// 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.

// programme : smartpoker-MEGA-V6.2_TFT4.ino

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

//#include <Adafruit_GFX.h>
#include <TouchScreen.h>
#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;
#include <TouchScreen.h>

const String SMARTPOKER_SERIE=" SMP001";
const String SMARTPOKER_TYPE="SMP";
const String SMARTPOKER_VERSION=" 1.0";
const String SMARTPOKER_DATE="06/12/2018";

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", "","" },
    { "abs", "Mort", "Out", "Pause","" },
    { "Niv", "Stak", "", "","->" }
  }
};

int  button_value[3][4][5] =
{
  {
    { 120, 121, 122, 123,124 },
    { 125, 126, 127, 128,129 },
    { 130, 131, 132, 133,134 },
    { 115, 116, 117, 118,119 }
  },
   {
    { -1, -2, -3, -4,-5 },
    { 0, 1, 2, 3,4 },
    { 5, 6, 7, 8,9 },
    { 100, 1000, 10000, 10000,-6 }
  }
  ,
   {
    { 200, 201, 202, 203,204 },
    { 205, 206, 207,208,209 },
    { 210, 211, 212, 213,214 },
    { 215, 216, 217, 218,219 }
  }
};

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=20000; // define in vivo
int oldcol=0; //send
int oldligne=0; //send
unsigned long user_value=0;
unsigned long multiple=1;
int pixel_x,pixel_y,col,ligne;
int col_temp=0;
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 inputstring="";
char inChar;
String inputstringRX="";
char inCharRX;
bool stringComplete=false;
bool stringCompleteRX=false;
unsigned long cpt=0;
unsigned long maxcpt=1000000;
bool initwifi=true;
bool iprecu=false;


void setup()
{
   Serial.begin(9600);
   Serial.println(TITLE);
   tft.reset();
   uint16_t ID = tft.readID();
   Serial.print("ID = 0x");
   Serial.println(0x7793, HEX);
   tft.reset();
   tft.begin(ID);
   tft.setRotation(0);  //PORTRAIT
   //clavier0a();
   delay(2000);
   clavier0b();
   page=0;
   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();



    /**********************cas page 2 ********************
 if (change_page==1)
   {
      if (b == 0) {

        cmdfromarduino="Tag=";
        cmdfromarduino+=Stringfield;
        Serial.print("cmdfromarduino :");
        Serial.println(cmdfromarduino);
        Serial2.println(cmdfromarduino);
      }
      // clr button! delete char
      if (b == 1) {

        textfield[textfield_i] = 0;
        if (textfield > 0) {
          textfield_i--;
          textfield[textfield_i] = ' ';
        }
```

```
    }

    // its always OK to just hang up
    if (b == 2) {



    }

   // if a numberpad button, append the relevant # to the textfield
    if (b >= 3) {

        //saisie Niveau
       if (b == 12) {
         change_niveau=1;
         goto suite;
       }

       //changement de page
       if (b == 14) {
         change_page=1;
         goto suite ;
       }

       if (textfield_i < TEXT_LEN)
       {
         //textfield[textfield_i] = button_label[page][b][0]; //
         textfield_i++;
         textfield[textfield_i] = 0; // zero terminate
       }
    } //if (b >= 3)
suite:

    /* update the current text field
    Serial.println(textfield);
    tft.setCursor(TEXT_X + 2, TEXT_Y+10);
    tft.setTextColor(TEXT_TCOLOR, BLACK);
    tft.setTextSize(TEXT_TSIZE);
```

```
        tft.print(textfield);

    } //page==1 */

      //delay(100); // UI debouncing

}


/************** Fonction ecran  **************/
int clavier0a()
{

 tft.setRotation(1);
 tft.fillScreen(BLACK);
 tft.setTextColor(WHITE);
 tft.setTextSize(1);
 tft.setCursor(1,2);
 tft.println("SMARTPOKER_SERIE:");
 tft.println(SMARTPOKER_SERIE);
 tft.println("SMARTPOKER_TYPE:");
 tft.println(SMARTPOKER_TYPE);
 tft.println("SMARTPOKER_VERSION:");
 tft.println(SMARTPOKER_VERSION);
 tft.println("SMARTPOKER_DATE:");
 tft.println(SMARTPOKER_DATE);
 tft.println("");
 tft.setTextSize(2);
 tft.println("Visitez le site");
 tft.setTextSize(1);
 tft.println("");
 tft.println("smartpoker.fr");
 delay(5000);
}

int clavier0b()
{
```

```
 Serial.println("Affichage clavier0b");
 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(user_value); //calculer dans la fonction onclick
       }
}


bool Touch_getXY(void)
{
int change_key=0;

   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++;
   if (tp.z==0)
     {
      cpt_unpressed++;
      oldkey="-1";
     }
   else
     {
      cpt_unpressed=0;
     }
    if (cpt_unpressed>=cpt_unpressed_max)
     {
      cpt_unpressed=0; //Evite le debordement de pile
     }
    // 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;


   col_temp=(tp.y-180)/50; //5 colonnes donc /50
   ligne_temp=((480-tp.x)/24);  //
   Serial.print("col_temp");
    Serial.println(col_temp);
   Serial.print("ligne_temp");
    Serial.println(ligne_temp);

   switch (col_temp)
   {
     case 0 ... 2 : col=0;
                    Serial.println("Switch col_Temp=0..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;
```

```
            }
             Serial.print("Switch ligne_Temp=");
             Serial.println(ligne);
            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 ... 18 :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;
 } //end  switch (col_temp)
 key_pressed= button_value[page][col][ligne];
 Serial.print ("oldKey value : ");
 Serial.println(oldkey);
 Serial.print ("Key value : ");
 Serial.println(key_pressed);
 Serial.print ("change_key : ");
 Serial.println(change_key);
//en mode calculatrice - saisir 2 fois le meme nombre
if (change_key==0 &&  oldkey==key_pressed)
 {
  Serial.println("doublon de key");
  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();

 }

//Si 2 touches différentes
if (oldkey!=key_pressed)
 {
 //Serial.print ("Key value : ");
 //Serial.println(key_pressed);
 oldkey=key_pressed;
 Serial.print(oldcol);
 Serial.print(oldligne);
 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();

        }
       if (tp.z<100)
       {
         key_pressed=-1;
         change_key=-1;
       }
       else
       {
         change_key=0;
       }

    } //End else
   }// en if pressed


}// end function


void onclick()
{
  int z=0;
  Serial.println("onclick");
  Serial.println(ligne);
  Serial.println(col);
  Serial.println("*");

  //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 (page==0) // Stak
          {
           //je bascule sur ecran 1 - format calculatrice et j'enregistre la commande
           cmdfromarduino=button_value[page][ligne][col];
          }
     if (page==2) // Stak
          {
           //je bascule sur ecran 1 - format calculatrice et j'enregistre la commande
           cmdfromarduino=button_value[page][ligne][col];

          }
     /*
     cmdfromarduino="Tag=";
     cmdfromarduino+=Stringcmd;
     Serial.print("cmdfromarduino :");
     Serial.println(cmdfromarduino);
     Serial2.println(cmdfromarduino);
     Stringfield="";
     //efface ecran
     */
     draw_buttons(page);
      cmdfromarduino="";
      datafromarduino="";
      user_value=0;

    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;
     goto suite;
  } //end if touche CLR

if (ligne==3 && col==4) // touche ->
  {

    if (page==page_max)
     {
      page=0;
     }
    else
     {
      page++;
     }
    draw_buttons(page);
    col=0;
    ligne=0;
    oldcol=0;
    oldligne=0;
    key_pressed=-1;
    oldkey=-1;
    goto suite;
  }  //end if touche ->

 switch (page)
 {
  case 0 :
         cmdfromarduino=button_value[page][ligne][col];
         break; //page 0
  case 1 :
          if(ligne==1 )
          {
```

```
        Serial.println("page1 - ligne1 ");
        Serial.println(button_value[page][ligne][col]);
        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==2 )
       {
        Serial.println("page1 - ligne 2");
        Serial.println(button_value[page][ligne][col]);
        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 - ligne3");
        Serial.println(button_value[page][ligne][col]);
        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 :
       cmdfromarduino=button_value[page][ligne][col];
       if (ligne==3 && col==0) // Niv
        {
          page=1; //calculatrice
          //enregistrement de commande
          draw_buttons(page);
        }

       if (ligne==3 && col==1) // Stack
        {
```

```
            page=1; //calculatrice
            //enregistrement de commande
            draw_buttons(page);
           }
          break;//page 2
  }

suite:
  z=0;
  /*
  Serial.print("page ? ");
  Serial.println(page);
  Serial.print("ligne ? ");
  Serial.println(ligne);
  Serial.print("col ? ");
  Serial.println(col);
  */
} //end onclick
```