

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)

[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 de 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 SLigne = 89; // annule souligné

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

[programme2-Minitel.ino](#)

```
#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;
}
```

From:

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

Permanent link:

<https://www.fablab37110.chanterie37.fr/doku.php?id=start:arduino:minitel:programme&rev=1657056311>

Last update: **2023/01/27 16:08**

