```
// TouchScreen_Calibr_native for MCUFRIEND UNO Display Shields
// adapted by David Prentice
// for Adafruit's <TouchScreen.h> Resistive Touch Screen Library
// from Henning Karlsen's original UTouch_Calibration program.
// Many Thanks.

/* Typical pin layout used:
 * -----------------------------------------------------------------------------------------
 *           MFRC522      Arduino       Arduino   Arduino    Arduino          Arduino
 *           Reader/PCD   Uno/101       Mega      Nano v3    Leonardo/Micro   Pro Micro
 * Signal      Pin          Pin           Pin       Pin        Pin              Pin
 * -----------------------------------------------------------------------------------------
 * RST/Reset   RST         9             5         D9         RESET/ICSP-5     RST
 * SPI SS 1    SDA(SS)      ** custom, take a unused pin, only HIGH/LOW required **
 * SPI SS 2    SDA(SS)      ** custom, take a unused pin, only HIGH/LOW required **
 * SPI MOSI    MOSI         11 / ICSP-4   51        D11        ICSP-4           16
 * SPI MISO    MISO         12 / ICSP-1   50        D12        ICSP-1           14
 * SPI SCK     SCK          13 / ICSP-3   52        D13        ICSP-3           15
 *
 */

//utilise programme MEGA_ESP01V6.4 obligatoirement.

//V4 du 03/06/2019 - stable : Gestion des ecran, sliding et mode calculatrice
// La table Button_value n'est pas à jour
// voir comment faciliter le calibrage pour chaque ecran ( programme dédié ?)
//-> ajout de l'ESP via le port Serial2

// V5 - Abandonnée

//V6 - version intermédiaire

//V7 du 05/06/2019 - stable : Gestion des ecran, sliding et mode calculatrice
// correction des bugs de la gestion de l'ecran tactile
// minimisation des actions dans la boucle loop
// deplacement de l'init reseau dans le setup

//V8 du 07/06-
//Mise à jour des valeur de la table button_value
// A corriger : Tag non renseigné
// Etat : Ecran - stable
//        Mode debug ajouté
//        Choix main caché ajouté
//        Tag Mise libre configuré
// ESP01 : boucle init a tester
// MFRC522 : Init -spi.begin()?

//V8 du 08/06
// Etat : Ecran - stable
// ESP01 : boucle init a tester
// MFRC522 : Init -spi.begin()?
```

```
//V9 du 09/06
// Ecran : Stable
//ESPO1 : boucle init a tester
// MFRC22: integration du code multireader et detection carte enlevee

//V10 du 23/06
// Ecran : Stable
//ESPO1 : boucle init a tester - rajout d'une tempo a 20 secondes si ipaddress=="" (premiere connexion ou pb connexion)
//        test cmd=25? OK ( recuperation adresse ip de l'esp01)
//        test cmd=26? OK ( recuperation macadress de l'esp01)
//        rajout de Posid=reader dans les trames Tag=
//        verification des mots clé des trame : Tag, Data, Posid
//        recuperation dans les variable ipaddress et macaddress
//        Faut il rajouter Posid à la trame Tag=55 (tag enlevé)
//        Faut il eviter d'envoyer en double une meme trame ? oui pour economiser de la Band passante.
// MFRC22: stable


#define TOUCH_ORIENTATION  LANDSCAPE
#define TITLE "TouchScreen.h GFX Calibration"

#include <SPI.h>
//#include <Adafruit_GFX.h>
#include <TouchScreen.h>
#include <MCUFRIEND_kbv.h>
#include <TouchScreen.h>
#include <MFRC522.h>

MCUFRIEND_kbv tft;

const String SMARTPOKER_SERIE="SMP001";
const String SMARTPOKER_TYPE="SMP";
const String SMARTPOKER_VERSION="1.0";
const String SMARTPOKER_DATE="23/06/2019";

const int XP=6,XM=A1,YP=A2,YM=7; //240x400 ID=0x7793
const int TS_LEFT=372,TS_RT=476,TS_TOP=672,TS_BOT=761;


TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
TSPoint tp;
uint8_t Orientation = 1;    //PORTRAIT

#define MINPRESSURE 100
#define MAXPRESSURE 1000

// Assign human-readable names to some common 16-bit color values:
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
```

```
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

// la result box
#define TEXT_X 10
#define TEXT_Y 10
#define TEXT_W 220
#define TEXT_H 50
#define TEXT_TSIZE 3
#define TEXT_TCOLOR  MAGENTA
// the data (phone #) we store in the textfield
#define TEXT_LEN 12
char textfield[TEXT_LEN+1] = "";
uint8_t textfield_i=0;
String Stringfield="";
String Stringcmd="";

String  button_label [3][4][5] =
{
  {
    { "", "", "", "SND","CLR" },
    { "Fold", "Check", "Call", "Alin","" },
    { "*2", "*2.5", "*3", "*4","" },
    { "1/3P", "1/2P", "2/3P", "Pot","->" }
  },
  {
    { "", "", "", "SND","CLR" },
    { "0", "1", "2", "3","4" },
    { "5", "6", "7", "8","9" },
    { "00", "000", "0000", "00000","->" }
  },
  {
    { "", "", "", "SND","CLR" },
    { "Deal", "SB", "BB", "","Cache" },
    { "abs", "Mort", "Out", "Pause","" },
    { "Niv", "Stak", "Ver", "Debug","->" }
  }
};

int  button_value[3][4][5] =
{
  {
    { 0, 0, 0, 0,0 },
    { 33, 117, 116, 32,0 },
    { 106, 107, 109, 110,0 },
    { 113, 114, 115, 111,0 }
  },
   {
    { 0, 0, 0, 0,0 },
```

```
    { 0, 1, 2, 3,4 },
    { 5, 6, 7, 8,9 },
    { 100, 1000, 10000, 100000,0 }
  }
  ,
   {
    { 0, 0, 0, 0,0 },
    { 17, 43, 42, 0,27 },
    { 26, 53, 54, 24,0 },
    { 25, 118, 0, 0,0 }
  }
};

uint16_t buttoncolors[3][4][5] =
{
  {
    {  YELLOW,  YELLOW, YELLOW,  RED,  GREEN, },
    {  BLUE,  BLUE,  BLUE,   BLUE,  BLUE,},
    {   BLUE,  BLUE, BLUE, BLUE,  BLUE,},
    { BLUE,  BLUE,  BLUE, BLUE,  BLUE, }
  },
  {
    {   RED,  YELLOW, YELLOW,  RED,  GREEN, },
    {  BLUE,  BLUE,  BLUE,   BLUE,  BLUE,},
    {   BLUE,  BLUE, BLUE, BLUE,  BLUE,},
    { BLUE,  BLUE,  BLUE, BLUE,  BLUE, }
  }
  ,
  {
    {  BLUE,  YELLOW, YELLOW,  RED,  GREEN, },
    {  BLUE,  BLUE,  BLUE,   BLUE,  BLUE,},
    {   BLUE,  BLUE, BLUE, BLUE,  BLUE,},
    { BLUE,  BLUE,  BLUE, BLUE,  BLUE, }
  }

};

int page_max=2 ; // a modifier en fonction en fonction du nombre de page, compter à partir de zero
                 //si page_max=1, alors j'ai 2 pages dispo 0 et 1

int key_pressed=-1;
int oldkey=-1;
int cpt_unpressed=0;
int cpt_unpressed_max=500; // define in vivo
int oldcol=4; //send
int oldligne=4; //send
unsigned long user_value=0;
unsigned long multiple=1;
int pixel_x,pixel_y,col,ligne;
int ligne_temp=0;

//float px, py;
```

```
int change_page=0;
int change_niveau=0;
int page=0;
int b=-1;

/***Gestion des echanges avec l'arduino **/
String cmdtoarduino="";
String cmdfromarduino="";
String datafromarduino="";
//String tramefromEsp01="";
//String cmdfromEsp01="";
String datafromEsp01="";
String inputstring="";
char   inChar;
String inputstringRX="";
char   inCharRX;
bool   stringComplete=false;
bool   stringCompleteRX=false;
unsigned long cpt=0;
unsigned long maxcpt=1000000;
bool   initwifi=false;
bool   iprecu=false;
String ipaddress="";
String macaddress="";
bool   mode_debug=false;
bool   demande_debug=false;

#define RST_PIN         47          // Configurable, see typical pin layout above
#define SS_1_PIN        43         // Configurable, take a unused pin, only HIGH/LOW required, must
be diffrent to SS 2
#define SS_2_PIN        41        // Configurable, take a unused pin, only HIGH/LOW required, must
be diffrent to SS 1

#define NR_OF_READERS   2

byte ssPins[] = {SS_1_PIN, SS_2_PIN};

MFRC522 mfrc522[NR_OF_READERS];   // Create MFRC522 instance.
//String datafromarduino="";
//String cmdfromarduino="";
String tag="";
unsigned long previousMillis=0 ;
unsigned long interval = 5000; ///lecture de la carte toutes les 5 secondes
//uint8_t max_interation=3;
uint8_t reader;

unsigned long oldcpt_rfid[NR_OF_READERS];
unsigned long cpt_rfid[NR_OF_READERS];
String tag2_rfid[NR_OF_READERS];
String tag_rfid="";
```

```
void setup()
{
   Serial.begin(9600);
   Serial2.begin(9600); //2nd port serie de l'ardduino MEGa pour echanger avec l'ESP01
   Serial.println(TITLE);
    SPI.begin();        // Init SPI bus pour MFRC522
   tft.reset();
   uint16_t ID = tft.readID();
   Serial.print("ID = 0x");
   Serial.println(0x7793, HEX);
   tft.reset();
   tft.begin(ID);
   tft.setRotation(0);  //PORTRAIT
   affiche_version();
   delay(15000);
   //page=0;
   //clavier0b();
   //tft.setRotation(1);  //PAYSAGE
   /*** Procédure de test du touchpoint  -- a conserver pour debug
   tft.setRotation(1);
   tft.fillScreen(BLACK);
     while (1) {
       tp = ts.getPoint();
       pinMode(XM, OUTPUT);
       pinMode(YP, OUTPUT);
       if (tp.z < MINPRESSURE || tp.z > MAXPRESSURE) continue;
       tft.setCursor(0, (tft.height() * 3) / 4);
        tft.fillScreen(BLACK);
       tft.print("tp.x=" + String(tp.x) + " tp.y=" + String(tp.y) + + " tp.z=" + String(tp.z));
   }
   */



}

void loop()
{
   Touch_getXY();
   lire_mfrc522();
   //data_from_esp01();
   /*RFID - Ne pas deplacer ***/
 /*

  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++)
   {
   // Look for new cards

   if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial())
   {
     Serial.print(F("Reader "));
     Serial.print(reader);
```

```
      // Show some details of the PICC (that is: the tag/card)
      Serial.print(F(": Card UID:"));
      dump_byte_array(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size);
      Serial.println();
      Serial.print(F("PICC type: "));
      MFRC522::PICC_Type piccType = mfrc522[reader].PICC_GetType(mfrc522[reader].uid.sak);
      Serial.println(mfrc522[reader].PICC_GetTypeName(piccType));

      // Halt PICC
      mfrc522[reader].PICC_HaltA();
      // Stop encryption on PCD
      mfrc522[reader].PCD_StopCrypto1();
    } //if (mfrc522[reader].PICC_IsNewC
  } //for(uint8_t reader
*/
}


/*************** Fonction ecran  **************/
int affiche_version()
{

 tft.setRotation(1);
 tft.fillScreen(BLACK);
 tft.setTextColor(WHITE);
 tft.setTextSize(1);
 tft.setCursor(1,2);
 tft.print("SMARTPOKER_SERIE   : ");
 tft.println(SMARTPOKER_SERIE);
 tft.print("SMARTPOKER_TYPE    : ");
 tft.println(SMARTPOKER_TYPE);
 tft.print("SMARTPOKER_VERSION : ");
 tft.println(SMARTPOKER_VERSION);
 tft.print("SMARTPOKER_DATE    : ");
 tft.println(SMARTPOKER_DATE);
 tft.println("-----------------------------------");
 tft.println("WIFI");
 if (ipaddress=="")
   {
     //premiere connection, attendre 20 secondes
     Serial.println("Please Wait 20 secondes");
     tft.println("Attendez 20 secondes pour l'init Wifi");
     delay(5000);
    }
 init_connexion_esp01();
 tft.print("Ip addresse : ");
 tft.println(ipaddress);
 tft.print("Mac address : ");
 tft.println(macaddress);
 tft.println("Point d'acces :routeur_smartpoker");
 tft.println("-----------------------------------");
 tft.println(" RFID");
```

```
 for (reader = 0; reader < NR_OF_READERS; reader++)
 {
   mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN); // Init each MFRC522 card
   tft.println(mfrc522[reader].PCD_PerformSelfTest() ? "ok" : "ERROR!");
   mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);
   delay(200);
  }

delay(10000);

 // changement de page
  tft.setRotation(1);
  tft.fillScreen(BLACK);
  tft.setTextColor(WHITE);
  tft.setTextSize(1);
  tft.setCursor(1,2);
  page=0;
  draw_buttons(page);
}


/************************** fonctions ************************/

void draw_buttons(int page_actuelle)
{
     //Draw the Result Box
   tft.fillRect(0, 0, 240, 60, buttoncolors[page_actuelle][0][0]); // ori_x, ori_y, longueur,largeur
   tft.fillRect  (240,0,80,60,buttoncolors[page_actuelle][0][3]);
   tft.fillRect  (320,0,80,60,buttoncolors[page_actuelle][0][4]);
   // Ligne 1
   for (int i=0;i<5;i++) // j'ai 5 cases par lignes
    {
       for (int y=1;y<4;y++) //j'ai 3 lignes sous la result box
         {
           tft.fillRect(i*80,y*60,80,y*60,buttoncolors[page_actuelle][y][i]);
         }
    }
   //Draw Horizontal Lines
  for (int h=60; h<=240; h+=60)
    {
      tft.drawFastHLine(0, h, 400, WHITE);
    }

  //Draw Vertical Lines
  for (int v=0; v<=400; v+=80)
  {
    tft.drawFastVLine(v, 60, 240, WHITE);
  }

//Display keypad lables
  for ( int j=0;j<4;j++) {
    for ( int i=0;i<5;i++) {
```

```
      tft.setCursor(10 + (80*i), 20+(60*j));
      tft.setTextSize(2);
      tft.setTextColor(WHITE);
      tft.println( button_label[page_actuelle][j][i]);
     }
   }

    user_value=0; //reinit valeur user_value a chaque changement de page;


}

void invert_pressed_button(int page_actuelle,int xold2,int yold2,int x2,int y2)
{
    /*
    Serial.print ("page_actuelle");
    Serial.println(page_actuelle);
    Serial.print("old=");
    Serial.print(xold2);
    Serial.print(",");
    Serial.print(yold2);
    Serial.print("NEW=");
    Serial.print(x2);
    Serial.print(",");
    Serial.print(y2);
    */
    tft.fillRect(xold2*80,yold2*60,80,60,buttoncolors[page_actuelle][yold2][xold2]);
    tft.setCursor(10 + (80*xold2), 20+(60*yold2));
    tft.setTextSize(2);
    tft.setTextColor(WHITE);
    tft.println( button_label[page_actuelle][yold2][xold2]);


   tft.fillRect(x2*80,y2*60,80,60,RED);
   tft.setCursor(10 + (80*x2), 20+(60*y2));
   tft.setTextSize(2);
   tft.setTextColor(WHITE);
   tft.println( button_label[page_actuelle][y2][x2]);

   //redraw le quadrillage
//Draw Horizontal Lines
   for (int h=60; h<=240; h+=60)
     {
       tft.drawFastHLine(0, h, 400, WHITE);
     }

   //Draw Vertical Lines
   for (int v=0; v<=400; v+=80)
     {
      tft.drawFastVLine(v, 60, 240, WHITE);
     }//end for
```

```
  //Draw the Result Box
  tft.fillRect(0, 0, 240, 60, buttoncolors[page_actuelle][0][0]); // ori_x, ori_y, longueur,largeur

}

void affiche_texte(int mode,int ligne2, int col2)
{
      if (mode==0) //affiche une touche
      {
        tft.fillRect(0, 0, 240, 60, buttoncolors[page][0][0]); // ori_x, ori_y, longueur,largeur
        tft.setCursor(10 ,22 );
        tft.setTextSize(2);
        tft.setTextColor(BLACK);
        tft.print(button_label[page][col2][ligne2]);
        user_value=button_value[page][col2][ligne2];
      }
      if (mode==1) //mode calculatrice
      {
       tft.fillRect(0, 0, 240, 60, buttoncolors[page][0][0]); // ori_x, ori_y, longueur,largeur
        tft.setCursor(10 ,22 );
        tft.setTextSize(2);
        tft.setTextColor(BLACK);
        //tft.print(itoa(user_value)); //calculer dans la fonction onclick
        tft.print(user_value); //calculer dans la fonction onclick
      }
}


bool Touch_getXY(void)
{
   tp = ts.getPoint();
   pinMode(XM, OUTPUT);
   pinMode(YP, OUTPUT);
   digitalWrite(YP, HIGH);   //because TFT control pins
   digitalWrite(XM, HIGH);
   if (tp.z < MINPRESSURE)
   {
   //gestion du toggle
   cpt_unpressed++;
   // fin du toggle
   }
   else
   {

     //identification de la touche
      //redressement des coordonnées

     /* pour debuggage **********************/
      //Serial.print("tp.z=");
      //Serial.print(tp.z);
      //Serial.print(" tp.x=");
      //Serial.print(tp.x);
```

```
 //Serial.print("  tp.y=");
 //Serial.println(tp.y);
/* ************************************/
 if (tp.x<160)  tp.x=160;
 if (tp.x >480) tp.x=480;

 if (tp.y<180)  tp.y=180;
 if (tp.y >900) tp.y=900;

 ligne_temp=0;
 col=(tp.y-180)/50; //5 colonnes donc /50
 ligne_temp=((480-tp.x)/24);  //
 switch (col)
 {
   case 0 ... 2 : col=0;
                  switch (ligne_temp)
                  {
                   case 0 ... 3 : ligne=0;break;
                   case 5 ... 7 : ligne=1;break;
                   case 9 ... 11 : ligne=2;break;
                   case 12 ... 18 : ligne=3;break;
                  }
                  break;
   case 3 ... 5 : col=1;
                   switch (ligne_temp)
                  {
                   case 0 ... 5 : ligne=0;break;
                   case 6 ... 8 : ligne=1;break;
                   case 10 ... 11 : ligne=2;break;
                   case 12 ... 18 : ligne=3;break;
                  }
                  break;
   case 6 ... 8 : col=2;
                   switch (ligne_temp)
                  {
                   case 0 ... 4 : ligne=0;break;
                   case 5 ... 7 : ligne=1;break;
                   case 9 ... 11 : ligne=2;break;
                   case 12 ... 18 : ligne=3;break;
                  }
                  break;
   case 9 ... 11 :col=3;
                   switch (ligne_temp)
                  {
                   case 0 ... 1 : ligne=0;break;
                   case 3 ... 5 : ligne=1;break;
                   case 8 ... 10 : ligne=2;break;
                   case 12 ... 18 : ligne=3;break;
                  }
                  break;
   case 12 ... 14 :col=4;
                  switch (ligne_temp)
```

```
                {
                 case 0 ... 1 : ligne=0;break;
                 case 2 ... 5 : ligne=1;break;
                 case 7 ... 9 : ligne=2;break;
                 case 10 ... 18 : ligne=3;break;
                }
                break;
  }
 ligne_temp=0;
 key_pressed= button_value[page][col][ligne];
 /* pour debuggage **********************/
 //Serial.print ("keypressed : page");
 //Serial.println(page);
 //Serial.print ("keypressed : col");
 //Serial.println(col);
 //Serial.print ("keypressed : ligne");
 //Serial.println(ligne);
 //Serial.print ("old keypressed : ");
 //Serial.println(oldkey);
 //Serial.print ("keypressed : ");
 //Serial.println(key_pressed);
/* ************************************/

 if (oldkey!=key_pressed)
  {

   //Serial.print("oldkey");
   //Serial.println(oldkey);
   //Serial.print ("Key value : ");
   //Serial.println(key_pressed);
   oldkey=key_pressed;

   invert_pressed_button(page,oldcol,oldligne,col,ligne);
   oldcol=col;
   oldligne=ligne;

   switch (page)
     {
    case 0 : affiche_texte(0,col,ligne);break;
    case 1 : affiche_texte(1,col,ligne);break;
    case 2 : affiche_texte(0,col,ligne);break;
     }
   // gestion des touches
   onclick();
   //Serial.println("sortie onclick");
   // ajout du 05/06/2019
   ligne=0;
   col=0;
   }
 } //End else
if (tp.z<100)
  {
```

```
        key_pressed=-1;
        cpt_unpressed++;
      }
      else
      {
      cpt_unpressed=0;
      }
      if (cpt_unpressed>=cpt_unpressed_max)
        {
         oldkey=-1;
         //Serial.println("reinit Oldkey");
        }

}// end function


void onclick()
{
  int z;
    Serial.println("devut onclick");
   Serial.println(cmdfromarduino);
  //traitement de touches SND et CLR
  if (ligne==0 && col==3) // touche SND
     {
        Serial.println("SND");
        if (page==1)
            {
            datafromarduino=String(user_value,DEC);       //transforme uservalue en string
            if (cmdfromarduino="")
                {
                 cmdfromarduino="Tag=55"; // tag pour stack
                }
            if (datafromarduino!="")
               {
                  cmdfromarduino=cmdfromarduino+"?Data="+datafromarduino;
                } //end if   datafromarduino!=""
            }
        if (page==0 || page==2)
            {
              cmdfromarduino="Tag="+ cmdfromarduino;
            }
        Serial.println("Mode debug?");
        Serial.print("debug ?:demande debug");
        Serial.println(demande_debug);
        Serial.print("debug ? : cmdfromarduino");
        Serial.println(cmdfromarduino);
        Serial.print("debug ? : datafromarduino");
        Serial.println(datafromarduino);
        if (demande_debug==false)
         {
           Serial.println("demande_debug=Vrai");
           if (datafromarduino=="895")
```

```
          {
           mode_debug=true;
           cmdfromarduino="mode debug :";
           goto suite;
          }
         }
        if(mode_debug==true)
          {
             tft.fillRect(0, 0, 400, 10, BLACK); // ori_x, ori_y, longueur,largeur
             tft.setCursor(0,0);
             tft.setTextSize(1);
             tft.setTextColor(WHITE);
             tft.println(cmdfromarduino);
          }// end mode_debug==true
        Serial.print("cmdfromarduino :");
        Serial.println(cmdfromarduino);
        Serial2.println(cmdfromarduino);
        //efface ecran


         //draw_buttons(page);
         cmdfromarduino="";
         datafromarduino="";
         user_value=0;

       //Serial.println("SND=(ligne==0 && col==3)");
       goto suite;
   } //end if touche SND

if (ligne==0 && col==4) // touche CLR
   {
      Serial.println("CLR");
      //Draw the Result Box
       tft.fillRect(0, 0, 240, 60, buttoncolors[page][0][0]); // ori_x, ori_y, longueur,largeur
       user_value=0;
       Serial.println("(ligne==0 && col==4)");
       // réinit des variables
       if (page==0 || page==2)
         {
           cmdfromarduino="";
         }
       else
         {
            datafromarduino="";
         }
      goto suite;
   } //end if touche CLR

if (ligne==3 && col==4) // touche ->
   {
     Serial.println("->");
     if (page==1)
```

```
      {
       //si je sors de la page 1 apres y etre arrivé par Stak ou niv, il faut que je reinitialise la variable
        cmdfromarduino="";
        datafromarduino="";
      }
    if (page==page_max)
      {
       page=0;
       Serial.println("Page_max atteint -> page=0");
      }
    else
     {
       page++;
       Serial.print("page=page_max? ");
       Serial.print(page);
       Serial.print(" ? ");
       Serial.println(page_max);
     }
    draw_buttons(page);
    col=0;
    ligne=0;
    oldcol=0;
    oldligne=0;
    goto suite;
  }  //end if touche ->

switch (page)
{
  case 0 :
          cmdfromarduino=button_value[page][ligne][col];
          Serial.println("page 0 - cmdarduino : ");
          Serial.println(cmdfromarduino);
          break; //page 0
  case 1 :
           Serial.println("je suis page 1");
           if(ligne==1 || ligne==2)
           {
            Serial.println("page1 - Ligne 1 ou 2");
            Serial.println(button_value[page][ligne][col]);
            // ne pas depasser 9 999 999 millions
            if ((user_value*10+button_value[page][ligne][col] )<10000000)
             {
              user_value=user_value*10+button_value[page][ligne][col];
             }
            Serial.println(user_value);
            affiche_texte(1,0,0) ;
           }//end if ligne==1 ou  2

           if(ligne==3) //ligne multiplicatrice *100, *1000, * 10 000
           {
            Serial.println("page1 - Ligne 3");
            Serial.println(button_value[page][ligne][col]);
```

```
          if ((user_value*button_value[page][ligne][col])<10000000)
          {
            user_value=user_value*button_value[page][ligne][col];
          }
          Serial.println(user_value);
          affiche_texte(1,0,0) ;
         }//end if ligne==1 ou  2
         break; //page1;
    case 2 :
          Serial.println("swith page, debut case 2");
          Serial.print("page2 - ligne :");
          Serial.println (ligne);
          Serial.print("page2 - colonne :");
          Serial.println (col);
          Serial.print("page2 - cmdfromardion :");
          Serial.println (cmdfromarduino);
         if (ligne==3 && col==0) // Niv
          {
            //Serial.println("page 2 et (ligne==3 && col==0)");
            cmdfromarduino=button_value[page][ligne][col];
            page=1;
            draw_buttons(page);
            goto suite;
          };
          if (ligne==3 && col==1) // Stak
          {
            //Serial.println("page 2 et ligne==3 && col==1");
            cmdfromarduino=button_value[page][ligne][col];
            page=1;
            draw_buttons(page);
            goto suite;
          };
          if (ligne==3 && col==2) // version
          {
            //Serial.println("page 1 et (ligne==3 && col==2)");
            affiche_version();
            //--
            delay(100);
            draw_buttons(page);
          };
          if (ligne==3 && col==3) // debug
          {
            //Serial.println("page 1 et (ligne==3 && col==2)");
            if (mode_debug==true) //sortie du mode debug en appuyant a nouveau sur la touche
Debug
              {
                mode_debug=false;
              }
            else
              {
                demande_debug==true;
              }
```

```
              cmdfromarduino="";
              datafromarduino="";
              page=1;
              draw_buttons(page);
              goto suite;
             };
             // valeur de cmdfromarduino pour toutes les autres touches
             cmdfromarduino=button_value[page][ligne][col];
             Serial.println("swith page, fin case 2");
             Serial.print("page2 - ligne :");
             Serial.println (ligne);
             Serial.print("page2 - colonne :");
             Serial.println (col);
             Serial.print("page2 - cmdfromardion :");
             Serial.println (cmdfromarduino);
            break;//page 2
      default:break;
   }

suite:
    z=0;
     Serial.println(cmdfromarduino);
} //end onclick


//************ gestion des echanges avec l'ESP01 **********************
/* j'ai plusieurs choix d'architecture :
 *  soit je suis lineaire c'est l'arduino qui pilote l'ESP01 (envoi de commande et data), en retour j'ai
des acquittements
 *  Soit je laisse une partie des commandes cotés ESP01 et j'envoie des commandes réduites
 *  cmd=xx?yyyyyyy (yyy etant limité à 32 caracteres
 *  le Watchdog est positionné sur l'ESP01
 *    Commandes recues de l'ESP01
 *

 *
*/

void init_connexion_esp01()
{
/***boucle de connexion au reseau
     *
     */
    initwifi=false;
    iprecu==false;
    //demande adresse ip
    Serial2.println("cmd=25?");
    Serial.println("cmd=25? - demande de l'adresse mac envoyé à l'ESP01");

    //iprecu=true; // pour debug
    while(stringComplete==false)
     {
```

```
     data_from_esp01();
     }
    stringComplete=false;
    //stockage de la valeur
    macaddress=datafromEsp01;
    datafromEsp01="";

    //demande adresse mac
    Serial2.println("cmd=26?");
    Serial.println("cmd=26?- demande de l'adresse ip envoyé à l'ESP01");
    while(stringComplete==false)
     {
      data_from_esp01();
     }
     stringComplete=false;
     ipaddress=datafromEsp01;
    datafromEsp01="";
}


void data_from_esp01()
{
// lecture des ports series
int cmd=0;
int delai_from_esp01=0;

while (Serial2.available()) //les données sont recus de l'ESP01
    {
      // get the new byte:
     inChar = (char)Serial2.read();
     Serial.println(inChar);
   // add it to the inputString:
      inputstring += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
      if (inChar == '\n')
      {
       stringComplete = true;
       int nbcar=inputstring.length()-1;
       datafromEsp01=inputstring.substring(0,nbcar);
       Serial.print(" Debug data_from_esp01: ");
       Serial.println(datafromEsp01);
      }//end if Inchar
     } //endwhile Serial2.available

    //reinit des variables
suite_data_from_esp01 :
     inputstring="";

}
```

```
/***** RFID ******  /
**
 * Helper routine to dump a byte array as hex values to Serial.
 */

void lire_mfrc522()
{
if( millis() - previousMillis >= interval)
 {
  Serial.println("debug : Lecture rFID");
  previousMillis = millis();
  for (reader = 0; reader < NR_OF_READERS; reader++)
  {
   cmdfromarduino="";
   datafromarduino="";
   mfrc522[reader].PCD_Init();
    // Look for new cards
    if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial())
    {
      cpt_rfid[reader]++;
      datafromarduino="?Data="+String(reader);
      //Serial.print("Dump 1 : datadfromarduino");
      //Serial.println(datafromarduino);
      cmdfromarduino=getCode(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size);
      cmdfromarduino.toUpperCase();
      cmdfromarduino.trim();
      tag=cmdfromarduino;
      cmdfromarduino="Tag="+cmdfromarduino+datafromarduino+"?Posid="+reader;

      tag2_rfid[reader]=tag;


      Serial.print("Dump 2 : Lecture RFID ");
      Serial.print(reader);
      Serial.print(" - ");
      Serial.println(cmdfromarduino);
      // envoi vers ESP01 si different


      //Serial.print("Dump 4 : tag - ");
      //Serial.println(tag);
     // Halt PICC
     mfrc522[reader].PICC_HaltA();
     // Stop encryption on PCD
     mfrc522[reader].PCD_StopCrypto1();
   } //end if
  } //for( reader=0....

        //Serial.println("Dump : reader 0 - Mise à jour previousMillis");
        //Serial.print("debug cpt0 - ");
        //Serial.println(cpt[0]);
        //Serial.print("debug oldcpt0 - ");
```

```
        //Serial.println(oldcpt[0]);
        //Serial.print("debug tag2[reader] - ");
        //Serial.println(tag2[0]);
        if (oldcpt_rfid[0]!=cpt_rfid[0] && tag2_rfid[0]=="")
        {
              //Serial.print("Lecteur ");
              //Serial.println(reader);
              Serial.println(" Lecteur 0 - carte enlevée");
              oldcpt_rfid[0]=0;
              cpt_rfid[0]=0;
         }
         tag_rfid[0]=""; tag2_rfid[0]="";
        //Serial.println("----------------");
        // Serial.println("Dump : reader 1 - Mise à jour previousMillis");
        //Serial.print("debug cpt1 - ");
        //Serial.println(cpt[1]);
        //Serial.print("debug oldcpt1 - ");
        //Serial.println(oldcpt[1]);
        //Serial.print("debug tag2[reader} - ");
        //Serial.println(tag2[1]);
        if (oldcpt_rfid[1]!=cpt_rfid[1] && tag2_rfid[1]=="")
        {
              //Serial.print("Lecteur ");
              //Serial.println(reader);

              Serial.println(" Lecteur 1 - carte enlevée");
              oldcpt_rfid[1]=0;
              cpt_rfid[1]=0;
         }
         tag_rfid[1]=""; tag2_rfid[1]="";
       //Serial.println("**********************");
 } //end if millis

} //end lire MFRC522

 /**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void dump_byte_array(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.println(buffer[i], HEX);
  }
}

String getCode(byte *buffer, byte bufferSize) {
  String code = "";
  for (byte i = 0; i < bufferSize; i++) code = code + String(buffer[i], HEX);
  return (code);

}
```