|  |
| --- |
| **Horloge à palettes** |

Table des matières

[1. Présentation 3](#_Toc511651852)

[2. Liste du matériel 4](#_Toc511651853)

[2.1. Palettes 4](#_Toc511651854)

[2.2. Commande 4](#_Toc511651855)

[2.3. Outillage 5](#_Toc511651856)

[2.4. Logiciel 5](#_Toc511651857)

[2.5. Boîtier 6](#_Toc511651858)

[3. Montage 7](#_Toc511651859)

[3.1. Assemblage de base 7](#_Toc511651860)

[3.2. Câblage 7](#_Toc511651861)

[4. Manuel de l’utilisateur du logiciel 8](#_Toc511651862)

[4.1. Keypad 8](#_Toc511651863)

[4.2. Menu „Horloge“ 8](#_Toc511651864)

[4.3. Menu „Mise a l‘heure“ 8](#_Toc511651865)

[4.4. Menu „Calibration“ 8](#_Toc511651866)

[4.5. Menu „Pas a pas“ 9](#_Toc511651867)

[Annexe 1, code Arduino 10](#_Toc511651868)

[Annexe 2, Arduino LCD Keypad Shield 22](#_Toc511651869)

[Annexe 3, positions des feuillets sur les palettes 23](#_Toc511651870)

[Annexe 4, boîtier en bois 24](#_Toc511651871)

Table des illustrations

[Figure 1 : deux exemples de boîtier 6](#_Toc511650902)

[Figure 2 : Arduino avec shield LCD 7](#_Toc511650903)

[Figure 3 : schéma du câblage 7](#_Toc511650904)

[Figure 4 : Arduino Uno 22](#_Toc511650905)

[Figure 5 : LCD Keypad Shield 22](#_Toc511650906)

[Tableau 1 : hardware 5](#_Toc511651837)

[Tableau 2 : software 6](#_Toc511651838)

[Tableau 3 : positions des feuillets „heures“ 23](#_Toc511651839)

[Tableau 4 : positions des feuillets „minutes“ 23](#_Toc511651840)

[Tableau 5 : pièces de bois pour le boîtier 24](#_Toc511651841)

# Présentation

Ce document décrit la création d’une horloge à partir d‘anciens modules de panneaux d’affichage à palettes des CFF.

Le concept de l’horloge est une création d’un ancien collaborateur de la société OMEGA. Il a mis gracieusement à disposition le logiciel qu’il a écrit pour ce projet.

Il n’existe aucune garantie quant à l’exactitude du contenu de ce document ni de responsabilité en ce qui concerne le montage proposé. Aucun support n’est fourni.

La reproduction du montage décrit et/ou la copie du logiciel sont permises avec mention de la source et des conditions ci-dessus.

# Liste du matériel

## Palettes

Le montage nécessite des palettes du type incrémental. Il faut un module « heures » (40 feuillets) et un module « minutes » (62 feuillets). Ces modules existent en plusieurs tailles, avec ou sans l’interface enfichable.

## Commande

La commande permet d’une part de configurer les palettes (adressage et calibrage) et d’autre part de donner le signal horaire. Elle se compose des éléments suivants :

|  |  |  |
| --- | --- | --- |
| Description | Quantité | Image |
| Microcontrôleur Arduino Uno R3 | 1 |  |
| « Shield » écran LCD 2 x 16 avec boutons de commande pour Arduino | 1 |  |
| Module RTC DS3231 pour Arduino | 1 |  |
| Carte d’interface RS485 pour Arduino | 1 |  |
| Alimentation 7 à 12 V DC pour Arduino | 1 |  |
| Alimentation 24 V DC pour les palettes  **Il faut choisir une alimentation qui puisse être réglée pour augmenter la tension à 28 V !**  (par exemple Mean Well DR-15-24 0.63 A 15.2 W, article 1297354 chez Conrad) | 1 | http://www.conrad.ch/medias/global/ce/7000_7999/7600/7680/7681/1297355_RB_00_FB.EPS_1000.jpg |
| Plaque d’essai | 1 |  |
| Câbles, fiche 220V, etc. | Divers | http://www.matceram.ch/catalogues/fiches/3p10%20A.jpg |

Tableau 1 : hardware

Tout le matériel est disponible dans les magasins spécialisés ou sur internet (miniinthebox.com ou aliexpress.com par ex.)

## Outillage

Quelques outils sont utiles comme un câble USB, un tournevis, une pince à dénuder, un fer à souder, etc.

## Logiciel

Pour gérer le microcontrôleur Arduino, il faut installer sur un PC l’outil de développement depuis internet ainsi que le pilote de port USB si nécessaire.

Le code peut ensuite être chargé sur l’Arduino.

|  |  |
| --- | --- |
| IDE  www.arduino.cc |  |
| Pilote pour port USB selon le type  par ex. CH341SER  www.driverscape.com/download/usb-serial-ch341 |  |
| Logiciel de commande  Voir annexe 1 |  |

Tableau 2 : software

## Boîtier

Chacun est libre de créer le boîtier qu’il souhaite pour son horloge. Il est utile de prévoir une ouverture facile pour effectuer des réparations. L’écran LCD et ses boutons doivent être accessibles pour pouvoir régler l’heure et calibrer les palettes.

Figure 1 : deux exemples de boîtier

# Montage

## Assemblage de base

Le shield écran est directement branché sur le microcontrôleur Arduino.

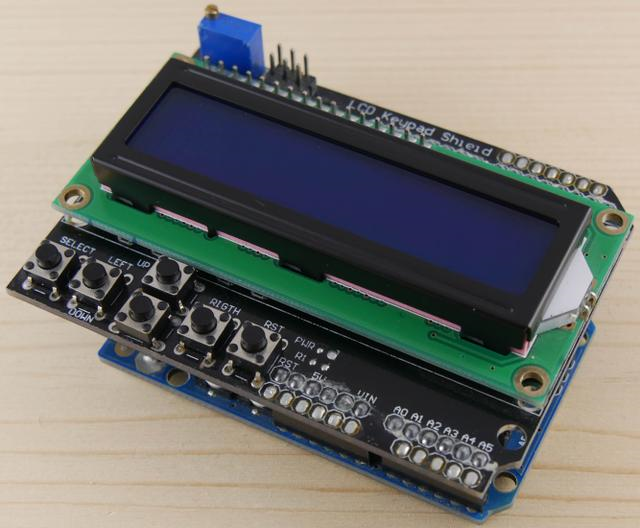


Figure 2 : Arduino avec shield LCD

## Câblage

Le câblage est réalisé par exemple en utilisant une plaque d’essai. Le schéma est le suivant :

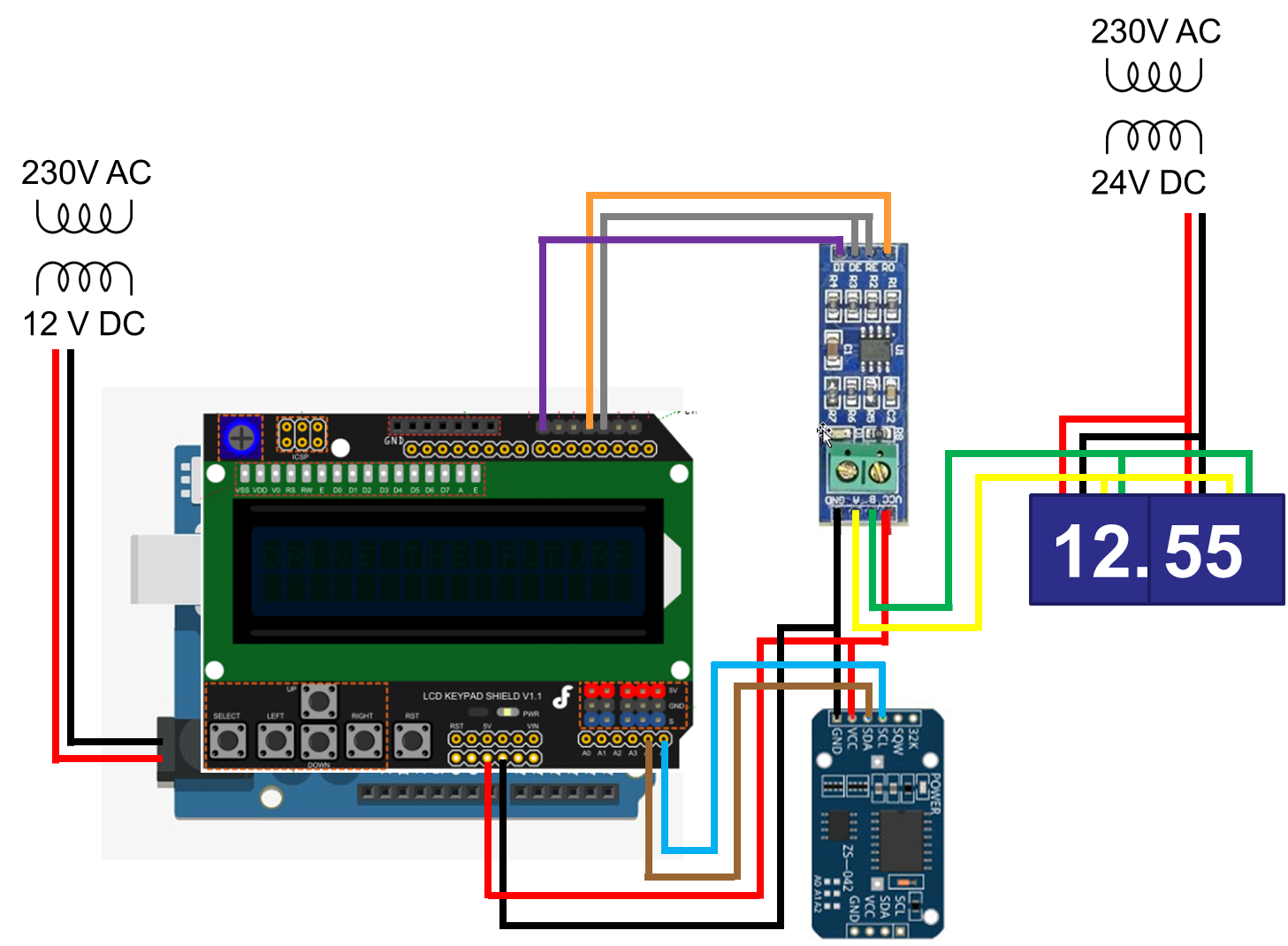


Figure 3 : schéma du câblage

Chargement du logiciel

Le logiciel est chargé sur l’Arduino à partir d’un PC. Le code se trouve dans l’annexe 1. Il suffit de le copier/coller dans l’outil de développement puis de le transférer.

Le montage est maintenant terminé, le dispositif peut être mis en service. L’alimentation de l’Arduino est aussi possible à partir du port USB.

# Manuel de l’utilisateur du logiciel

## Keypad

Pour activer les menus presser une première fois sur le bouton « SELECT » pour allumer le rétroéclairage de l’écran, une seconde fois pour entrer dans le mode « menus ».

Les boutons « UP » et « DOWN » permettent de naviguer dans les menus, successivement « Horloge », « Mise a l’heure », « Calibration » et « Pas a pas ».

Le bouton « SELECT » valide les données affichées.

Le bouton « RST » (reset) quitte le mode menu et relance le logiciel. Le rétroéclairage de l’écran est éteint et le mode « horloge » démarre automatiquement.

## Menu „Horloge“

La commande de l’horloge est enclenchée. Ce mode est automatiquement activé au démarrage de l’Arduino ou après l’usage du bouton « RST ».

## Menu „Mise a l‘heure“

Affiche la valeur actuelle de l’horloge (provenant du module RTC) et permet la mise à l’heure. Le format du message RTC est le suivant :

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 7 | 0 | 2 | 1 | 5 | 3 | 1 | 0 | 2 | 5 | 4 | 8 |
| Y | Y | M | M | D | D | w | h | h | m | m | s | s |

YY = année (de 00 à 99)

MM = mois (de 01 à 12)

DD = jour (de 01 à 31)

w = jour de la semaine (1 = lundi, 2 = mardi, … 7 = dimanche)

hh = heures (de 00 à 23)

mm = minutes (de 00 à 59)

ss = secondes (de 00 à 59)

Pour la mise à l’heure de l’horloge :

* placer le curseur (« \_ ») sous le chiffre à modifier avec les boutons « LEFT » et « RIGHT »
* choisir la valeur avec les boutons « UP » et « DOWN »
* répéter l’opération pour chaque position à modifier
* valider avec le bouton « SELECT »

## Menu „Calibration“

Cette fonction permet de configurer les palettes et de régler le défilement. Pour la configuration initiale il ne faut brancher qu’un seul des modules de palettes !

Les adresses des modules sont « 001 » pour les heures et « 002 » pour les minutes.

Le bouton « down » permet de faire avancer le moteur pas à pas, l’actionner jusqu’à la chute du feuillet. Le bouton « right » fait tourner la palette, l’utiliser pour afficher une valeur afin de passer à l’étape suivante.

Le type de palette est « 1 » pour le module à 40 feuillets (heures) et « 2 » pour celui à 62 feuillets (minutes).

La position du feuillet visible est à sélectionner en fonction des deux tables de l’annexe 4.

## Menu „Pas a pas“

Déclenche une rotation automatique des palettes, sans tenir compte de la configuration de l’adresse ou du type de celles-ci. La vitesse de défilement (en millisecondes) est modifiée avec les boutons « UP » et « DOWN ». Cette fonction est utile pour contrôler le bon fonctionnement du montage.

# Annexe 1, code Arduino

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* F L A P S C L O C K \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* Menu \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 00 MENU

// 01 HORLOGE

// 02 MISE A L'HEURE

// 03 CALIBRATION

// 04 PAS A PAS

int backlightMode = 0;

int menuMode = 0;

int smenuMode = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* Affichage LCD \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <LiquidCrystal.h> //Pour gestion du LCD1602

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

int backLight = 10; // pin 10 will control the backlight

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* Clavier \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define btnRIGHT 0

#define btnUP 1

#define btnDOWN 2

#define btnLEFT 3

#define btnSELECT 4

#define btnNONE 5

int lcd\_key = 0;

int adc\_key\_in = 0;

int keyboard\_key = 0;

int i\_key = 0;

int itmp = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* Horloge \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// format XHMSSXHMMSSXHHMMSSX

// nb.pos. 4464444444444444444

// Adresse 0000000000111111111

// Adresse 0123456789012345678

#include <Wire.h> //Pour gestion du RT C 3231

#define CLOCK\_ADDRESS 0x68

bool Century = true;

bool h12;

bool PM;

byte ADay, AHour, AMinute, ASecond, ABits;

byte yy, mo, hh, mm, ss, ss1, ss2, dow, dow1, dow2, dt, cinq;

byte pos1, pos2, pos3, pos4, pos5, pos6, pos7, pos8, pos9;

byte pos10, pos11, pos12, pos13, pos14, pos15, pos16, pos17, pos18, pos19;

byte pos20, pos21, pos22, pos23, pos24, pos25, pos26, pos27, pos28;

bool ADy, A12h, Apm;

int led = 13;

String ss01 = "";

String mm1 = "";

String hh1 = "";

String mo1 = "";

String dt1 = "";

String timeSetup = "";

String timeSetupD = "";

String timeSetupF = "";

String timeSetupX = "";

int posTimeSetup = 0;

int iitmp;

byte Year;

byte Month;

byte Date;

byte Day;

byte DoW;

byte Hour;

byte Minute;

byte Second;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* RS485 \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <SoftwareSerial.h>

SoftwareSerial mySerial(3, 13); // RX, TX

#define RS485\_control 2 //Pin 2 RS485 direction control

#define RS485Transmit HIGH

#define RS485Receive LOW

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* Protocole \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

byte CMD\_break[] = {

0x00

};

byte texte1[] = {

0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A

};

byte header[] = {

0xFF, 0xC0, 0x00

};

char readbuffer[] = {

0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

byte CMD\_r\_adr[] = {

0xFF, 0x9E, 0x7B

};

byte CMD\_w\_adr[] = {

0xFF, 0x8E, 0x00, 0x7B

};

byte CMD\_r\_typ[] = {

0xFF, 0x9D, 0x8C

};

byte CMD\_w\_typ[] = {

0xFF, 0x8D, 0x00, 0x01

};

byte CMD\_w\_step[] = {

0xFF, 0x86, 0x7B, 0x00

};

byte CMD\_w\_puls[] = {

0xFF, 0x87, 0x7B, 0x00

};

byte CMD\_w\_rotation[] = {

0xFF, 0x8A, 0x01

};

byte CMD\_w\_cal[] = {

0xFF, 0x8B, 0xAD, 0x00

};

byte CMD\_w\_step\_all[] = {

0xFF, 0xA6, 0x00, 0x00

};

int readchar;

//char readchar;

int incomingByte = 0;

int adrFlap = -1;

int adrFlapnew = -1;

int cen = -1;

int diz = -1;

int uni = -1;

int typFlap = -1;

int typFlaptmp = -1;

int curpos = -1;

int posFlap = -1;

int nbPuls = -1;

int cnt\_step = -1;

int cnt\_step1 = -1;

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXX START APPLICATION XXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

void setup()

{

backlightMode = 0; // Flag LCD backlight off

menuMode = 1; // Flag Menu HORLOGE

smenuMode = 0; // Flag sousMenu

pinMode(RS485\_control, OUTPUT);

Serial.begin(19200);

mySerial.begin(19200);

lcd.begin(16, 2); // start the library

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("HORLOGE"); // print a simple message

pinMode(backLight, OUTPUT); // LCD backlight off

digitalWrite(backLight, LOW);

// Start the I2C interface

Wire.begin();

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXX void loop XXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

void loop()

{

/////////////////////////////////////////////////////////// Lecture Clavier /////////////////////////////////

keyboard\_key = read\_keyboard();

/////////////////////////////////////////////////////////// LCD Backlight ON /////////////////////////////////

if (backlightMode == 0)

{

if (keyboard\_key == btnSELECT)

{

digitalWrite(backLight, HIGH);

backlightMode = 1;

keyboard\_key = btnNONE;

}

}

else //if (backlightMode == 0)

{

/////////////////////////////////////////////////////////// MENU /////////////////////////////////

if (menuMode == 0) //Menu

{

if (keyboard\_key == btnSELECT)

{

menuMode = smenuMode;

smenuMode = 0;

keyboard\_key == btnNONE;

}

lcd.noCursor();

// lcd.clear();

lcd.setCursor(0, 0);

lcd.print("MENU ");

if (keyboard\_key == btnDOWN)

{

smenuMode = smenuMode + 1;

if (smenuMode > 4) smenuMode = 1;

}

if (keyboard\_key == btnUP)

{

smenuMode = smenuMode - 1;

if (smenuMode == 0) smenuMode = 4;

}

lcd.setCursor(0, 1);

if (smenuMode == 1) lcd.print("Horloge ? ");

if (smenuMode == 2) lcd.print("Mise a l'heure ?");

if (smenuMode == 3) lcd.print("Calibration ? ");

if (smenuMode == 4) lcd.print("Pas a pas ? ");

keyboard\_key = btnNONE;

}

/////////////////////////////////////////////////////////// MISE A L'HEURE ///////////////////////////

if (menuMode == 2) //Mode MISE A L'HEURE

{

if (smenuMode == 0)

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("MISE A L'HEURE ");

lcd.cursor();

lcd.setCursor(0, 1);

lcd.print(timeSetup);

posTimeSetup = 8;

lcd.setCursor(posTimeSetup, 1);

smenuMode = 1;

}

if (smenuMode == 1)

{

switch (keyboard\_key) // depending on which button was pushed, we perform an action

{

case btnRIGHT:

{

if (posTimeSetup < 12) posTimeSetup++;

lcd.setCursor(posTimeSetup, 1);

break;

}

case btnLEFT:

{

if (posTimeSetup > 0) posTimeSetup--;

lcd.setCursor(posTimeSetup, 1);

break;

}

case btnUP:

{

lcd.setCursor(posTimeSetup, 1);

timeSetupD = timeSetup.substring(0, posTimeSetup);

timeSetupF = timeSetup.substring(posTimeSetup + 1);

timeSetupX = timeSetup.substring(posTimeSetup, posTimeSetup + 1);

itmp = timeSetupX.toInt();

if (itmp < 9) itmp++;

else itmp = 0;

timeSetupX = String(itmp);

timeSetup = timeSetupD + timeSetupX + timeSetupF;

lcd.setCursor(0, 1);

lcd.print(timeSetup);

lcd.setCursor(posTimeSetup, 1);

break;

}

case btnDOWN:

{

lcd.setCursor(posTimeSetup, 1);

timeSetupD = timeSetup.substring(0, posTimeSetup);

timeSetupF = timeSetup.substring(posTimeSetup + 1);

timeSetupX = timeSetup.substring(posTimeSetup, posTimeSetup + 1);

itmp = timeSetupX.toInt();

if (itmp > 0) itmp--;

else itmp = 9;

timeSetupX = String(itmp);

timeSetup = timeSetupD + timeSetupX + timeSetupF;

lcd.setCursor(0, 1);

lcd.print(timeSetup);

lcd.setCursor(posTimeSetup, 1);

break;

}

case btnSELECT:

{

GetDateStuff();

setYear(Year);

setMonth(Month);

setDate(Day);

setDoW(DoW);

setHour(Hour);

setMinute(Minute);

setSecond(Second);

menuMode = 1;

smenuMode = 0;

delay (10);

keyboard\_key = btnNONE;

break;

}

} //end switch

}

}

/////////////////////////////////////////////////////////// CALIBRATION ///////////////////////////

if (menuMode == 3) //Mode Calibration

{

if (smenuMode == 0) //Mettre les cables

{

lcd.clear();

lcd.print("CALIBRATION");

lcd.setCursor(0, 1);

lcd.print("Connect Flap ? ");

smenuMode = 1;

}

if (smenuMode == 1) //Mettre les cables

{

if (keyboard\_key == btnSELECT)

{

smenuMode = 2;

keyboard\_key == btnNONE;

}

}

if (smenuMode == 2) //Cherche adresse

{

lcd.setCursor(0, 1);

lcd.print("Cherche adresse?");

for (int i = 0; i < 256; i++)

{

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

lcd.print("Cherche adr: ");

lcd.print(i);

mySerial.read();

mySerial.read();

CMD\_r\_adr[2] = i;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_r\_adr, 3);

digitalWrite(RS485\_control, RS485Receive);

delay (10);

if (mySerial.available() > 0)

{

// mySerial.read();

readchar = mySerial.read();

mySerial.read();

incomingByte = readchar;

if (incomingByte != -1) i = 256;

delay (10);

}

}

adrFlap = incomingByte;

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

lcd.print("adr=");

lcd.print(adrFlap);

if (adrFlap == -1)

{

lcd.setCursor(0, 1);

lcd.print("adr = NOT FOUND ");

smenuMode = 0;

delay (2000);

}

else

{

smenuMode = 3;

delay (1000);

}

}

if (smenuMode == 3) //Cherche Type

{

lcd.setCursor(0, 1);

lcd.print("Cherche type ");

delay (1000);

// i\_key = wait\_select();

mySerial.read();

mySerial.read();

CMD\_r\_typ[2] = adrFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_r\_typ, 3);

digitalWrite(RS485\_control, RS485Receive);

delay (5);

if (mySerial.available() > 0)

{

// mySerial.read();

readchar = mySerial.read();

mySerial.read();

incomingByte = readchar;

delay (5);

}

typFlap = incomingByte;

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

lcd.print("adr=");

lcd.print(adrFlap);

lcd.setCursor(8, 1);

lcd.print("type=");

lcd.print(typFlap);

delay (1000);

lcd.setCursor(0, 1);

lcd.print("Rotation ");

// i\_key = wait\_select();

CMD\_w\_rotation[2] = adrFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_rotation, 3);

delay (5000);

smenuMode = 4;

}

if (smenuMode == 4) //Modification et écriture adresse et type

{

// Edit ad=123;ty=1

// 0123456789012345

itmp = adr\_to\_cdu();

lcd.setCursor(0, 1);

lcd.print("Edit ad=");

lcd.print(cen);

lcd.print(diz);

lcd.print(uni);

lcd.setCursor(11, 1);

lcd.print(",ty=");

typFlaptmp = typFlap;

if (typFlap == 65) typFlaptmp = 4;

if (typFlap == 66) typFlaptmp = 6;

if (typFlap == 67) typFlaptmp = 8;

lcd.print(typFlaptmp);

lcd.cursor();

curpos = 8;

lcd.setCursor(curpos, 1);

itmp = edit();

typFlap = typFlaptmp;

if (typFlaptmp == 4) typFlap = 65;

if (typFlaptmp == 6) typFlap = 66;

if (typFlaptmp == 8) typFlap = 67;

adrFlapnew = (cen \* 100) + (diz \* 10) + (uni);

CMD\_w\_adr[2] = adrFlap;

CMD\_w\_adr[3] = adrFlapnew;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_adr, 4);

adrFlap = adrFlapnew;

CMD\_w\_typ[2] = adrFlap;

CMD\_w\_typ[3] = typFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_typ, 4);

smenuMode = 5;

delay (1000);

}

if (smenuMode == 5) //Calibration

{

// Position = 00 /

// 0123456789012345/

// RI=Step; DO=Puls/

lcd.setCursor(0, 1);

lcd.print("RI=Step; DO=Puls");

diz = 0;

uni = 0,

itmp = editpos();

smenuMode = 6;

delay (1000);

}

if (smenuMode == 6) //Test le nombre de puls

{

// Tst Puls 8-12 0 /

// 0123456789012345/

lcd.setCursor(0, 1);

if (typFlap == 0) lcd.print("Presser SELECT ");

if (typFlap == 1) lcd.print("Tst Puls 8-12 0 ");

if (typFlap == 65) lcd.print("Tst Puls 8-12 0 ");

if (typFlap == 2) lcd.print("Tst Puls 7-11 0 ");

if (typFlap == 66) lcd.print("Tst Puls 7-11 0 ");

if (typFlap == 3) lcd.print("Tst Puls 6-10 0 ");

if (typFlap == 67) lcd.print("Tst Puls 6-10 0 ");

nbPuls = 0;

keyboard\_key = btnNONE;

while (keyboard\_key != btnSELECT)

{

if (keyboard\_key == btnDOWN)

{

nbPuls = nbPuls + 1;

if (nbPuls > 16)

{

nbPuls = 0;

lcd.setCursor(14, 1);

lcd.print("0 ");

}

lcd.setCursor(14, 1);

lcd.print(nbPuls);

Serial.println("PULS");

CMD\_w\_puls[2] = adrFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_puls, 4);

}

keyboard\_key = read\_keyboard();

delay(2);

}

delay (1000);

lcd.setCursor(0, 1);

lcd.print("FIN CALIBRATION ");

menuMode = 0;

smenuMode = 1;

delay (2000);

}

}

/////////////////////////////////////////////////////////// PAS A PAS ///////////////////////////

if (menuMode == 4) //Mode Calibration

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("PAS A PAS ");

cnt\_step = 10;

cnt\_step1 = 10;

keyboard\_key = btnNONE;

while (keyboard\_key != btnSELECT)

{

// if (keyboard\_key == btnPREV)

// {

// break;

// }

if (keyboard\_key == btnUP)

{

cnt\_step = cnt\_step + 1;

if (cnt\_step > 50) cnt\_step = 50;

}

if (keyboard\_key == btnDOWN)

{

cnt\_step = cnt\_step - 1;

if (cnt\_step < 1) cnt\_step = 1;

}

cnt\_step1 = cnt\_step1 - 1;

if (cnt\_step1 == 0)

{

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_step\_all, 4);

cnt\_step1 = cnt\_step;

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 1);

lcd.print(cnt\_step \* 100);

}

delay (100);

keyboard\_key = read\_keyboard();

} //end while

menuMode = 0;

smenuMode = 0;

}

delay (200);

//}

} // end if (backlightMode == 0)

/////////////////////////////////////////////////////////// HORLOGE ///////////////////////////////

if (menuMode == 1) //Mode Horloge

{

if (keyboard\_key == btnSELECT)

{

menuMode = 0;

smenuMode = 1;

}

else

{

lcd.noCursor();

lcd.setCursor(0, 0);

lcd.print("HORLOGE ");

lcd.setCursor(0, 1);

yy = getYear();

lcd.print(yy);

timeSetup = String(yy);

mo = getMonth(Century);

mo1 = String (mo);

if (mo <= 9)

{

mo1 = "0";

mo1 += mo;

}

lcd.print(mo1);

timeSetup += mo1;

dt = getDate();

dt1 = String (dt);

if (dt <= 9)

{

dt1 = "0";

dt1 += dt;

}

lcd.print(dt1);

timeSetup += dt1;

dow = getDoW();

lcd.print(dow);

timeSetup += String(dow);

hh = getHour(h12, PM);

hh1 = String (hh);

if (hh <= 9)

{

hh1 = "0";

hh1 += hh;

}

lcd.print(hh1);

timeSetup += hh1;

mm = getMinute();

mm1 = String (mm);

if (mm <= 9)

{

mm1 = "0";

mm1 += mm;

}

lcd.print(mm1);

timeSetup += mm1;

ss = getSecond();

ss01 = String (ss);

if (ss <= 9)

{

ss01 = "0";

ss01 += ss;

}

lcd.print(ss01);

timeSetup += ss01;

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\* MESSAGE Horloge \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//XHMSSXHMMSSXHHMMSSXHHMMSSX

// HMSS

// HMMSS

// HHMMSS

// HHMMSSX

//01234567890123456789012345

// 4644 44444 444444 666666

//Heure

pos1 = hh;

pos6 = pos1;

pos12 = posHeure1(hh);

pos13 = posHeure2(hh);

pos19 = posHeure16(hh);

pos20 = posHeure26(hh);

//Minute

pos2 = posMinute(mm);

pos7 = posMinute1(mm);

pos8 = posMinute2(mm);

pos14 = pos7;

pos15 = pos8;

pos21 = posMinute16(mm);

pos22 = posMinute26(mm);

// <25s DoW; <50 Date; >50 ss

if (ss < 25)

{

pos3 = posDoW1(dow);

pos4 = posDoW2(dow);

pos9 = pos3;

pos10 = pos4;

pos16 = pos3;

pos17 = pos4;

pos23 = posDoW16(dow);

pos24 = posDoW26(dow);

}

else

{

pos3 = posDate1(dt);

pos4 = posDate2(dt);

pos9 = pos3;

pos10 = pos4;

pos16 = pos3;

pos17 = pos4;

pos23 = posDate16(dt);

pos24 = posDate26(dt);

}

if (ss > 50)

{

pos3 = 20;

pos3 = posSec1(ss);

pos4 = posSec2(ss);

pos9 = pos3;

pos10 = pos4;

pos16 = pos3;

pos17 = pos4;

pos23 = posSec16(ss);

pos24 = posSec26(ss);

}

texte1[0] = 0;

texte1[1] = pos1;

texte1[2] = pos2;

texte1[3] = pos3;

texte1[4] = pos4;

texte1[5] = 0;

texte1[6] = pos6;

texte1[7] = pos7;

texte1[8] = pos8;

texte1[9] = pos9;

texte1[10] = pos10;

texte1[11] = 0;

texte1[12] = pos12;

texte1[13] = pos13;

texte1[14] = pos14;

texte1[15] = pos15;

texte1[16] = pos16;

texte1[17] = pos17;

texte1[18] = 0;

texte1[19] = pos19;

texte1[20] = pos20;

texte1[21] = pos21;

texte1[22] = pos22;

texte1[23] = pos23;

texte1[24] = pos24;

texte1[25] = 0;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

//header

mySerial.write(header, 3);

//texte1

mySerial.write(texte1, 26);

if (ss > 50 || ss == 0)

{

delay(30);

}

else

{

delay(100);

}

}

}

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

// FONCTION read the keyboard

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

int read\_keyboard()

{

adc\_key\_in = analogRead(0);

// lcd.clear();

// lcd.setCursor(0, 0);

// lcd.print(adc\_key\_in);

// delay (100);

if (adc\_key\_in > 1000) i\_key = btnNONE;

if ((adc\_key\_in < 1000) && (adc\_key\_in > 500)) i\_key = btnSELECT;

if ((adc\_key\_in < 500) && (adc\_key\_in > 320)) i\_key = btnLEFT;

if ((adc\_key\_in < 320) && (adc\_key\_in > 200)) i\_key = btnDOWN;

if ((adc\_key\_in < 200) && (adc\_key\_in > 50)) i\_key = btnUP;

if ((adc\_key\_in < 70) && (adc\_key\_in >= 0)) i\_key = btnRIGHT;

if (i\_key != btnNONE)

{

while (analogRead(0) < 1000) delay(2);

}

return i\_key;

}

//XXXXXXXXXXXXXXXXXXXXX

// FONCTION wait\_select

//XXXXXXXXXXXXXXXXXXXXX

int wait\_select()

{

keyboard\_key = btnNONE;

while (keyboard\_key != btnSELECT)

{

keyboard\_key = read\_keyboard();

delay(2);

}

return keyboard\_key;

}

//XXXXXXXXXXXXXXXXXXXX

// FONCTION adr\_to\_cdu

//XXXXXXXXXXXXXXXXXXXX

int adr\_to\_cdu()

{

cen = 2;

if (adrFlap < 200) cen = 1;

if (adrFlap < 100) cen = 0;

int adrFlaptmp = adrFlap - (cen \* 100);

diz = 9;

if (adrFlaptmp < 90) diz = 8;

if (adrFlaptmp < 80) diz = 7;

if (adrFlaptmp < 70) diz = 6;

if (adrFlaptmp < 60) diz = 5;

if (adrFlaptmp < 50) diz = 4;

if (adrFlaptmp < 40) diz = 3;

if (adrFlaptmp < 30) diz = 2;

if (adrFlaptmp < 20) diz = 1;

if (adrFlaptmp < 10) diz = 0;

uni = adrFlaptmp - (diz \* 10);

return 1;

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

// FONCTION edit (Fonctions autorisées pour sortir SELECT ou RESET)

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

int edit()

{

keyboard\_key = btnNONE;

while (keyboard\_key != btnSELECT)

{

if (keyboard\_key == btnUP)

{

if (curpos == 8)

{

cen = cen + 1;

if (cen > 2) cen = 0;

}

if (curpos == 9)

{

diz = diz + 1;

if (diz > 9) diz = 0;

}

if (curpos == 10)

{

uni = uni + 1;

if (uni > 9) uni = 0;

}

if (curpos == 15)

{

typFlaptmp = typFlaptmp + 1;

if (typFlaptmp == 5) typFlaptmp = 6; // 0=lamp;1=40;2=62;3=80;4=40L;6=60L; 8=80L

if (typFlaptmp == 7) typFlaptmp = 8;

if (typFlaptmp > 8) typFlaptmp = 0;

}

if (curpos == 8) lcd.print(cen);

if (curpos == 9) lcd.print(diz);

if (curpos == 10) lcd.print(uni);

if (curpos == 15) lcd.print(typFlaptmp);

} // fin UP

if (keyboard\_key == btnRIGHT)

{

if (curpos == 15) curpos = 8;

else if (curpos == 8) curpos = 9;

else if (curpos == 9) curpos = 10;

else if (curpos == 10) curpos = 15;

}

lcd.setCursor(curpos, 1);

keyboard\_key = read\_keyboard();

delay(2);

}

return 1;

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

// FONCTION editpos (Fonctions autorisées RIGHT=Step; DOWN=Puls / SELECT ou RESET)

//XXXXXXXXXXXXXXXXXXXXXXXXXXX

int editpos()

{

if (mySerial.available() > 0)

{

mySerial.read();

readchar = mySerial.read();

mySerial.read();

delay (10);

}

keyboard\_key = btnNONE;

while (keyboard\_key != btnSELECT)

{

if (keyboard\_key == btnRIGHT) //Step

{

Serial.println("STEP");

CMD\_w\_step[2] = adrFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_step, 4);

} // fin btnRIGHT

if (keyboard\_key == btnDOWN) //Puls

{

Serial.println("PULS");

CMD\_w\_puls[2] = adrFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_puls, 4);

} // fin btnDOWN

keyboard\_key = read\_keyboard();

delay(2);

} //end while

// Position = 00 /

// 0123456789012345/

lcd.setCursor(0, 1);

lcd.print("Position = 00 ");

curpos = 11;

diz = 0;

uni = 0;

keyboard\_key = btnNONE;

lcd.cursor();

lcd.setCursor(curpos, 1);

while (keyboard\_key != btnSELECT)

{

if (keyboard\_key == btnUP)

{

if (curpos == 11)

{

diz = diz + 1;

if (typFlap == 0)

{

if (diz > 2) diz = 0;

}

if (typFlap == 1)

{

if (diz > 4) diz = 0;

}

if (typFlap == 2)

{

if (diz > 6) diz = 0;

}

if (typFlap == 3)

{

if (diz > 8) diz = 0;

}

}

if (curpos == 12)

{

uni = uni + 1;

if (uni > 9) uni = 0;

}

if (curpos == 11) lcd.print(diz);

if (curpos == 12) lcd.print(uni);

} // fin UP

if (keyboard\_key == btnRIGHT)

{

if (curpos == 12) curpos = 11;

else if (curpos == 11) curpos = 12;

}

lcd.setCursor(curpos, 1); // move cursor to second line "1" and 9 spaces over

keyboard\_key = read\_keyboard();

delay(2);

} //end while

posFlap = (10 \* diz) + uni;

CMD\_w\_cal[2] = adrFlap;

CMD\_w\_cal[3] = posFlap;

digitalWrite(RS485\_control, RS485Transmit);

itmp = gen\_break();

mySerial.write(CMD\_w\_cal, 4);

digitalWrite(RS485\_control, RS485Receive);

delay (10);

if (mySerial.available() > 0)

{

mySerial.read();

readchar = mySerial.read();

mySerial.read();

delay (10);

}

return 1;

}

///////////////////////////////////////////////////////////FONCTIONS HORLOGE ///////////////////////////////

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

// FONCTIONS RTC DS3231

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

byte getYear() {

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x06);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return bcdToDec(Wire.read());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getMonth(bool& Century) {

byte temp\_buffer;

byte hour;

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x05);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

temp\_buffer = Wire.read();

Century = temp\_buffer & 0b10000000;

return (bcdToDec(temp\_buffer & 0b01111111)) ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getDate() {

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x04);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return bcdToDec(Wire.read());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getDoW() {

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x03);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return bcdToDec(Wire.read());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getHour(bool& h12, bool& PM) {

byte temp\_buffer;

byte hour;

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x02);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

temp\_buffer = Wire.read();

h12 = temp\_buffer & 0b01000000;

if (h12) {

PM = temp\_buffer & 0b00100000;

hour = bcdToDec(temp\_buffer & 0b00011111);

}

else {

hour = bcdToDec(temp\_buffer & 0b00111111);

}

return hour;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getMinute() {

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x01);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return bcdToDec(Wire.read());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte getSecond() {

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x00);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return bcdToDec(Wire.read());

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte bcdToDec(byte val) {

// Convert binary coded decimal to normal decimal numbers

return ( (val / 16 \* 10) + (val % 16) );

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

// FONCTIONS Set RTC DS3231

//SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setYear(byte Year) {

// Sets the year

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x06);

Wire.write(decToBcd(Year));

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setMonth(byte Month) {

// Sets the month

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x05);

Wire.write(decToBcd(Month));

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setDate(byte Date) {

// Sets the Date

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x04);

Wire.write(decToBcd(Date));

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setDoW(byte DoW) {

// Sets the Day of Week

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x03);

Wire.write(decToBcd(DoW));

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setHour(byte Hour) {

// Sets the hour, without changing 12/24h mode.

// The hour must be in 24h format.

bool h12;

// Start by figuring out what the 12/24 mode is

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x02);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

h12 = (Wire.read() & 0b01000000);

// if h12 is true, it's 12h mode; false is 24h.

if (h12) {

// 12 hour

if (Hour > 12) {

Hour = decToBcd(Hour - 12) | 0b01100000;

}

else {

Hour = decToBcd(Hour) & 0b11011111;

}

}

else {

// 24 hour

Hour = decToBcd(Hour) & 0b10111111;

}

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x02);

Wire.write(Hour);

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setMinute(byte Minute) {

// Sets the minutes

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x01);

Wire.write(decToBcd(Minute));

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setSecond(byte Second) {

// Sets the seconds

// This function also resets the Oscillator Stop Flag, which is set

// whenever power is interrupted.

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x00);

Wire.write(decToBcd(Second));

Wire.endTransmission();

// Clear OSF flag

byte temp\_buffer = readControlByte(1);

writeControlByte((temp\_buffer & 0b01111111), 1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setClockMode(bool h12) {

// sets the mode to 12-hour (true) or 24-hour (false).

// One thing that bothers me about how I've written this is that

// if the read and right happen at the right hourly millisecnd,

// the clock will be set back an hour. Not sure how to do it better,

// though, and as long as one doesn't set the mode frequently it's

// a very minimal risk.

// It's zero risk if you call this BEFORE setting the hour, since

// the setHour() function doesn't change this mode.

byte temp\_buffer;

// Start by reading byte 0x02.

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x02);

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

temp\_buffer = Wire.read();

// Set the flag to the requested value:

if (h12) {

temp\_buffer = temp\_buffer | 0b01000000;

}

else {

temp\_buffer = temp\_buffer & 0b10111111;

}

// Write the byte

Wire.beginTransmission(CLOCK\_ADDRESS);

Wire.write(0x02);

Wire.write(temp\_buffer);

Wire.endTransmission();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte decToBcd(byte val) {

// Convert normal decimal numbers to binary coded decimal

return ( (val / 10 \* 16) + (val % 10) );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte readControlByte(bool which) {

// Read selected control byte

// first byte (0) is 0x0e, second (1) is 0x0f

Wire.beginTransmission(CLOCK\_ADDRESS);

if (which) {

// second control byte

Wire.write(0x0f);

}

else {

// first control byte

Wire.write(0x0e);

}

Wire.endTransmission();

Wire.requestFrom(CLOCK\_ADDRESS, 1);

return Wire.read();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void writeControlByte(byte control, bool which) {

// Write the selected control byte.

// which=false -> 0x0e, true->0x0f.

Wire.beginTransmission(CLOCK\_ADDRESS);

if (which) {

Wire.write(0x0f);

}

else {

Wire.write(0x0e);

}

Wire.write(control);

Wire.endTransmission();

}

//SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

//SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

void GetDateStuff()

{

// the order YYMMDDwHHMMSS, with an 'x' at the end.

byte Temp1, Temp2;

char InString[20];

byte j = 0;

int i = 0;

int itmp1;

String stmp;

timeSetup.toCharArray(InString, 19);

Serial.println(InString);

// Read Year first

Temp1 = (byte)InString[0] - 48;

Temp2 = (byte)InString[1] - 48;

Year = Temp1 \* 10 + Temp2;

// now month

Temp1 = (byte)InString[2] - 48;

Temp2 = (byte)InString[3] - 48;

Month = Temp1 \* 10 + Temp2;

// now date

Temp1 = (byte)InString[4] - 48;

Temp2 = (byte)InString[5] - 48;

Day = Temp1 \* 10 + Temp2;

// now Day of Week

DoW = (byte)InString[6] - 48;

// now Hour

Temp1 = (byte)InString[7] - 48;

Temp2 = (byte)InString[8] - 48;

Hour = Temp1 \* 10 + Temp2;

// now Minute

Temp1 = (byte)InString[9] - 48;

Temp2 = (byte)InString[10] - 48;

Minute = Temp1 \* 10 + Temp2;

// now Second

Temp1 = (byte)InString[11] - 48;

Temp2 = (byte)InString[12] - 48;

Second = Temp1 \* 10 + Temp2;

}

///////////////////////////////////////////////////////////FONCTIONS AFFICHAGE PALETTES ////////////////////

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

// FONCTIONS pour l'affichage palettes

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Palette Minutes

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posMinute(byte val) {

if (val == 0) {

return (30);

}

else if (val == 1) {

return (31);

}

else if (val == 2) {

return (32);

}

else if (val == 3) {

return (33);

}

else if (val == 4) {

return (34);

}

else if (val == 5) {

return (35);

}

else if (val == 6) {

return (36);

}

else if (val == 7) {

return (37);

}

else if (val == 8) {

return (38);

}

else if (val == 9) {

return (39);

}

else if (val == 10) {

return (40);

}

else if (val == 11) {

return (41);

}

else if (val == 12) {

return (42);

}

else if (val == 13) {

return (43);

}

else if (val == 14) {

return (44);

}

else if (val == 15) {

return (45);

}

else if (val == 16) {

return (46);

}

else if (val == 17) {

return (47);

}

else if (val == 18) {

return (48);

}

else if (val == 19) {

return (49);

}

else if (val == 20) {

return (50);

}

else if (val == 21) {

return (51);

}

else if (val == 22) {

return (52);

}

else if (val == 23) {

return (53);

}

else if (val == 24) {

return (54);

}

else if (val == 25) {

return (55);

}

else if (val == 26) {

return (56);

}

else if (val == 27) {

return (57);

}

else if (val == 28) {

return (58);

}

else if (val == 29) {

return (59);

}

else if (val == 30) {

return (60);

}

else if (val == 31) {

return (0);

}

else if (val == 32) {

return (1);

}

else if (val == 33) {

return (2);

}

else if (val == 34) {

return (3);

}

else if (val == 35) {

return (4);

}

else if (val == 36) {

return (5);

}

else if (val == 37) {

return (6);

}

else if (val == 38) {

return (7);

}

else if (val == 39) {

return (8);

}

else if (val == 40) {

return (9);

}

else if (val == 41) {

return (10);

}

else if (val == 42) {

return (11);

}

else if (val == 43) {

return (12);

}

else if (val == 44) {

return (13);

}

else if (val == 45) {

return (14);

}

else if (val == 46) {

return (15);

}

else if (val == 47) {

return (16);

}

else if (val == 48) {

return (17);

}

else if (val == 49) {

return (18);

}

else if (val == 50) {

return (19);

}

else if (val == 51) {

return (20);

}

else if (val == 52) {

return (21);

}

else if (val == 53) {

return (22);

}

else if (val == 54) {

return (23);

}

else if (val == 55) {

return (24);

}

else if (val == 56) {

return (25);

}

else if (val == 57) {

return (26);

}

else if (val == 58) {

return (27);

}

else if (val == 59) {

return (28);

}

return (61);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDoW1(byte val) {

if (val == 1) return (11);

else if (val == 2) return (12);

else if (val == 3) return (12);

else if (val == 4) return (9);

else if (val == 5) return (21);

else if (val == 6) return (18);

else if (val == 7) return (3);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDoW16(byte val) {

if (val == 1) return (41);

else if (val == 2) return (42);

else if (val == 3) return (42);

else if (val == 4) return (39);

else if (val == 5) return (51);

else if (val == 6) return (48);

else if (val == 7) return (33);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDoW2(byte val) {

if (val == 1) return (20);

else if (val == 2) return (0);

else if (val == 3) return (4);

else if (val == 4) return (4);

else if (val == 5) return (4);

else if (val == 6) return (0);

else if (val == 7) return (8);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDoW26(byte val) {

if (val == 1) return (50);

else if (val == 2) return (30);

else if (val == 3) return (34);

else if (val == 4) return (34);

else if (val == 5) return (34);

else if (val == 6) return (30);

else if (val == 7) return (38);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDate1(byte val) {

if (val <= 9) return (39);

if (val > 9 && val <= 19) return (28);

if (val > 19 && val <= 29) return (29);

if (val > 29 && val <= 39) return (30);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDate16(byte val) {

if (val <= 9) return (0);

if (val > 9 && val <= 19) return (16);

if (val > 19 && val <= 29) return (17);

if (val > 29 && val <= 39) return (18);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDate2(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val == 0) return (37);

return (val + 27);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posDate26(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val == 0) return (15);

return (val + 15);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posHeure1(byte val) {

if (val <= 9) return (37);

if (val > 9 && val <= 19) return (28);

if (val > 19 && val <= 29) return (29);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posHeure2(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val == 0) return (37);

return (val + 27);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posHeure16(byte val) {

if (val <= 9) return (15);

if (val > 9 && val <= 19) return (16);

if (val > 19 && val <= 29) return (17);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posHeure26(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val == 0) return (15);

return (val + 15);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posMinute1(byte val) {

if (val <= 9) return (37);

if (val > 9 && val <= 19) return (28);

if (val > 19 && val <= 29) return (29);

if (val > 29 && val <= 39) return (30);

if (val > 39 && val <= 49) return (31);

if (val > 49 && val <= 59) return (32);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posMinute16(byte val) {

if (val <= 9) return (15);

if (val > 9 && val <= 19) return (16);

if (val > 19 && val <= 29) return (17);

if (val > 29 && val <= 39) return (18);

if (val > 39 && val <= 49) return (19);

if (val > 49 && val <= 59) return (20);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posMinute2(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val > 39 && val <= 49) val = val - 40;

if (val > 49 && val <= 59) val = val - 50;

if (val == 0) return (37);

return (val + 27);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posMinute26(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val > 39 && val <= 49) val = val - 40;

if (val > 49 && val <= 59) val = val - 50;

if (val == 0) return (15);

return (val + 15);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posSec1(byte val) {

if (val <= 9) return (37);

if (val > 9 && val <= 19) return (28);

if (val > 19 && val <= 29) return (29);

if (val > 29 && val <= 39) return (30);

if (val > 39 && val <= 49) return (31);

if (val > 49 && val <= 59) return (32);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posSec16(byte val) {

if (val <= 9) return (15);

if (val > 9 && val <= 19) return (16);

if (val > 19 && val <= 29) return (17);

if (val > 29 && val <= 39) return (18);

if (val > 39 && val <= 49) return (19);

if (val > 49 && val <= 59) return (20);

return (0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posSec2(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val > 39 && val <= 49) val = val - 40;

if (val > 49 && val <= 59) val = val - 50;

if (val == 0) return (37);

return (val + 27);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte posSec26(byte val) {

if (val > 9 && val <= 19) val = val - 10;

if (val > 19 && val <= 29) val = val - 20;

if (val > 29 && val <= 39) val = val - 30;

if (val > 39 && val <= 49) val = val - 40;

if (val > 49 && val <= 59) val = val - 50;

if (val == 0) return (15);

return (val + 15);

}

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXX

// FONCTION gen\_break

//XXXXXXXXXXXXXXXXXXX

int gen\_break()

{

mySerial.begin(300);

mySerial.write(CMD\_break, 1);

delay(10); // wait

mySerial.begin(19200);

delay(2); // wait

return 1;

}

# Annexe 2, Arduino et LCD Keypad Shield

L’Arduino se présente ainsi :

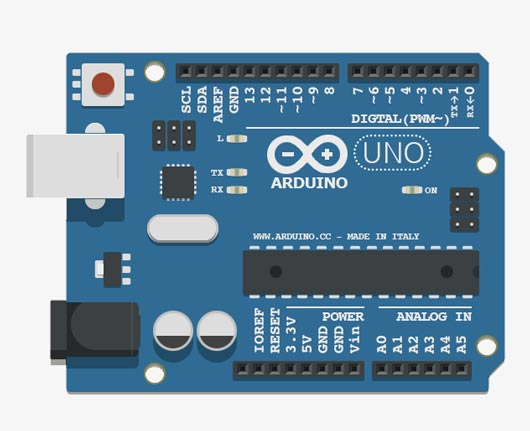


Figure 4 : Arduino Uno

Le microcontrôleur peut être alimenté soit par le port USB (depuis un PC ou un adaptateur), soit par la prise jack ( 7 à 12 volts DC, respecter la polarité). En cas d’alimentation par le port USB il est à relever que la puissance à disposition est limitée, ce qui ne joue toutefois pas de rôle pour l’emploi décrit dans ce document.

Les contacts du shield sont légèrement différents de ceux du module de base Arduino. Voici l’aperçu des contacts et leurs équivalents sur l’Arduino

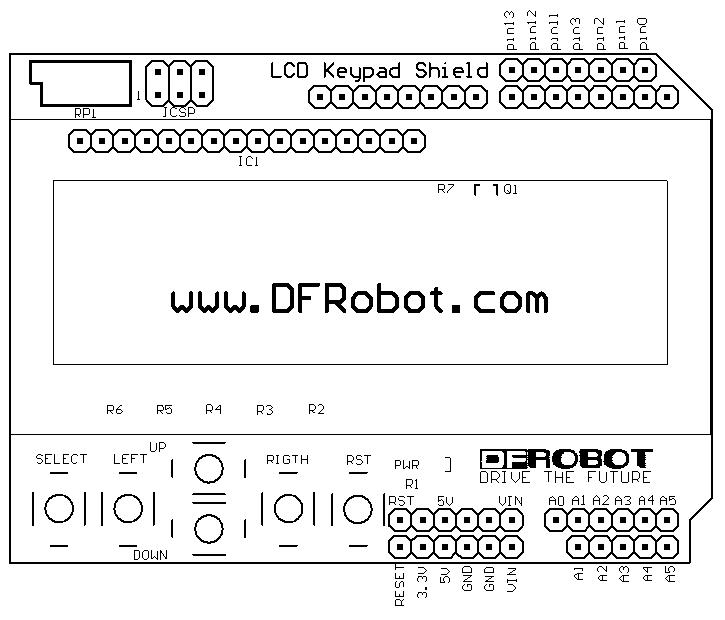


Figure 5 : LCD Keypad Shield

# Annexe 3, positions des feuillets sur les palettes

Pour la configuration des palettes il faut se référer aux tables suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Affichage | Position | Affichage | Position |
| 0. | 0 | 13. | 13 |
| 1. | 1 | 14. | 14 |
| 2. | 2 | 15. | 15 |
| 3. | 3 | 16. | 16 |
| 4. | 4 | 17. | 17 |
| 5. | 5 | 18. | 18 |
| 6. | 6 | 19. | 19 |
| 7. | 7 | 20. | 20 |
| 8. | 8 | 21. | 21 |
| 9. | 9 | 22. | 22 |
| 10. | 10 | 23. | 23 |
| 11. | 11 | « vide » | 24 à 38 |
| 12. | 12 | « initial » | 39 |

Tableau 3 : positions des feuillets „heures“

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Affichage | Position | Affichage | Position | Affichage | Position |
| 00 | 30 | 21 | 51 | 41 | 10 |
| 01 | 31 | 22 | 52 | 42 | 11 |
| 02 | 32 | 23 | 53 | 43 | 12 |
| 03 | 33 | 24 | 54 | 44 | 13 |
| 04 | 34 | 25 | 55 | 45 | 14 |
| 05 | 35 | 26 | 56 | 46 | 15 |
| 06 | 36 | 27 | 57 | 47 | 16 |
| 07 | 37 | 28 | 58 | 48 | 17 |
| 08 | 38 | 29 | 59 | 49 | 18 |
| 09 | 39 | 30 | 60 | 50 | 19 |
| 10 | 40 | « initial » | 61 | 51 | 20 |
| 11 | 41 | 31 | 0 | 52 | 21 |
| 12 | 42 | 32 | 1 | 53 | 22 |
| 13 | 43 | 33 | 2 | 54 | 23 |
| 14 | 44 | 34 | 3 | 55 | 24 |
| 15 | 45 | 35 | 4 | 56 | 25 |
| 16 | 46 | 36 | 5 | 57 | 26 |
| 17 | 47 | 37 | 6 | 58 | 27 |
| 18 | 48 | 38 | 7 | 59 | 28 |
| 19 | 49 | 39 | 8 | « vide » | 29 |
| 20 | 50 | 40 | 9 |  |  |

Tableau 4 : positions des feuillets „minutes“

# Annexe 4, boîtier en bois

La construction d’un boîtier en bois pour les modules à palettes du type « Grand » (cf figure 1 à droite) nécessite par exemple les planches suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Quantité | Longueur x largeur | Epaisseur | Emploi |
| 2 | 343 x 220 mm | 9 mm | Haut et bas |
| 2 | 142 x 220 mm | 9 mm | Parois latérales |
| 1 | 343 x 160 mm | 4 mm | Arrière |

Tableau 5 : pièces de bois pour le boîtier

Il faut fixer les palettes aux parois latérales en remplaçant les vis. Assembler alors les parois latérales et le fond. Fixer l’alimentation. Ajouter la face supérieure.

Prévoir un passage pour les câbles dans la parois arrière et y découper une ouverture pour l’écran et les boutons de commande, fixer la commande.

Le boîtier peut ensuite être fermé avec la parois arrière.

Le bleu CFF est la référence RAL 5002 « bleu outremer ».