuel lent ...
              rot_val = frequence;            // ... à partir de la fréquence actuelle
            break;
            case 3:
              if (nb_memo > 0) {              // si fréquence mémorisée
                mode_nav = 1;                 // mise en mode auto ....
                frequence = freq_memo[der_lue];    // ...  sur la derniere frequence utilisée
                initialisation();
                SetFreq();
              } else mode_nav = 2;     // on revient en mode auto rapide si on n'a pas mémorisé une station
            break;
```

```
      }
      #if TRACES
        Serial.println("Clic double");
      #endif
     }
     break;
   case 3:                                // clic long
     if (mute == false) {                 // opérationnel si en mode écoute seulement
      if (mode_nav != 1)  Memorise_frequence();  // ajouter à liste fréquences si en mode manuel
      else Efface_frequence();                   // pour effacer derniere station memorisée en mode auto
      #if TRACES
        Serial.println("Clic long");
      #endif
     }
   break;
  }
  bouton_ret = 0;
  Affichage();
}

  if (rot_val != rot_val_old) {
   if (mode_nav == 1) {
    if (rot_val > rot_val_old)  Cherche_suiv();
    else if (rot_val < rot_val_old)  Cherche_prec();
   }
   if (mode_nav == 2) {
    if (rot_val > rot_val_old)      Scan(0);
      else if (rot_val < rot_val_old)    Scan(1);
   }
      if (mode_nav == 3) {
    if (rot_val > rot_val_old) {
      frequence = frequence + 10;
      if (frequence > freq_max)   frequence = freq_min;
      SetFreq();
     }
     else if (rot_val < rot_val_old)  {
      frequence = frequence - 10;
      if (frequence < freq_min)   frequence = freq_max;
      SetFreq();;
     }
   }
   rot_val_old = rot_val;
   Affichage();
   delay(100);
  }

if (dem_raz > 0) {
  #if TRACES
   Serial.print("dem_raz : ");
   Serial.print(dem_raz);
   Serial.print("   timraz : ");
   Serial.print(timraz);
   Serial.print("   millis : ");
   Serial.println(millis());
  #endif
  if (dem_raz == 2) {                          // en attente confirmation
     if (millis() - timraz > 10000)  dem_raz = 0;    // annulation demeande raz au bout de 10 sec
  }
  else if (dem_raz == 1) {
        digitalWrite(beep,20);
        delay(50);
        digitalWrite(beep,0);
        dem_raz = 2;         // etat 2 pour attentre confirmation raz
```

```
        timraz = millis();
       }
       else if (dem_raz == 3) Raz_frequences();
   Affichage();
  }
  if (millis() - lastfretime > 300 000 && freq_allu != frequence) {    //  si ecoute > 5 min et freq differente
   #if TRACES
    Serial.print("   lastfretime : ");
    Serial.print(lastfretime);
    Serial.print("   millis : ");
    Serial.println(millis());
   #endif
   lastfretime = millis();   // on stocke les frequences dans la carte SD
   Enreg_frequences();       // la frequence écoutée deviendra la fréquence à l'allumage
   Affichage();

  }

  radio.checkRDS();     // check for RDS data
}

//==================Full graphic smart controller=======================
void Affichage() {
  if (mute)  Affiaidemuet();
  else Affinormal();
}
//-----------------------------------------
  void Affinormal() {
  u8g2.firstPage();          // debut affichage
  do {
// mode navigation
   u8g2.setFont(pol_5an);
   if (mode_nav == 1)   u8g2.drawStr(35,63,"Auto");
   else {
     u8g2.drawStr(35,56,"Manu");
     if (mode_nav == 2)  u8g2.drawStr(35,63,"scan");
     else u8g2.drawStr(35,63,"lent");
   }


    u8g2.setFont(pol_5an);
    if (stereo)   u8g2.drawStr(1,5,"STEREO");
    else u8g2.drawStr(2,5,"MONO");
    if (mute)  Affiaidemuet();
    else {
      if (freq_allu > 0)  {
        u8g2.drawStr(89,5,"Dem.");
        u8g2.setCursor(108, 5);
        float freq_aff = (freq_allu / 100);   // frequence réelle d'affichage
        u8g2.print(freq_aff,1);       // aff avec 1 decimale
      }
    }
    int j =0;
    for (int i = 0; i<30; i++)  {
      if (frestation[i] == frequence) {
        station = nomstation[i];
        j++;
      }
    }
    if (j==0)  station ="   ???   ";
    u8g2.setCursor(1, 18);
    u8g2.setFont(pol_9an );
    u8g2.print(station);
```

```
    if (dem_raz > 0)  message ="RAZ: RE-APPUI POUR CONFIRMER";
    else if (sd_lec1 == false)  message = "ERREUR fichier FrRennes.txt";
        else if (sd_lec2 == false)  message = "ERREUR fichier MemoSta.txt";
            else message = radio_text;
    u8g2.setCursor(1, 28);
    u8g2.setFont(pol_5an);
    u8g2.print(message);

// aff volume
    u8g2.setFont(pol_5an);
    u8g2.drawStr(1,54,"Vol.");
    u8g2.setCursor(1, 60);
    u8g2.print(volume);
    u8g2.drawLine(0,62, 30,62);
    u8g2.drawLine(30,62, 30,55);
    u8g2.drawLine(30,55, 0,62);
    u8g2.drawTriangle(0,62, 2*volume,62, 0+2*volume,62-(volume/2));

// aff frequence
    u8g2.drawHLine(0,40, 128);
    float valx = (frequence - freq_min) / ((freq_max - freq_min) / 128);
    int x = int(valx);
    u8g2.drawTriangle(x,41, x-4,48, x+4,48);
    u8g2.setFont(pol_5an);
    u8g2.drawStr(40,5,"Bande FM");
    u8g2.drawStr(111,63,"Mhz");
    u8g2.setFont(pol_16n);
    u8g2.setCursor(59, 64);
    float freq_aff = frequence;  // frequence réelle d'affichage
    freq_aff = (freq_aff / 100);
    u8g2.print(freq_aff,1);             // affichage avec 1 decimale


// aff frequence memorisee sur ligne frequences
    if (nb_memo > 0) {
      for (int i = 1; i <= nb_memo; i++) {
        float valx = (freq_memo[i] - freq_min) / ((freq_max - freq_min) / 128);
        int x = int(valx);
        u8g2.drawBox(x,38, 1, 5);
        u8g2.setFont(pol_5an);
        u8g2.setCursor(x-2,37);
        u8g2.print(i);
      }
    }
  } while( u8g2.nextPage() );      // fin affichage
}
//-----------------------------------------
void Affiaidemuet() {
  u8g2.clearDisplay();
  u8g2.firstPage();          // debut affichage de l'aide + muet
  do {
    u8g2.setFont(pol_9an );
    u8g2.drawStr(101,9,"Muet");
    u8g2.setFont(pol_5an);
    u8g2.drawStr(1,5,"AIDE");
    u8g2.drawStr(1,14,"Clic court : muet/ecoute");
    u8g2.drawStr(1,22,"Clic double : change le mode = ");
    u8g2.drawStr(5,28,"   auto, manuel, ou manuel lent");
    u8g2.drawStr(1,36,"Clic long :   si mode =");
    u8g2.drawStr(1,42,"          manuel  => memo station");
    u8g2.drawStr(1,48,"        auto  => raz dern.station");
```

```
    u8g2.drawStr(1,56,"RAZ : efface toutes stations");
    u8g2.drawStr(1,64,"Ecoute + 5min => Station demar.");

  } while( u8g2.nextPage() );      // fin affichage
}
//---------------------------------------
void Memorise_frequence() {   // clic long sur bouton en mode 2 pour memoriser frequence en cours
  if (frequence != 0) {
    if (nb_memo < 9)  nb_memo++;         // nb de frequence memorisees (maxi 9)
    der_memo++;                          // dernier poste de memorisation
    if (der_memo > 9)    der_memo = 1;    // bouclage sur poste de memorisation
    freq_memo[der_memo] = frequence;
    der_lue = der_memo;
    bouton_ret = 0;
    Enreg_frequences();
    #if TRACES
      Serial.println("Memorise_frequence");
    #endif
  }
}
//---------------------------------------
void Efface_frequence() {    // clic long sur bouton en mode 1 pour effacer dernière frequence mémorisée
  if (nb_memo > 0) {
    nb_memo--;
    if (der_memo == 0) der_memo = 9;
    else der_memo--;
    der_lue = der_memo;
    bouton_ret = 0;
    freq_allu = freq_memo[der_memo];
    frequence = freq_allu;       // restaurer frequence sur freq demarrage
    SetFreq();
    Enreg_frequences();
    #if TRACES
      Serial.println("Efface_frequence");
    #endif
  }
}
//---------------------------------------
void Cherche_suiv() {              // rotation droite pour chercher station memorisée suivante
  #if TRACES
    Serial.println("Cherche_suiv");
  #endif
  if (nb_memo > 0) {
    der_lue++;                           // dernier poste affiché
    if (der_lue > nb_memo)    der_lue = 1;    // bouclage sur poste de memorisation
    frequence = freq_memo[der_lue];
    SetFreq();
    delay(400);
  }
}
//---------------------------------------
void Cherche_prec() {              // rotation gauche pour chercher station memorisée précédente
  #if TRACES
    Serial.print("Cherche_prec : (nb_memo / der_lue) ");
    Serial.print(nb_memo);
    Serial.print(" / ");
    Serial.println(der_lue);
  #endif
  if (nb_memo > 0) {
    der_lue--;                           // dernier poste affiché
    if (der_lue == 0)    der_lue = nb_memo;    // bouclage sur poste de memorisation
    frequence = freq_memo[der_lue];
    SetFreq();
```

```
    delay(400);
  }
}
//----------------------------------------
void Appui_Boutons() {
  if(bouton_val == 0)   bouton_val = poussebouton.loop(1, !digitalRead(rot_bouton));
       switch(bouton_val) {
        case PUSH_CMD_CLICK:
          bouton_ret = 1;    // clic
          break;
        case PUSH_CMD_DOUBLE_CLICK:
          bouton_ret = 2;    // double clic
          break;
        case PUSH_CMD_LONG_CLICK:
          bouton_ret = 3;    // clic long
          break;
      }
      if (bouton_val != 0)  bouton_val = 0;
      delay(10);
}
//----------------------------------------
void Rotation() {
  if (mute == false) {                    // opérationnel si en mode écoute seulement
   rot_etat = rot_etat << 1 | digitalRead(rot_porta);
   rot_etat = rot_etat << 1 | digitalRead(rot_portb);
   byte testbin = (rot_etat | B11110000) - B11110000;   //test des 4 bits de poids faible
   if (testbin == B0111){
     rot_val++;
     #if TRACES
       Serial.println("rotation droite");
       Serial.println(rot_val);
     #endif
   } else if (testbin == B1011){
           if (rot_val > 0)  rot_val--;
           #if TRACES
             Serial.print("rotation gauche ");
             Serial.println(rot_val);
           #endif
         }
   delay(20);
  }
}
//----------------------------------------
void Charger_freq()  {    //   lire fréquences sauvegardées dans la carte SD
  #if TRACES
    Serial.println("Charger_freq");
  #endif
// enreg = 1(x5) nb frequences,   9(x5) frequences favorites, 1(x5) frequence demarrage => 55+1 car.
  if (digitalRead(sd_detect) == LOW) {
    sd_present = true;
    SD.begin(sd_select);
    sd_file=SD.open("MemoSta.txt",FILE_READ);
    if (!sd_file) {
      sd_lec2 = false;   //  échec ouverture fichier
    }
    else {
      while (sd_file.available()) {
        for (int i=0; i<11; i++) {
          readString = "";
          for (int j=0; j<5; j++) {
            char c = sd_file.read();
            readString += c;
          }
```

```
        #if TRACES
          Serial.print("i =  ");
          Serial.print(i);
          Serial.print("  ");
          Serial.println(readString);
        #endif
        if (i==0)  nb_memo = readString.toInt();
        else if (i<10)  freq_memo[i] = readString.toInt();
             else {
               freq_allu = readString.toInt();
             }
       }
      }
      der_memo = der_lue = nb_memo;
      #if TRACES
        Serial.println(" ");
        Serial.print("nb_memo  ");
        Serial.println(nb_memo);
        Serial.print("freq_allu  ");
        Serial.println(freq_allu);
        for (int i=1; i<10; i++) {
          Serial.print(" ");
          Serial.println(freq_memo[i]);
        }
      #endif
    }
    sd_file.close();
  }
  if ((sd_present) && (sd_lec2 == true) && (nb_memo > 0))  mode_nav = 1;   // mode auto
  else mode_nav = 2;                                                      // mode manuel
  if (freq_allu > 0) {
    frequence = freq_allu;          // frequence a l'allumage
    SetFreq();
  }
}
//----------------------------------------
void Enreg_frequences()  {
  #if TRACES
    Serial.println("enregt frequences preferees");
  #endif
  String str = "";
  if (sd_present) {
    sd_file=SD.open("MemoSta.txt",FILE_WRITE);
    if (!sd_file) {
      sd_lec2 = false;   //  échec ouverture fichier
      #if TRACES
        Serial.println("err.ecrit fichier");
      #endif
    }
    else {
      sd_file.seek(0);              // reecrire sur l'enregistrement
      str = zerogauch5(nb_memo);
      #if TRACES
        Serial.print("nb_memo  ");
        Serial.print(nb_memo);
        Serial.print("  ");
        Serial.println(str);
      #endif
      sd_file.print(str);
      for (int i=1; i<10; i++) {
        str = zerogauch5(freq_memo[i]);
        #if TRACES
          Serial.print("Freq  ");
```

```
          Serial.println(str);
         #endif
         sd_file.print(str);
       }
       str = zerogauch5(frequence);
       #if TRACES
        Serial.print("Freq demarr ");
        Serial.println(str);
       #endif
       sd_file.print(str);
       sd_file.close();
       freq_allu = frequence;        // pour actualiser l'affichage
     }
   }
}
//----------------------------------------
String zerogauch5(int nombre) {   // ajouter zéros non significatifs si nombre nnnnn < taille voulue
  String valnch = "";
  if (nombre < 10000) { valnch = "0"; }
  if (nombre < 1000)  { valnch = "00"; }
  if (nombre < 100)   { valnch = "000"; }
  if (nombre < 10)    { valnch = "0000"; }
  valnch += String(nombre, DEC);
  return valnch;      // renvoi le nombre en 4 chiffres quelle que soit sa valeur
}
//----------------------------------------
void Dem_raz_freq() {
  if (mute == false) {                    // opérationnel si en mode écoute seulement
    #if TRACES
      Serial.println("Demande RAZ");
    #endif
    timbouact = millis();
    if (timbouact - timboupre > 500) {    // temporisation appui sur bouton
      if (dem_raz == 0)  dem_raz = 1;     // demande raz activée
      if (dem_raz == 2)  dem_raz = 3;     // demande raz activée
      timboupre = timbouact ;
    }
  }
}
//----------------------------------------
void Raz_frequences() {
  #if TRACES
    Serial.println("RAZ frequences");
  #endif
  for (int i=0; i<10; i++)   freq_memo[i] = 0;
  nb_memo = 0;
  der_memo = 0;
  der_lue = 0;
  dem_raz = 0;
  frequence = freq_allu;
  mode_nav = 2;          // retour au mode manuel
  rot_val = frequence;
  SetFreq();
}
//================================RDA5807=======================================
void Scan(byte sens) {
  #if TRACES
    Serial.println("scan");
  #endif
  if (sens == 0)  radio.seekUp(true);
  else radio.seekDown(true);
  delay(150);
  frequence = radio.getFrequency();     // stockage pour decalage info
```

```
  #if TRACES
    Serial.print("Frequence == ");
    Serial.println(frequence);
  #endif
  radio_text = "";
 }
//--------------------------------------
void Regvolume() {
  int val = analogRead(A0);
  volume = map(val, 0, 1023, 0, 15);
  if (volume != vol_old) {
    radio.setVolume(volume);
    #if TRACES
      Serial.print("Change volume = ");
      Serial.println(volume);
    #endif
    vol_old = volume;
    Affichage();
  }
}
//=================================radio=======================================
void SetFreq() {     // selection de la frequence
  radio.setFrequency(frequence);
  radio_text = "";
  #if TRACES
    Serial.print("Set frequence a : ");
    Serial.println(frequence);
  #endif
}
//--------------------------------------
void DisplayServiceName(char *name) {  // function called by the RDS module when a new ServiceName is available.
  String text;
  String recu;
  #if TRACES
    Serial.print("texte : ");
    Serial.println(name);
  #endif
  if (dem_raz == 0)  {
    recu = name;
    recu.trim();
    radio_text = radio_text + recu + " ";
    #if TRACES
      Serial.print("recu : ");
      Serial.print(recu);
      Serial.print("    L = ");
      Serial.println(recu.length());
      Serial.println(radio_text);
    #endif
    if (radio_text.length() > 28) {
      #if TRACES
        Serial.print("long data : ");
        Serial.println(radio_text.length());
        Serial.print("data : ");
        Serial.println(radio_text);
      #endif
      text = radio_text.substring(radio_text.length() - 27);
      radio_text = text;
      #if TRACES
        Serial.println(radio_text);
      #endif
      }
    Affichage();
    }
```

```
}
//--------------------------------------
void Lecfistations() {    // charger les fréquence et les noms de stations à Rennes
  fichier=SD.open("FrRennes.txt",FILE_READ);
  if (!fichier) {   //  échec ouverture
    sd_lec1 = false;
    #if TRACES
      Serial.println("Echec ouverture lecture fichier FrRennes.txt");
    #endif
  }
  else {
    int j = 0;
    while (fichier.available()) {
      String fresta,nomsta = "";
       for (int i=0; i<23; i++) {   // enregts de 20 car (5 + 15)
         char c = fichier.read();
         if (i<5)  fresta +=c;
         if (i>4 && i<21) nomsta +=c;
       }
       frestation[j] = fresta.toInt();
       nomstation[j] = nomsta;
       #if TRACES
         Serial.print("Freq : ");
         Serial.print(fresta);
         Serial.print("    ");
         Serial.println(nomsta);
       #endif
       if (j < 30)   j = j+1;
     }
    fichier.close();
    sd_lec1 = true;        // lecture fichier OK
  }
}
//*****************************************************************************
```