

Programmes Arduino Minitel

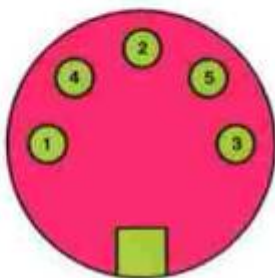
Librairies

[Libraries Minitel1B_Soft](#)

1.1 Prise mécanique

La prise péri-informatique est du type DIN 5 broches femelle sur laquelle sont disponibles les signaux suivants :

- **broche 1** : réception des données par le terminal (signal Rx) ;
- **broche 2** : masse ;
- **broche 3** : émission de données par le terminal (signal Tx) ;
- **broche 4** : périphérique en transmission (signal PT) ;
- **broche 5** : sortie alimentation disponible pour les périphériques. Cette fonction n'est pas disponible sur les versions dont l'identification porte les références Cu2 à Cu4 incluses.



Prise femelle vue de face

- prise Arduino D2(RX) sur 3 minitel (TX)
- prise Arduino D3(TX) sur 1 minitel (RX)
- prise Arduino GND sur 2 minitel (Masse)

Minitel Demo

Minitel-Esp32 Test Laison serie entre minitel et terminal arduino ou esp32

[ESP32_Minitel-Demo000.ino](#)

```
// OK apres test

void setup() {
  Serial.begin(115200); // port debug
  Serial2.begin(1200, SERIAL_7E1); // port minitel
}
```

```
void loop() {  
  // redirection debug -> minitel  
  while (Serial.available() > 0) {  
    Serial2.write(Serial.read());  
  }  
  // redirection minitel -> debug  
  while (Serial2.available() > 0) {  
    Serial.write(Serial2.read());  
  }  
}
```

Minitel-ESP32 Test differant affichage

[ESP32_Minitel-Demo001.ino](#)

```
// Code OK apres test  
////////////////////////////////////  
/  
/*  
  Minitel1B_Hard - Démo - Version du 11 juin 2017 à 16h00  
  Copyright 2016, 2017 - Eric Sérandour  
  
  >> Légèrement adapté pour l'ESP32 par iodeo  
  
  Documentation utilisée :  
  Spécifications Techniques d'Utilisation du Minitel 1B  
  http://543210.free.fr/TV/stum1b.pdf  
  
////////////////////////////////////  
/  
  
  This program is free software: you can redistribute it and/or modify  
  it under the terms of the GNU General Public License as published by  
  the Free Software Foundation, either version 3 of the License, or  
  (at your option) any later version.  
  
  This program is distributed in the hope that it will be useful,  
  but WITHOUT ANY WARRANTY; without even the implied warranty of  
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
  GNU General Public License for more details.  
  
  You should have received a copy of the GNU General Public License  
  along with this program. If not, see <http://www.gnu.org/licenses/>.  
*/  
////////////////////////////////////  
/
```

```
////////////////////////////////////  
/  
  
// DEBUT DU PROGRAMME  
  
////////////////////////////////////  
/  
  
#include <Minitel1B_Hard.h>  
  
Minitel minitel(Serial2); // Le port utilisé sur ESP32  
  
int wait = 10000;  
  
////////////////////////////////////  
/  
  
void setup() {  
  Serial.begin(115200); // Le port de debug  
  minitel.changeSpeed(minitel.searchSpeed());  
}  
  
////////////////////////////////////  
/  
  
void loop() {  
  demoCaracteres();  
  demoGraphic();  
  demoTailles();  
  demoCouleurs();  
  demoCurseur();  
}  
  
////////////////////////////////////  
/  
  
void newPage(String titre) {  
  minitel.newScreen();  
  minitel.println(titre);  
  for (int i=1; i<=40; i++) {  
    minitel.writeByte(0x7E);  
  }  
  minitel.moveCursorReturn(1);  
}  
  
////////////////////////////////////  
/  
  
void demoCaracteres() {
```

```
newPage("LES CARACTERES");

// Mode texte

minitel.println("MODE TEXTE SANS LIGNAGE :");
for (int i=0x20; i<=0x7F; i++) {
    minitel.writeByte(i);
}
minitel.moveCursorReturn(2);

minitel.println("MODE TEXTE AVEC LIGNAGE :");
minitel.attributs(DEBUT_LIGNAGE); // En mode texte, le lignage est
déclenché par le premier espace rencontré (0x20).
for (int i=0x20; i<=0x7F; i++) {
    minitel.writeByte(i);
}
minitel.attributs(FIN_LIGNAGE);
minitel.moveCursorReturn(2);

// Mode semi-graphique

minitel.textMode();
minitel.println("MODE SEMI-GRAPHIQUE SANS LIGNAGE :");
minitel.graphicMode();
for (int i=0x20; i<=0x7F; i++) {
    minitel.writeByte(i);
}
minitel.moveCursorReturn(2);

minitel.textMode();
minitel.println("MODE SEMI-GRAPHIQUE AVEC LIGNAGE :");
minitel.graphicMode();
minitel.attributs(DEBUT_LIGNAGE);
for (int i=0x20; i<=0x7F; i++) {
    minitel.writeByte(i);
}
minitel.attributs(FIN_LIGNAGE);
minitel.moveCursorReturn(2);

delay(wait);
}

////////////////////////////////////
/

void demoGraphic() {
    newPage("LA FONCTION GRAPHIC");
    minitel.textMode();
    minitel.println("Un caractère semi-graphique est composé de 6 pseudo-
pixels :");
```

```

minitel.println();
minitel.graphicMode();
minitel.attributs(DEBUT_LIGNAGE);
minitel.writeByte(0x7F);
minitel.attributs(FIN_LIGNAGE);
minitel.textMode();
minitel.print(" avec lignage ou ");
minitel.graphicMode();
minitel.writeByte(0x7F);
minitel.textMode();
minitel.println(" sans lignage.");
minitel.println();
String chaine = "";
chaine += "minitel.graphic(0b101011) donne ";
minitel.textMode();
minitel.print(chaine);
minitel.graphicMode();
minitel.graphic(0b101011);
minitel.textMode();
minitel.println();
minitel.println();
chaine = "";
chaine += "minitel.graphic(0b110110,30,15) donne ";
minitel.print(chaine);
minitel.graphicMode();
minitel.graphic(0b110110,30,15);
minitel.noCursor();
delay(2*wait);
}

//////////////////////////////////////
/

void demoTailles() {
  newPage("LES TAILLES");
  minitel.println("GRANDEUR_NORMALE");
  minitel.attributs(DOUBLE_HAUTEUR);
  minitel.print("DOUBLE_HAUTEUR");
  minitel.attributs(DOUBLE_LARGEUR);
  minitel.println();
  minitel.println("DOUBLE_LARGEUR");
  minitel.attributs(DOUBLE_GRANDEUR);
  minitel.println("DOUBLE_GRANDEUR");
  minitel.println();
  minitel.attributs(GRANDEUR_NORMALE);
  minitel.attributs(DEBUT_LIGNAGE); // En mode texte, le lignage est
  déclenché par le premier espace rencontré (0x20).
  minitel.println("SEULEMENT EN MODE TEXTE");
  minitel.attributs(FIN_LIGNAGE);
  minitel.println();
  delay(wait);
}

```

```
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/

void demoCouleurs() {
  newPage("LES COULEURS");
  for (int i=0; i<=1; i++) {
    if (i==0) { minitel.textMode(); }
    if (i==1) { minitel.graphicMode(); }
    minitel.attributs(INVERSION_FOND);
    minitel.print("CARACTERE_NOIR, FOND_BLANC");
    minitel.attributs(FOND_NORMAL);
    minitel.println(" (INVERSION)");
    minitel.attributs(CARACTERE_ROUGE);
    minitel.println("CARACTERE_ROUGE");
    minitel.attributs(CARACTERE_VERT);
    minitel.println("CARACTERE_VERT");
    minitel.attributs(CARACTERE_JAUNE);
    minitel.println("CARACTERE_JAUNE");
    minitel.attributs(CARACTERE_BLEU);
    minitel.println("CARACTERE_BLEU");
    minitel.attributs(CARACTERE_MAGENTA);
    minitel.println("CARACTERE_MAGENTA");
    minitel.attributs(CARACTERE_CYAN);
    minitel.println("CARACTERE_CYAN");
    minitel.attributs(CARACTERE_BLANC);
    minitel.println("CARACTERE_BLANC");
    minitel.println();
  }
  delay(wait);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/

void demoCurseur() {
  minitel.cursor();
  newPage("DEPLACER LE CURSEUR");
  minitel.moveCursorXY(20,12);
  for (int i=1; i<=100; i++) {
    delay(100);
    switch (random(4)) {
      case 0: minitel.moveCursorRight(1+random(3)); break;
      case 1: minitel.moveCursorLeft(1+random(3)); break;
      case 2: minitel.moveCursorDown(1+random(3)); break;
      case 3: minitel.moveCursorUp(1+random(3)); break;
    }
  }
  newPage("POSITIONNER LE CURSEUR");
}
```

```

minitel.textMode();
for (int i=1; i<1000; i++) {
  minitel.moveCursorXY(1+random(40),3+random(22));
  minitel.writeByte(0x20 + random(0x60));
}

minitel.newScreen();
minitel.textMode();
minitel.noCursor();
for (int i=1; i<1000; i++) {
  if (random(4)<3) { minitel.textMode(); }
  else {
    minitel.graphicMode();
    minitel.attributs(DEBUT_LIGNAGE);
  }
  minitel.attributs(0x4C+random(5));
  minitel.writeByte(0x20 + random(0x60));
  minitel.attributs(FIN_LIGNAGE);
}
}

////////////////////////////////////
/

```

ESP32-Minitel Pong

[Esp32-Minitel_Pong.ino](#)

```

//Programme OK

#include <Minitel1B_Hard.h>

#define MINITEL_PORT Serial2 //for ESP32
//#define MINITEL_PORT Serial1 //for Leonardo

#define DEBUG true
#define DEBUG_PORT Serial

#if DEBUG // Debug enabled
  #define debugBegin(x)    DEBUG_PORT.begin(x)
  #define debugPrint(x)    DEBUG_PORT.println(x)
  #define debugPrintHEX(x) DEBUG_PORT.println(x,HEX)
  #define debugPrintBIN(x) DEBUG_PORT.println(x,BIN)
#else // Debug disabled : Empty macro functions
  #define debugBegin(x)
  #define debugPrint(x)
  #define debugPrintHEX(x)
  #define debugPrintBIN(x)

```

```
#endif

// SOUND FX
#define SOUND 1 // score using bip
// #define SOUND2 1 // hit using modem connection

// SCREEN SIZE
#define WIDTH 40
#define HEIGHT 24

// GAME FIELD
#define X1 2 //player1 column
#define X2 40 //player2 column
#define XNET 21 //net column
#define SPACE 2 //score digit space
#define YSCORE 2 //score top row
#define WSCORE 2 //score width
#define HSCORE 4 //score height
#define XSCORE1 14 //player 1 score column - align right
#define XSCORE1_U XSCORE1-WSCORE+1 //unit digit
#define XSCORE1_T XSCORE1_U-WSCORE-SPACE //tenth digit
#define XSCORE2 26 //player 2 score column - align left
#define XSCORE2_T XSCORE2 // unit digit
#define XSCORE2_U XSCORE2+WSCORE+SPACE // tenth digit

// GAME PARAMETER
#define NBALL 15 //NB OF BALLS IN THE MATCH

// BALL DIRECTION
#define HAUT_DROITE 1
#define BAS_DROITE 2
#define BAS_GAUCHE 3
#define HAUT_GAUCHE 4

Minitel minitel(MINITEL_PORT);

int yP1 = 13; //player1 position
int yP2 = 13; //player2 position
int xBall = 0;
int yBall = 0;
int xBallOld = 0;
int yBallOld = 0;
int d = HAUT_DROITE; //ball direction
int p1 = 0; //player1 score
int p2 = 0; //player2 score
int startingCounter = 0;
int level = 2; //the smaller the quicker
int bypassCounter = 0;
int hitCounter = 0; //nb hit
bool ping = true;
```


x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x37, 0x1B, 0x57, 0x23, 0x23, 0x23, 0x2F, 0x1B, 0x50, 0x6F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x56, 0x3F, 0x1B, 0x57, 0x23, 0x1B, 0x41, 0x21, 0x1B, 0x45, 0x23, 0x1B, 0x40, 0x23, 0x1B, 0x52, 0x6F, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x57, 0x3F, 0x1B, 0x41, 0x21, 0x1B, 0x47, 0x7F, 0x1B, 0x45, 0x22, 0x1B, 0x40, 0x6B, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x54, 0x3F, 0x1B, 0x57, 0x2F, 0x23, 0x1B, 0x44, 0x23, 0x1B, 0x40, 0x1B, 0x53, 0x6B, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x40, 0x2A, 0x1B, 0x54, 0x6F, 0x1B, 0x50, 0x7F, 0x7F, 0x1B, 0x57, 0x27, 0x1B, 0x47, 0x21, 0x7F, 0x1B, 0x42, 0x60, 0x1B, 0x46, 0x30, 0x1B, 0x47, 0x7F, 0x1B, 0x45, 0x22, 0x1B, 0x40, 0x1B, 0x52, 0x6B, 0x1B, 0x50, 0x7F, 0x7F, 0x1B, 0x57, 0x3F, 0x1B, 0x43, 0x21, 0x1B, 0x46, 0x60, 0x1B, 0x40, 0x70, 0x1B, 0x45, 0x30, 0x1B, 0x43, 0x22, 0x1B, 0x40, 0x6F, 0x1B, 0x50, 0x7F, 0x7F, 0x1B, 0x52, 0x3F, 0x1B, 0x41, 0x1B, 0x57, 0x21, 0x1B, 0x47, 0x7F, 0x1B, 0x46, 0x60, 0x1B, 0x42, 0x70, 0x1B, 0x44, 0x1B, 0x53, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x6A, 0x1B, 0x51, 0x7C, 0x1B, 0x57, 0x34, 0x1B, 0x47, 0x22, 0x1B, 0x40, 0x1B, 0x56, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x53, 0x3F, 0x1B, 0x43, 0x1B, 0x57, 0x21, 0x1B, 0x44, 0x60, 0x1B, 0x40, 0x1B, 0x53, 0x7E, 0x1B, 0x50, 0x7E, 0x7D, 0x1B, 0x57, 0x74, 0x1B, 0x43, 0x30, 0x1B, 0x44, 0x2A, 0x1B, 0x40, 0x1B, 0x50, 0x7F, 0x7F, 0x1B, 0x57, 0x21, 0x1B, 0x47, 0x60, 0x1B, 0x40, 0x7E, 0x1B, 0x50, 0x7F, 0x1B, 0x53, 0x75, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x54, 0x3F, 0x1B, 0x57, 0x21, 0x1B, 0x47, 0x60, 0x1B, 0x40, 0x78, 0x1B, 0x54, 0x7E, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x55, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x35, 0x1B, 0x40, 0x1B, 0x56, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x44, 0x1B, 0x57, 0x25, 0x1B, 0x47, 0x7F, 0x1B, 0x40, 0x1B, 0x56, 0x7E, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x57, 0x75, 0x1B, 0x47, 0x7F, 0x1B, 0x40, 0x1B, 0x53, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x42, 0x1B, 0x57, 0x35, 0x1B, 0x47, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x6A, 0x7F, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x56, 0x37, 0x1B, 0x47, 0x1B, 0x57, 0x21, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x40, 0x60, 0x1B, 0x54, 0x7E, 0x1B, 0x50, 0x7F, 0x1B, 0x43, 0x1B, 0x57, 0x35, 0x1B, 0x44, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x35, 0x1B, 0x45, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x7F, 0x1B, 0x42, 0x1B, 0x57, 0x35, 0x1B, 0x47, 0x6A, 0x1B, 0x40, 0x1B, 0x50, 0x6A, 0x7F, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x57, 0x35, 0x1B, 0x47, 0x6A, 0x1B, 0x40, 0x1B, 0x51, 0x6A, 0x1B, 0x50, 0x37, 0x1B, 0x44, 0x1B, 0x57, 0x23, 0x23, 0x1B, 0x40, 0x1B, 0x53, 0x6B, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x52, 0x35, 0x1B, 0x47, 0x1B, 0x57, 0x7F, 0x1B, 0x40, 0x60, 0x70, 0x7C, 0x1B, 0x54, 0x7E, 0x1B, 0x50, 0x7F, 0x7F, 0x1B, 0x46, 0x1B, 0x57, 0x35, 0x1B, 0x42, 0x22, 0x1B, 0x40, 0x1B, 0x54, 0x6F, 0x1B, 0x50, 0x7F, 0x7F, 0x7F, 0x7F, 0x1B, 0x55, 0x3F, 0x1B, 0x47, 0x1B, 0x57, 0x25, 0x1B, 0x40, 0x6A, 0x1B, 0x50, 0x7F, 0x1B, 0x42, 0x1B, 0x57, 0x35, 0x1B, 0x47, 0x6A, 0x1B, 0x40, 0x1B, 0


```
void setup() {

  debugBegin(115200);
  debugPrint("debug port ready");

  delay(500); // wait minitel to init

  //init minitel at 4800 bauds
  if (minitel.searchSpeed() != 4800) { // search speed
    if (minitel.changeSpeed(4800) < 0) { // set to 4800 if different
      minitel.searchSpeed(); // search speed again if
change has failed
    }
  }

  minitel.modeVideotex();

  minitel.echo(false);
}

void loop() {
  welcome();
  startGame();
}

void welcome() {

  //draw welcome screen
  minitel.newScreen();
  minitel.attributes(CARACTERE_BLANC);
  minitel.attributes(FOND_NOIR);
  for (int i=0; i<LONGUEUR_TRAME_IMAGE; i++) {
    minitel.writeByte(pgm_read_byte_near(IMAGE + i));
  }

  minitel.textMode();
  minitel.noCursor();
  minitel.attributes(CARACTERE_BLANC);
  minitel.attributes(DOUBLE_HAUTEUR);
  minitel.moveCursorXY(12,19);
  minitel.attributes(CLIGNOTEMENT);
  minitel.print("APPUYER SUR ENTREE");

  minitel.attributes(CARACTERE_BLEU);
  minitel.attributes(GRANDEUR_NORMALE);
  minitel.attributes(FIXE);
  minitel.moveCursorXY(1,23);
}
```

```
// 40 char -----*****-----*****
minitel.print("PLAYER 1                      PLAYER 2");
minitel.print("UP:Q DOWN:W                  UP:J DOWN:N");

//flush any input
while(MINITEL_PORT.available()) {
  byte b = MINITEL_PORT.read();
  debugPrint(b);
}
//wait touch is pressed
while(getKeyCodeOverride() != 141) {
  delay(10);
}

//clean up
minitel.newScreen();
minitel.attributes(FIXE);
minitel.graphicMode();
}

void startGame() {

  //draw game field
  drawGameField();
  debugPrint("game field done");
  //init game parameters
  initGame();
  debugPrint("init game done");
  //start
  while (p1+p2 < NBALL) {
    playGame();
  }
  minitel.attributes(CLIGNOTEMENT);
  if (p1>p2) drawScore1(p1);
  else drawScore2(p2);
#ifdef SOUND
  minitel.bip();
  delay(1000);
  minitel.bip();
  delay(1000);
  minitel.bip();
  delay(1000);
#else delay(5000);
#endif
#ifdef SOUND2
  if (!ping) ping = pingpong(ping);
#endif

  p1 = 0;
  p2 = 0;
}
```

```
yP1 = 13;
yP2 = 13;
xBall = 0;
yBall = 0;
ping = true;

}

void handlePlayer() {

    int dy1 = 0;
    int dy2 = 0;

    byte key = getKeyCodeOverride();

    if (key == 215 && yP1<22) dy1++;
    if (key == 209 && yP1>3) dy1--;
    if (key == 78 && yP2<22) dy2++;
    if (key == 202 && yP2>3) dy2--;

    minitel.graphic(0b111111, X1, yP1+3*dy1);
    minitel.moveCursorXY(X1, yP1-2*dy1);
    if (dy1!=0) minitel.graphic(0b000000);
    else minitel.graphic(0b111111); //preserve frame rate
    yP1+=dy1;

    minitel.graphic(0b111111, X2, yP2+3*dy2);
    minitel.moveCursorXY(40, yP2-2*dy2);
    if (dy2!=0) minitel.graphic(0b000000);
    else minitel.graphic(0b111111); //preserve frame rate
    yP2+=dy2;

}

void playGame() {

    handlePlayer();

    if (startingCounter > 0) {
        countdown();
    }
    else {
        //updateLevel
        if (hitCounter == 3) level=1;
        if (hitCounter == 6) level=0;
        if (bypassCounter < level) {
            //bypass frame to slow down
            bypassCounter++;
        }
    }
    else {
```

```
bypassCounter=0;

xBallOld = xBall;
yBallOld = yBall;

// move ball
if(d == HAUT_DROITE){
    xBall+=2;
    yBall--;
}
if(d == BAS_DROITE){
    xBall+=2;
    yBall++;
}
if(d == BAS_GAUCHE){
    xBall-=2;
    yBall++;
}
if(d == HAUT_GAUCHE){
    xBall-=2;
    yBall--;
}

//erase old ball
eraseBall(xBallOld, yBallOld);
//draw new ball
minitel.graphic(0b111111, xBall, yBall);

// top limit
if(yBall == 1) {
    if(d == HAUT_DROITE) d = BAS_DROITE;
    if(d == HAUT_GAUCHE) d = BAS_GAUCHE;
}

// bottom limit
if(yBall == HEIGHT) {
    if (d == BAS_GAUCHE) d = HAUT_GAUCHE;
    if (d == BAS_DROITE) d = HAUT_DROITE;
}

// player1 side
if (xBall <= X1 + 2) {
    // player1 send back
    if(abs(yBall-yP1) < 3) {
        if(d == BAS_GAUCHE) d = BAS_DROITE;
        if(d == HAUT_GAUCHE) d = HAUT_DROITE;
        hitCounter++;
#ifdef SOUND2
        ping = pingpong(ping);
#endif
    }
}
```

```
        else{ // player2 win
#ifdef SOUND
            minitel.bip();
#endif
            p2++;
            drawScore2(p2);
            initGame();
        }
    }

    // player2 side
    if (xBall >= X2 - 2) {
        // player2 send back
        if(abs(yBall-yP2) < 3) {
            if(d == BAS_DROITE) d = BAS_GAUCHE;
            if(d == HAUT_DROITE) d = HAUT_GAUCHE;
            hitCounter++;
#ifdef SOUND2
            ping = pingpong(ping);
#endif
        }
        else{ // player1 win
#ifdef SOUND
            minitel.bip();
#endif
            p1++;
            drawScore1(p1);
            initGame();
        }
    }
}

bool pingpong(bool ping) {
    if (ping) {
        minitel.connexion(true);
        return false;
    }
    else {
        minitel.connexion(false);
        return true;
    }
}

void eraseBall(int x, int y) {
    // erase ball preserving game field and score

    minitel.moveCursorXY(x,y);
}
```

```

bool erase = true;

if (x == XNET) { // Ball in net
  if (y%2 == 1) erase = false;
}

if (y >= YSCORE && y < YSCORE + HSCORE) { // Ball in score
  if (x >= XSCORE1_T && x < XSCORE1_T + WSCORE) { //ball in score1
tenth
    if (p1 >= 10) {
      drawDigit(p1/10, XSCORE1_T, YSCORE);
      erase = false;
    }
  }
  if (x >= XSCORE1_U && x < XSCORE1_U + WSCORE) { //ball in score1
unit
    drawDigit(p1%10, XSCORE1_U, YSCORE);
    erase = false;
  }
  if (x >= XSCORE2_T && x < XSCORE2_T + WSCORE) { //ball in score2
tenth
    if (p2 >= 10) {
      drawDigit(p2/10, XSCORE2_T, YSCORE);
      erase = false;
    }
  }
  if (x >= XSCORE2_U && x < XSCORE2_U + WSCORE) { //ball in score2
unit
    drawDigit(p2%10, XSCORE2_U, YSCORE);
    erase = false;
  }
}
if (erase) minitel.graphic(0b000000);
}

/*void drawDigit(int num, int x, int y) {
  for (int i = 0; i < WSCORE; i++) {
    for (int j = 0; j < HSCORE; j++) {
      minitel.graphic(digit[num][i+WSCORE*j],x+i,y+j);
    }
  }
}
*/

void initGame() {
  // init level
  hitCounter = 0;
  level = 2;
  bypassCounter = 2;
  // get random starting point

```

```
xBallOld = xBall;
yBallOld = yBall;
d = random(1,4);
xBall = random(1,5)*2+1;
if (d>2) xBall = 40-xBall;
yBall = random(3,22);
// draw new ball
minitel.graphic(0b111111, xBall, yBall);
// starting countdown
startingCounter = 10;
}

void countdown() {
  if (xBallOld != 0) {
    if (startingCounter == 8) {
      minitel.attributs(CARACTERE_BLEU);
      minitel.graphic(0b111111, xBallOld, yBallOld);
      minitel.attributs(CARACTERE_BLANC);
    }
    if (startingCounter == 6) {
      minitel.attributs(CARACTERE_VERT);
      minitel.graphic(0b111111, xBallOld, yBallOld);
      minitel.attributs(CARACTERE_BLANC);
    }
    if (startingCounter == 4) {
      minitel.attributs(CARACTERE_BLEU);
      minitel.graphic(0b111111, xBallOld, yBallOld);
      minitel.attributs(CARACTERE_BLANC);
    }
    if (startingCounter == 2) {
      minitel.attributs(CARACTERE_VERT);
      minitel.graphic(0b111111, xBallOld, yBallOld);
      minitel.attributs(CARACTERE_BLANC);
    }
    if (startingCounter == 1) {
      minitel.graphic(0b000000, xBallOld, yBallOld); //erase ball
#ifdef SOUND2
      ping = pingpong(ping);
#endif
    }
  }
  startingCounter--;
}

byte getkeyCodeOverride() {
  byte b = 255;
  if (MINITEL_PORT.available()) {
    b = MINITEL_PORT.read();
    MINITEL_PORT.flush();
  }
}
```

```

    debugPrint(b);
}
return b;
}

void drawGameField() {
    //draw net
    for (int i = 1; i < HEIGHT; i+=2) {
        minitel.graphic(0b111111, 21, i);
    }
    //draw players
    for (int i = -2; i < 3; i++) {
        minitel.graphic(0b111111,X1,yP1+i);
        minitel.graphic(0b111111,X2,yP2+i);
    }
    // draw score
    drawScore1(p1);
    drawScore2(p2);
}

void drawScore1(int score) {
    drawDigit(score%10, XSCORE1_U, YSCORE);
    if (score>=10) drawDigit(score/10, XSCORE1_T, YSCORE);
}

void drawScore2(int score) {
    drawDigit(score%10, XSCORE2_U, YSCORE);
    if (score>=10) drawDigit(score/10, XSCORE2_T, YSCORE);
}

void drawDigit(int num, int x, int y) {
    for (int i = 0; i < WSCORE; i++) {
        for (int j = 0; j < HSCORE; j++) {
            minitel.graphic(digit[num][i+WSCORE*j],x+i,y+j);
        }
    }
}
}
}

```

Arduino_Minitel.ino

```

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);

byte gauche = 8;
byte droite = 9;
byte bas = 10;
byte haut = 11;
byte debutDeLigne = 13;
byte hautGauche = 30;

```

```
byte hautGaucheEfface = 12;
byte separateurDeSousArticle = 31;
byte remplissageEspace = 24; //Remplit le reste da la rangée avec des
espaces
byte CBleu = 68; // caractère niveau gris bleu
byte CBlanc = 71; // caractère couleur blanche
byte Clignote = 72 ; // caractère clignote
byte Fixe = 73 ; // caractère fixe
byte NormalH = 76 ; // taille caractère normal
byte DoubleH = 77 ; // double hauteur
byte Ligne = 90 ; // caractère souligné
byte SLine = 89; // annule soulingé

short incomming;
char inascii = » »;
short outcomming;
int TS = 0; // touche spéciale
String TSS = « »; // touche spéciale texte

void setup() {

Serial.begin(1200); // port serie vers le PC
mySerial.begin(1200); // port serie vers le minitel

mySerial.write(hautGaucheEfface); //efface l'écran
// serialprint7(0x0E); // passe en mode graphique
delay(500);
sendMessage(« BONJOUR »);
CR();
sendMessage(« BONSOIR »);
Gauche(3);
sendMessage(« REBONSOIR »);
Droite(3);
delay(1000);
sendMessage(« JOUR »);
CR();
ESC(Clignote);
sendMessage(« BONJOUR »);
ESC(Fixe);
CR();
CR();
ESC(DoubleH);
sendMessage(« BONJOUR »);
CR();
ESC(NormalH);
ESC(CBleu);
sendMessage(« BONJOUR »);
CR();
ESC(CBlanc);
sendMessage(« BONJOUR »);
```

```

CR();
ESC(Ligne);
sendMessage( » BONJOUR »);
CR();
ESC(SLigne);
sendMessage(« BONJOUR »);
CR();
Serial.println( » « );
}

char modifyParity(char c) {
char i = 1 << 6;
boolean p = false;
c &= B01111111;
while (i) {
if (c & i) {
p = !p;
}
i >>= 1;
}
c |= p << 7;
return c;
}

void sendMessage(char *msg) {
int i = 0;
while (msg[i]) {
serialprint7(msg[i]);
i++;
}
Serial.write(msg);
Serial.flush();
}

void serialprint7(byte b) // permet d'ecrire en 7 bits + parité sur le
software serial
{
boolean i = false;
for (int j = 0; j < 8; j++)
{
if (bitRead(b, j) == 1) i = !i; //calcul de la parité
}
if (i) bitWrite(b, 7, 1); //écriture de la partié
else bitWrite(b, 7, 0); //écriture de la partié
mySerial.write(b); //écriture du byte sur le software serial
}

void Gauche(int g) {
for (int i = 0; i <= g; i++) {
serialprint7(9);
}
}

```

```
}

void Droite(int g) {
for (int i = 0; i <= g; i++) {
serialprint7(8);
}
}

void Haut(int g) {
for (int i = 0; i <= g; i++) {
serialprint7(11);
}
}

void ESC(int c){
serialprint7(27);
serialprint7(c);
}

void CR() {
serialprint7(13);
serialprint7(10);
}

void loop() //tout ce que je recois sur le port serie, je le renvoi sur
le software serial
{

// Serial.println(« loop »);
if (Serial.available()) {
outcomming = Serial.read();
Serial.print(« saisie arduino: »);
Serial.println (outcomming);
// serialprint7(incomming);
serialprint7(outcomming);
}

if (mySerial.available()) {
incomming = mySerial.read() & B01111111; // ignore parity check //
ignore parity check
Serial.print(« saisie minitel : »);
inascii = char(incomming);
Serial.println (inascii);
if (TS == 1) {
touchespeciales();
TS = 0;
}
if (incomming == 19) {
TS = 1;
}
}
```

```
}  
  
}  
  
void touchespeciales() {  
  switch (incomming) {  
  case 70:  
    Serial.println (« Sommaire »);  
    TSS = « Sommaire »;  
    break;  
  case 69:  
    Serial.println (« Annulation »);  
    TSS = « Annulation »;  
    break;  
  case 66:  
    Serial.println (« Retour »);  
    TSS = « Retour »;  
    break;  
  case 67:  
    Serial.println (« Repetition »);  
    TSS = « Repetition »;  
    break;  
  case 68:  
    Serial.println (« Guide »);  
    TSS = « Guide »;  
    break;  
  case 71:  
    Serial.println (« Correction »);  
    TSS = « Correction » ;  
    break;  
  case 72:  
    Serial.println (« Suite »);  
    TSS = « Suite »;  
    break;  
  case 65:  
    Serial.println (« Envoi »);  
    TSS = « Envoi »;  
    break;  
  case 89:  
    Serial.println (« Connexion »);  
    TSS = « Connexion »;  
    break;  
  
  }  
}
```

Programme 2 Minitel1B_ChessUI.ino

Minitel1B_ChessUI.ino

```
// programme tester = OK

#include <Minitel1B_Hard.h>

#define MINITEL_PORT Serial2 //for ESP32
//#define MINITEL_PORT Serial1 //for Leonardo

#define DEBUG true
#define DEBUG_PORT Serial

#if DEBUG // Debug enabled
    #define debugBegin(x)    DEBUG_PORT.begin(x)
    #define debugPrint(x)   DEBUG_PORT.println(x)
    #define debugPrintHEX(x) DEBUG_PORT.println(x,HEX)
    #define debugPrintBIN(x) DEBUG_PORT.println(x,BIN)
#else // Debug disabled : Empty macro functions
    #define debugBegin(x)
    #define debugPrint(x)
    #define debugPrintHEX(x)
    #define debugPrintBIN(x)
#endif

#define CASE_WIDTH 4
#define CASE_HEIGHT 3
#define BOARD_TOP 1
#define BOARD_LEFT 1
#define PIECE_WIDTH 3
#define PIECE_HEIGHT 3

#define SCORE_TOP 1
#define SCORE_LEFT 33
#define SCORE_WIDTH 8
#define SCORE_HEIGHT 24
#define SCORE_BLACK_TOP 1
#define SCORE_WHITE_TOP 16
#define SCORE_HEIGHT_2 9 // indiv. score frame
#define SCORE_MOVE_TOP 10
#define SCORE_HEIGHT_3 6 // move frame

Minitel minitel(MINITEL_PORT);

enum { VOID, PAWN, ROOK, KNIGHT, BISHOP, QUEEN, KING};
enum {_BLACK = 0, _WHITE = 128};
```

```

byte piece[7][PIECE_WIDTH*PIECE_HEIGHT] = {
  // pieces en caractères semi-graphiques 3 par 3 décrites par lignes
  // de haut-gauche à bas-droite
  {0b000000, 0b000000, 0b000000, 0b000000, 0b000000, 0b000000,
  0b000000, 0b000000, 0b000000}, // VOID
  {0b000000, 0b000000, 0b000000, 0b000101, 0b101111, 0b000000,
  0b000100, 0b101100, 0b000000}, // PAWN
  {0b000010, 0b000010, 0b000010, 0b110101, 0b111101, 0b100000,
  0b011100, 0b011100, 0b001000}, // ROOK
  {0b000000, 0b000111, 0b000010, 0b011110, 0b011101, 0b101010,
  0b001100, 0b111100, 0b001000}, // KNIGHT
  {0b000001, 0b001011, 0b000000, 0b111111, 0b101111, 0b101010,
  0b011100, 0b111100, 0b001000}, // BISHOP
  {0b001001, 0b000011, 0b001000, 0b000111, 0b101111, 0b000010,
  0b111100, 0b011100, 0b101000}, // QUEEN
  {0b000001, 0b001011, 0b000000, 0b000111, 0b101111, 0b000010,
  0b111100, 0b011100, 0b101000} // KING
};

byte board[8][8] { //top-left to bottom-right - _BLACK or _WHITE added
  later
  /*{ROOK,   KNIGHT, BISHOP, QUEEN, KING,   BISHOP, KNIGHT, ROOK },
  {PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN },
  {VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID },
  {VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID },
  {VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID },
  {VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID,   VOID },
  {PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN,   PAWN },
  {ROOK,   KNIGHT, BISHOP, QUEEN, KING,   BISHOP, KNIGHT, ROOK }*/
};

int cx = 0; // 0-7 > A-H
int cy = 7; // 0-7 > 8-1

int scx = -1; // first case selected
int scy = -1; //

String moveStr = "  - ";
String lastStr = "  - ";

byte player = _WHITE;

void setup() {

  debugBegin(115200);
  debugPrint("> Debug start");

  delay(500);

  // Minitel setup
  int baud = minitel.searchSpeed();

```

```
debugPrint("> Minitel is at " + String(baud) + " bds");
if (baud != 4800) {
  debugPrint("> Set to 4800 bauds");
  if (minitel.changeSpeed(4800) < 0) { // try set speed to 4800 if
needed
  debugPrint(" *** Failed to change speed ***");
  minitel.searchSpeed(); // search back if failed
  }
}

//minitel.modeVideotex();
minitel.echo(false);
minitel.extendedKeyboard(); //need arrows
minitel.clearScreen();
minitel.moveCursorXY(1,1);
minitel.noCursor();
minitel.attributes(FIXE);
debugPrint("> Minitel setup done");

// Intialize game board
initBoard();
drawBoard();
drawAllPieces();
drawScoreBoard();

//hoverCase(cx,cy, true);
}

String keyboardInput = "";

void loop() {

  char c = 0;

  c = getKeyboardInput();

  switch (c) {
    // nothing
    case 0:      break;

    // move on board
    case UP:     moveUp();      break;
    case DOWN:  moveDown();    break;
    case LEFT:  moveLeft();    break;
    case RIGHT: moveRight();   break;

    // cancel selection
    case DEL:
```

```
case CAN:
    if (scx != -1) { // cancel selection
        selectCase(scx, scy, false);
        scx = -1; scy = -1;
        moveStr = "  - ";
        writeMove();
    }
    break;

// move selection
case CR:
    if (scx == -1 || scy == -1) {
        // first case selection
        scx = cx;
        scy = cy;
        selectCase(cx, cy, true);
        moveStr.setCharAt(1, cx+65); // A(65)-H
        moveStr.setCharAt(2, 56-cy); // 8(56)-1
        writeMove();
    }
    else {
        if (cx == scx && cy == scy) {
            // cancel first case selection
            selectCase(cx, cy, false);
            moveStr = "  - ";
            writeMove();
            scx = -1; scy = -1;
        }
        else {
            // second case selection
            //TODO: verifiy legal move
            moveStr.setCharAt(4, cx+65); // A(65)-H
            moveStr.setCharAt(5, 56-cy); // 8(56)-1
            writeMove();
            board[cx][cy] = board[scx][scy];
            board[scx][scy] = VOID;
            erasePiece(scx, scy);
            selectCase(scx, scy, false);
            drawPiece(cx, cy, board[cx][cy]);
            scx = -1; scy = -1;
            if (player == _WHITE) player = _BLACK;
            else player = _WHITE;
            lastStr = moveStr;
            moveStr = "  - ";
            redrawMove();
        }
    }
    break;
}
}
```

```
void initBoard() {
  for (int i = 0; i < 5; i++) board[i][0] = (i+2) + _BLACK;
  for (int i = 5; i < 8; i++) board[i][0] = (5-i+4) + _BLACK;
  for (int i = 0; i < 8; i++) board[i][1] = PAWN + _BLACK;
  for (int j = 2; j < 6; j++) {
    for (int i = 0; i < 8; i++) board[i][j] = VOID;
  }
  for (int i = 0; i < 5; i++) board[i][7] = (i+2) + _WHITE;
  for (int i = 5; i < 8; i++) board[i][7] = (5-i+4) + _WHITE;
  for (int i = 0; i < 8; i++) board[i][6] = PAWN + _WHITE;
}
```

```
void drawBoard() {

  minitel.textMode();
  minitel.attributs(GRANDEUR_NORMALE);

  minitel.graphicMode();
  minitel.moveCursorXY(BOARD_LEFT, BOARD_TOP);
  bool dark = false;
  int cy = 8;
  while (cy > 0) {
    int row = 1;
    while (row <= CASE_HEIGHT) {
      int cx = 1;
      while (cx < 9) {
        if (dark) minitel.attributs(FOND_BLEU);
        else minitel.attributs(FOND_VERT);
        minitel.graphic(0b000000);
        minitel.repeat(CASE_WIDTH - 1);
        if (row < 3) {
          minitel.moveCursorLeft(CASE_WIDTH);
          minitel.textMode();
          if (dark) minitel.attributs(CARACTERE_BLEU);
          else minitel.attributs(CARACTERE_VERT);
          minitel.attributs(INVERSION_FOND);
          if (row == 1) minitel.printChar(cx+64); // A-H
          else minitel.printChar(cy+48); // 1-8
          minitel.moveCursorRight(CASE_WIDTH - 1);
          minitel.graphicMode();
        }
        dark = !dark;
        cx++;
      }
      minitel.moveCursorLeft(CASE_WIDTH*8);
      minitel.moveCursorDown(1);
      row++;
    }
    dark = !dark;
    cy--;
  }
```

```

}
}

void drawScoreBoard() {

    drawBackground();

    drawFrame(SCORE_LEFT, SCORE_BLACK_TOP, SCORE_WIDTH, SCORE_HEIGHT_2,
_BLACK);
    //drawFrame(SCORE_LEFT, SCORE_MOVE_TOP, SCORE_WIDTH, SCORE_HEIGHT_3,
_WHITE);
    drawFrame(SCORE_LEFT, SCORE_WHITE_TOP, SCORE_WIDTH, SCORE_HEIGHT_2,
_WHITE);

    minitel.textMode();
    minitel.attributs(GRANDEUR_NORMALE);
    int sx = SCORE_BLACK_TOP;
    minitel.attributs(CARACTERE_NOIR);
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print("BLACK ");
    minitel.attributs(FOND_NORMAL);
    sx+=2;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print("time:");
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print(" --:--");
    minitel.attributs(FOND_NORMAL);
    sx+=2;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print("str:");
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print("  --");

    sx = SCORE_MOVE_TOP;
    minitel.attributs(CARACTERE_BLANC);
    minitel.attributs(FOND_NORMAL);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print("move:");
    minitel.attributs(CARACTERE_BLANC);
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1, sx);
    minitel.print(moveStr);
    minitel.attributs(CARACTERE_NOIR);

```

```
minitel.attributs(FOND_NORMAL);
sx++;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print("last:");
minitel.attributs(CARACTERE_NOIR);
minitel.attributs(INVERSION_FOND);
sx++;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print(lastStr);

sx = SCORE_WHITE_TOP;
minitel.attributs(CARACTERE_BLANC);
minitel.attributs(INVERSION_FOND);
sx++;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print("WHITE ");
minitel.attributs(FOND_NORMAL);
sx+=2;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print("time:");
minitel.attributs(INVERSION_FOND);
sx++;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print(" --:--");
minitel.attributs(FOND_NORMAL);
sx+=2;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print("str:");
minitel.attributs(INVERSION_FOND);
sx++;
minitel.moveCursorXY(SCORE_LEFT+1,sx);
minitel.print("  --");
}

void drawBackground() {
  int sy = SCORE_TOP;
  minitel.graphicMode();
  minitel.attributs(FOND_MAGENTA);
  while (sy < SCORE_TOP + SCORE_HEIGHT) {
    minitel.moveCursorXY(SCORE_LEFT,sy);
    minitel.graphic(0b000000);
    minitel.repeat(SCORE_WIDTH-1);
    sy++;
  }
}

void writeMove() {
  minitel.textMode();
```

```
    if (player == _WHITE) {
        minitel.attributs(CARACTERE_BLANC);
        minitel.attributs(INVERSION_FOND);
    }
    minitel.moveCursorXY(SCORE_LEFT+1,SCORE_MOVE_TOP+2);
    minitel.print(moveStr);
}

void redrawMove() {
    int sx = SCORE_MOVE_TOP;
    minitel.textMode();
    if (player == _WHITE) minitel.attributs(CARACTERE_BLANC);
    else minitel.attributs(CARACTERE_NOIR);
    minitel.attributs(FOND_NORMAL);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1,sx);
    minitel.print("move:");
    if (player == _WHITE) minitel.attributs(CARACTERE_BLANC);
    else minitel.attributs(CARACTERE_NOIR);
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1,sx);
    minitel.print(moveStr);
    if (player == _WHITE) minitel.attributs(CARACTERE_NOIR);
    else minitel.attributs(CARACTERE_BLANC);
    minitel.attributs(FOND_NORMAL);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1,sx);
    minitel.print("last:");
    if (player == _WHITE) minitel.attributs(CARACTERE_NOIR);
    else minitel.attributs(CARACTERE_BLANC);
    minitel.attributs(INVERSION_FOND);
    sx++;
    minitel.moveCursorXY(SCORE_LEFT+1,sx);
    minitel.print(lastStr);
}

void drawFrame(int x, int y, int w, int h, int c) {
    int sy = y;
    minitel.graphicMode();
    minitel.attributs(FOND_MAGENTA);
    if (c == _BLACK) minitel.attributs(CARACTERE_NOIR);
    else minitel.attributs(CARACTERE_BLANC);
    minitel.moveCursorXY(x,sy);
    minitel.graphic(0b000001);
    minitel.graphic(0b000011);
    minitel.repeat(w-3);
    minitel.graphic(0b000010);
    sy++;
}
```

```
while (sy < y + h - 1) {
    minitel.moveCursorXY(x, sy);
    minitel.graphic(0b010101);
    minitel.graphic(0b000000);
    minitel.repeat(w-3);
    minitel.graphic(0b101010);
    sy++;
}
minitel.moveCursorXY(x, sy);
minitel.graphic(0b010000);
minitel.graphic(0b110000);
minitel.repeat(w-3);
minitel.graphic(0b100000);
}

void drawPiece(int cx, int cy, byte pc) {
    // x : from 0 to 7 - left to right
    // y : from 0 to 7 - top to bottom
    int x = cx * CASE_WIDTH + 1;
    int y = cy * CASE_HEIGHT + 1;

    byte color = _BLACK;
    if (pc > _WHITE) color = _WHITE;
    byte p = pc - color;

    minitel.graphicMode();

    if (color == _WHITE) {
        minitel.attributs(DEBUT_LIGNAGE);
        minitel.attributs(CARACTERE_BLANC);
    }
    else { // _BLACK
        minitel.attributs(CARACTERE_NOIR);
    }
    if ((cx+cy)%2 == 1) minitel.attributs(FOND_BLEU);
    else minitel.attributs(FOND_VERT);

    for (int j = 0; j < PIECE_HEIGHT; j++) {
        minitel.moveCursorXY(x+1, y+j);
        for (int i = 0; i < PIECE_WIDTH; i++) {
            minitel.graphic(piece[p][i+j*PIECE_WIDTH]);
        }
    }
    if (color == _WHITE) {
        minitel.attributs(FIN_LIGNAGE);
    }
}

void erasePiece(int cx, int cy) {
    // x : from 0 to 7 - left to right
```

```
// y : from 0 to 7 - top to bottom
drawPiece(cx, cy, VOID);
}

void drawAllPieces() {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      if (j<2 || j>5) drawPiece(i, j, board[i][j]);
    }
  }
}

void hoverCase(int cx, int cy, bool hover) {
  if (cx == scx && cy == scy) selectCase(cx, cy, true);
  else {
    int x = cx*CASE_WIDTH + 1;
    int y = cy*CASE_HEIGHT + 3;
    bool dark = false;
    if ((cx+cy)%2 == 1) dark = true;
    minitel.moveCursorXY(x,y);
    minitel.graphicMode();
    if (dark) minitel.attributs(FOND_BLEU);
    else minitel.attributs(FOND_VERT);
    if (hover) {
      minitel.attributs(CARACTERE_BLANC);
      minitel.graphic(0b111111);
    }
    else minitel.graphic(0b000000);
  }
}

void selectCase(int cx, int cy, bool sel) {
  int x = cx*CASE_WIDTH + 1;
  int y = cy*CASE_HEIGHT + 3;
  bool dark = false;
  if ((cx+cy)%2 == 1) dark = true;
  minitel.moveCursorXY(x,y);
  minitel.graphicMode();
  if (dark) minitel.attributs(FOND_BLEU);
  else minitel.attributs(FOND_VERT);
  if (sel) {
    minitel.attributs(CARACTERE_NOIR);
    minitel.graphic(0b111111);
  }
  else {
    minitel.graphic(0b000000);
  }
}

void moveUp() {
  if (cy > 0) {
```

```
    hoverCase(cx,cy, false);
    cy--;
    hoverCase(cx,cy, true);
}
}

void moveDown() {
    if (cy < 7) {
        hoverCase(cx,cy, false);
        cy++;
        hoverCase(cx,cy, true);
    }
}

void moveLeft() {
    if (cx > 0) {
        hoverCase(cx,cy, false);
        cx--;
        hoverCase(cx,cy, true);
    }
}

void moveRight() {
    if (cx < 7) {
        hoverCase(cx,cy, false);
        cx++;
        hoverCase(cx,cy, true);
    }
}

char getKeyboardInput() {

    unsigned long key = minitel.getKeyCode();
    if (key != 0) {
        debugPrintHEX(key);
        // key redirection/inhibition
        switch (key) {

            // cancel selection
            case CORRECTION:
            case ANNULATION:
            case RETOUR:
            case ESC:
                return CAN;    break;

            // validate selection
            case ENVOI:
            case SP:
                return CR;    break;

            // navigate
```

```
    case TOUCHE_FLECHE_HAUT:    return UP;    break;
    case TOUCHE_FLECHE_BAS:    return DOWN;  break;
    case TOUCHE_FLECHE_DROITE: return RIGHT; break;
    case TOUCHE_FLECHE_GAUCHE: return LEFT;  break;

    // inhibited
    case CONNEXION_FIN:
    case SOMMAIRE:
    case REPETITION:
    case GUIDE:
    case SUITE:

        return 0; break;

    default: return key;
}
}
else return 0;
}
```

Minitel-ESP32 Demo Teaser

[Minitel-ESP32-DEMO_TEASER.ino](#)

```
//Faire Reset de l'ESP32 à chaque démarrage du minitel

/*
 * Sample code for connexion to minitel videotex server via websocket
 * Requirements: ESP32 connected to minitel DIN port and a WiFi
connexion
 *
 * created by iodeo - dec 2021
 */

#include <WiFi.h>
#include <WebSocketsClient.h> // src:
https://github.com/Links2004/arduinoWebSockets.git
#include <Minitel1B_Hard.h> // src:
https://github.com/eserandour/Minitel1B_Hard.git

// -----
// ----- Minitel port configuration

#define MINITEL_PORT Serial2 // for Minitel-ESP32 devboard
#define MINITEL_BAUD 4800 // 1200 / 4800 / 9600 depending on
minitel type
#define MINITEL_DISABLE_ECHO true // true if characters are repeated
when typing
```

```
// -----  
// ----- Debug port configuration  
  
#define DEBUG true  
  
#if DEBUG  
    #define DEBUG_PORT Serial        // for Minitel-ESP32 devboard  
    #define DEBUG_BAUD 115200       // set serial monitor accordingly  
    #define debugBegin(x)           DEBUG_PORT.begin(x)  
    #define debugPrintf(...)        DEBUG_PORT.printf(__VA_ARGS__)  
#else // Empty macro functions  
    #define debugBegin(x)  
    #define debugPrintf(...)  
#endif  
  
// -----  
// ----- WiFi credentials  
  
const char* ssid      = "xxxxxxxxxxxx"; // your wifi network  
const char* password  = "xxxxxxxxxxxxxxxx"; // your wifi password  
  
// -----  
// ----- Websocket server  
  
/***** TELETEL.ORG ----- connecté le 2 mar 2022  
    */*  
// ws://home.teletel.org:9001/  
char* host = "home.teletel.org";  
int port = 9001;  
char* path = "/";  
bool ssl = false;  
int ping_ms = 0;  
char* protocol = "";  
/**/  
  
/***** 3615 ----- connecté le 2 mar 2022  
// wss://3615co.de/ws  
char* host = "3615co.de";  
int port = 80;  
char* path = "/ws";  
bool ssl = false;  
int ping_ms = 0;  
char* protocol = "";  
/**/  
  
/***** AE ----- connecté le 2 mar 2022  
// ws://3611.re/ws  
// websocket payload length of 0  
char* host = "3611.re";
```

```

int port = 80;
char* path = "/ws";
bool ssl = false;
int ping_ms = 0;
char* protocol = "";
/**/

/***** HACKER ----- connecté le 2 mar 2022
// ws://mntl.joher.com:2018/?echo
// websocket payload length up to 873
char* host = "mntl.joher.com";
int port = 2018;
char* path = "/?echo";
bool ssl = false;
int ping_ms = 0;
char* protocol = "";
/**/

/***** TEASER ----- connecté le 2 mar 2022
// ws://minitel.3614teaser.fr:8080/ws
char* host = "minitel.3614teaser.fr";
int port = 8080;
char* path = "/ws";
bool ssl = false;
int ping_ms = 10000;
char* protocol = "tty";
/**/

/***** SM ----- connecté le 2 mar 2022*/
// wss://wss.3615.live:9991/?echo
// websocket payload length up to 128
char* host = "wss.3615.live";
int port = 9991;
char* path = "/?echo";
bool ssl = true;
int ping_ms = 0;
char* protocol = "";
/**/

WiFiClient client;
WebSocketsClient webSocket;
Minitel minitel(MINITEL_PORT);

void setup() {

  debugBegin(DEBUG_BAUD);
  debugPrintf("\n-----\n");
  debugPrintf("\n> Debug port ready\n");

  // We initiate minitel communication
  debugPrintf("\n> Minitel setup\n");

```

```
int baud = minitel.searchSpeed();
if (baud != MINITEL_BAUD) baud = minitel.changeSpeed(MINITEL_BAUD);
debugPrintf(" - Baud detected: %u\n", baud);
if (MINITEL_DISABLE_ECHO) {
    minitel.echo(false);
    debugPrintf(" - Echo disabled\n");
}

// We connect to WiFi network
debugPrintf("\n> Wifi setup\n");
debugPrintf(" Connecting to %s ", ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    debugPrintf(".");
}
debugPrintf("\n WiFi connected with IP %s\n",
WiFi.localIP().toString().c_str());

// We connect to Websocket server
debugPrintf("\n> Websocket connection\n");
if (protocol[0] == '\0') {
    if (ssl) websocket.beginSSL(host, port, path);
    else websocket.begin(host, port, path);
}
else {
    debugPrintf(" - subprotocol added\n");
    if (ssl) websocket.beginSSL(host, port, path, protocol);
    else websocket.begin(host, port, path, protocol);
}

websocket.onEvent(webSocketEvent);

if (ping_ms != 0) {
    debugPrintf(" - heartbeat ping added\n");
    // start heartbeat (optional)
    // ping server every ping_ms
    // expect pong from server within 3000 ms
    // consider connection disconnected if pong is not received 2 times
    websocket.enableHeartbeat(ping_ms, 3000, 2);
}

debugPrintf("\n> End of setup\n\n");
}

void loop() {

    // Websocket -> Minitel
    websocket.loop();
}
```

```
// Minitel -> Websocket
uint32_t key = minitel.getKeyCode(false);
if (key != 0) {
    debugPrintf("[KB] got code: %X\n", key);
    // prepare data to send over websocket
    uint8_t payload[4];
    size_t len = 0;
    for (len = 0; key != 0 && len < 4; len++) {
        payload[3-len] = uint8_t(key);
        key = key >> 8;
    }
    websocket.sendTXT(payload+4-len, len);
}

}

void websocketEvent(WStype_t type, uint8_t * payload, size_t len) {
    switch(type) {
        case WStype_DISCONNECTED:
            debugPrintf("[WS] Disconnected!\n");
            break;

        case WStype_CONNECTED:
            debugPrintf("[WS] Connected to url: %s\n", payload);
            break;

        case WStype_TEXT:
            debugPrintf("[WS] got %u chars\n", len);
            if (len > 0) {
                debugPrintf(" > %s\n", payload);
                for (size_t i = 0; i < len; i++) {
                    minitel.writeByte(payload[i]);
                }
            }
            break;

        case WStype_BIN:
            debugPrintf("[WS] got %u binaries - ignored\n", len);
            break;

        case WStype_ERROR:
            debugPrintf("[WS] WStype_ERROR\n");
            break;

        case WStype_FRAGMENT_TEXT_START:
            debugPrintf("[WS] WStype_FRAGMENT_TEXT_START\n");
            break;

        case WStype_FRAGMENT_BIN_START:
            debugPrintf("[WS] WStype_FRAGMENT_BIN_START\n");
            break;
    }
}
```

```
break;

case WStype_FRAGMENT:
    debugPrintf("[WS] WStype_FRAGMENT\n");
    break;

case WStype_FRAGMENT_FIN:
    debugPrintf("[WS] WStype_FRAGMENT_FIN\n");
    break;
}
}
```

From: <https://www.magenealogie.chanterie37.fr/www/fablab37110/> - Castel'Lab le Fablab MJC de Château-Renault

Permanent link: <https://www.magenealogie.chanterie37.fr/www/fablab37110/doku.php?id=start:arduino:minitel:programme&rev=1658941896>

Last update: 2023/01/27 16:08

