

Horloge musicale

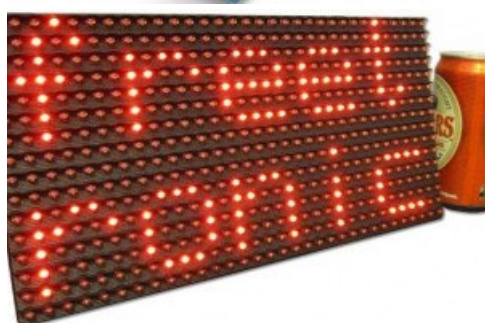
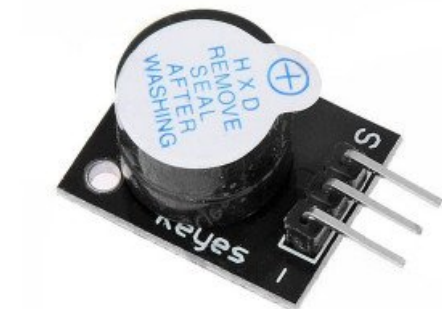
Auteur :
Grégory

Préambule

Le FabManager de La Fabrique avait besoin de mettre en place une horloge numérique affichant l'heure et prévenant les usagers de la fermeture prochaine du lieu, avec rappel toutes les minutes jusqu'à l'heure de fermeture. Pour ce faire, nous avons prototypé une horloge numérique sur un afficheur LED avec mise à jour automatique de l'heure, mise en mémoire d'alarmes et de musiques jouées sur un buzzer.

Matériel nécessaire

Nous avons utilisé une carte Arduino Uno, un module de buzzer dit actif, un module d'horloge en temps réel DS1302 et un afficheur LED.



L'horloge ne sera pas tout le temps branchée, d'où l'importance d'avoir un module DS1302 qui a une pile intégrée.

Schéma de branchement du module DS1302 sur Arduino Uno

Bibliothèque Arduino du DS1302 : <https://www.velleman.eu/products/view/?id=435516>

rgba(255,255,255,1)

Réalisation : fichiers et script du projet

Nous avons créé des boîtiers sur Fusion 360. Les fichiers sources, STL et Arduino sont à disposition.

<https://cloud.ccprf.fr/index.php/s/G66xdsadK3XL4fA>

Code de Mise en place des Bibliothèques :

```
/*-----
```

Code effectué pour la fabrique de AMANLIS et JANZE

créé le 24/12/2019

Grégory PINVIN

Premier projet Arduino

```
-----*/
```

```
/*-----
```

Includes

```
-----*/
```

```
#include <SPI.h>      //SPI.h must be included as DMD is written by SPI (the IDE complains otherwise)
```

```
#include <DMD.h>      //
```

```
#include <TimerOne.h> //
```

```
#include « SystemFont5x7.h »
```

```
#include « Arial_black_16.h »
```

```
//Fire up the DMD library as dmd
```

```
#define DISPLAYS_ACROSS 1
```

```
#define DISPLAYS_DOWN 1
```

```
DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);
```

```
//horloge
```

```
#include <stdio.h>
```

```
#include <DS1302.h>
```

Nous allons mettre en mémoire les alarmes. Pour optimiser l'espace mémoire, nous numérotions les jours, 1 correspondant à lundi, puis puis l'heure et enfin les minutes.

```
//mise en place des fermetures par jour jour (1 = lundi, heure, min (indiquez l'heure de fermeture)
```

```
// 20 emplacements memoire
```

```
int jhFermeture[20][3] = {
```

```
    3,19,50,
```

```
    6,11,50,
```

2,22,50,

2,20,50,

2,21,50

};

//temps en minutes entre l'alarme et la fermeture

int refFermeture = 10; // nombre de minutes avant la fermeture

int minFermeture; //variable nous servant au décompte

int minFermeture2 = 0; //variable de comparaison du défilement de minutes

//temps d'affichage de fermeture

int refFerme = 5; //nombre de minutes indiquant la fermeture

int minFerme; //variable nous servant au décompte

// variable d'alarme de fermeture déclenchée

int alarmeFerme = 0; // variable nous indiquant si une alarme est en cours

Mise en place des variables permettant le retrait des infos du DS1302, sous forme de texte.

//variable pour la comparaison des heures

char buf[7];

char bufj[2];

char bufh[3];

char bufm[3];

char bufday[3];

Mise en place des information pour la mélodie. Ici le BUZZER est placé en Pin A3 de l'arduino.

// mise en place du buzzer

int pinSon = A3; // pin de connection du haut-parleur

int joueM1 = 0; //joue la musique si à 1

int joueM2 = 0;

Mise en place des variables pour jouer les mélodies et mise en mémoire des notes de la mélodie.

// les variables de tempo 120

int ronde = 2000;

int blanche = 1000;

int noire = 500;

int noirept = 750;

int croche = 250;

int crochept = 375;

int dcroche = 125;

int pause = 2000;

int demipause = 1000;

int soupir = 500;

```
int dsoupir = 250;
```

```
int qsoupir = 125;
```

```
/*-----
```

```
Gamme ouverture
```

```
-----*/
```

```
int gamme[9][4] = {
```

```
    pinSon,262,dcroche,10,
```

```
    pinSon,294,dcroche,10,
```

```
    pinSon,330,dcroche,10,
```

```
    pinSon,349,dcroche,10,
```

```
    pinSon,392,dcroche,10,
```

```
    pinSon,440,dcroche,10,
```

```
    pinSon,494,dcroche,10,
```

```
    pinSon,523,dcroche,10,
```

```
    -1
```

```
};
```

```
/*-----
```

```
Changement de minute
```

```
-----*/
```

```
int chgtMinute[6][4] = {
```

```
    pinSon,175,croche,10,
```

```
    pinSon,175,croche,10,
```

```
    pinSon,175,croche,10,
```

```
    pinSon,175,croche,10,
```

```
    pinSon,175,croche,10,
```

```
    -1
```

```
};
```

```
/*-----
```

```
Musiques Fermetures
```

```
-----*/
```

```
int StarWars[20][4] = {
```

```
    pinSon,196,noire,0,
```

```
    pinSon,196,noire,0,
```

```
    pinSon,196,noire,0,
```

```
    pinSon,156,crochept,0,
```

```
    pinSon,233,dcroche,0,
```

```
    pinSon,196,noire,0,
```

```
pinSon,156,crochept,0,  
pinSon,233,dcroche,0,  
pinSon,196,blanche,0,  
pinSon,294,noire,0,  
pinSon,294,noire,0,  
pinSon,294,noire,0,  
pinSon,311,crochept,0,  
pinSon,233,dcroche,0,  
pinSon,185,noire,0,  
pinSon,155,crochept,0,  
pinSon,233,dcroche,0,  
pinSon,196,blanche,0,  
-1
```

```
};
```

```
int PtBonhomme[16][4] = {
```

```
pinSon,131,croche,0,  
pinSon,165,croche,0,  
pinSon,147,noire,0,  
pinSon,131,noire,0,  
pinSon,123,noire,0,  
pinSon,131,noire,0,  
pinSon,98,blanche,10,  
pinSon,98,noire,0,  
pinSon,131,croche,0,  
pinSon,165,croche,0,  
pinSon,147,noire,0,  
pinSon,131,noire,0,  
pinSon,123,noire,0,  
pinSon,131,noire,0,  
pinSon,147,blanche,0,  
-1
```

```
};
```

```
int IleEnfant[28][4] = {
```

```
pinSon,147,croche,0,  
pinSon,165,croche,0,  
pinSon,175,croche,0,  
pinSon,196,noire,0,
```

```

pinSon,196,croche,0,
pinSon,247,croche,0,
pinSon,294,noire,10,
pinSon,294,croche,0,
pinSon,330,croche,0,
pinSon,294,croche,0,
pinSon,220,croche,10,
pinSon,220,croche,10,
pinSon,220,croche,10,
pinSon,220,noire,0,
pinSon,294,croche,0,
pinSon,262,croche,0,
pinSon,196,croche,0,
pinSon,262,croche,0,
pinSon,247,croche,10,
pinSon,247,croche,0,
pinSon,196,croche,0,
pinSon,220,croche,0,
pinSon,247,croche,0,
pinSon,220,croche,0,
pinSon,165,croche,0,
pinSon,196,croche,0,
pinSon,220,croche,0,
-1

```

```
};
```

Variables de branchement du DS1302 et des variables de codage de l'heure.

Ce code a été récupéré directement sur le site de VELLMAN.

```

namespace {

// Set the appropriate digital I/O pin connections. These are the pin
// assignments for the Arduino as well for as the DS1302 chip. See the DS1302
// datasheet:
//
// http://datasheets.maximintegrated.com/en/ds/DS1302.pdf
const int kCePin  = 2; // Chip Enable
const int kloPin  = 3; // Input/Output
const int kSclkPin = 4; // Serial Clock
// Create a DS1302 object.

```

```

DS1302 rtc(kCePin, kIoPin, kSclPin);

String dayAsString(const Time::Day day) {
    switch (day) {
        case Time::kSunday: return « 7 »;
        case Time::kMonday: return « 1 »;
        case Time::kTuesday: return « 2 »;
        case Time::kWednesday: return « 3 »;
        case Time::kThursday: return « 4 »;
        case Time::kFriday: return « 5 »;
        case Time::kSaturday: return « 6 »;
    }
    return « (unknown day) »;
}

void printTime() {
    // Get the current time and date from the chip.
    Time t = rtc.time();
    // Name the day of the week.
    const String day = dayAsString(t.day);
    // Format the time and date and insert into the temporary buffer.
    snprintf(buf, sizeof(buf), « %02d:%02d », t.hr, t.min);
    snprintf(bufj, sizeof(bufj), « %s », day.c_str());
    snprintf(bufh, sizeof(bufh), « %02d », t.hr);
    snprintf(bufm, sizeof(bufm), « %02d », t.min);
    snprintf(bufday, sizeof(bufday), « %02d », t.date);
    // Print the formatted string to serial so we can see the time.
    Serial.println(buf);
    dmd.selectFont(SystemFont5x7);
    dmd.clearScreen( true );
    dmd.drawString(2, 4, buf, 5, GRAPHICS_NORMAL );
}

} // namespace

/*-----
Interrupt handler for Timer1 (TimerOne) driven DMD refresh scanning, this gets
called at the period set in Timer1.initialize();
-----*/

void ScanDMD()
{

```

```
dmd.scanDisplayBySPI();
}
```

Nous voici arrivés dans la partie SETUP.

Mise en place de la communication à 9600 Bauds pour la console.

```
/*-----
setup
Called by the Arduino architecture before the main loop begins
-----*/

void setup()
{
Serial.begin(9600);
//initialisation pour beeper
pinMode(pinSon,OUTPUT);
//initialize TimerOne's interrupt/CPU usage used to scan and refresh the display
Timer1.initialize( 5000 ); //period in microseconds to call ScanDMD. Anything longer
than 5000 (5ms) and you can see flicker.
Timer1.attachInterrupt( ScanDMD ); //attach the Timer1 interrupt to ScanDMD which goes
to dmd.scanDisplayBySPI()
//clear/init the DMD pixels held in RAM
dmd.clearScreen( true ); //true is normal (all pixels off), false is negative (all pixels on)
```

Voici les lignes permettant de régler le DS1302 lors du changement de pile :

- mettre la protection d'écriture sur faux (false)
- forcer la variable « t » avec la date choisie lors du téléversement
- écrire la nouvelle date heure sur le DS1302

```
// Initialize a new chip by turning off write protection and clearing the
// clock halt flag. These methods needn't always be called. See the DS1302
// datasheet for details.
rtc.writeProtect(true); // mettre la valeur à false pour regler ds1302
rtc.halt(false);
// Make a new time object to set the date and time.
// Sunday, September 22, 2013 at 01:38:50.
//Time t(2019, 12, 31, 15, 21, 45, Time::kTuesday); // commenter cette ligne pour regler
ds1302
// Set the time and date on the chip.
//rtc.time(t); // commenter cette ligne pour regler ds1302
```

Affichage du message « Hello » suivi d'une gamme musicale permettant la vérification de fonctionnement du buzzer.

```
//Fonction affichage mise sous tension
dmd.clearScreen( true );
dmd.selectFont(SystemFont5x7);
```

```

dmd.drawString( 1,4, « Hello », 5, GRAPHICS_NORMAL );
//jouer la musique de départ
for (int p=0; gamme[p][0] != -1; p++) {
    if (gamme[p][0] != -1){
        tone (gamme[p][0], gamme[p][1]);
        delay (gamme[p][2]);
    }
    if (gamme[p][3] != -1){
        noTone(gamme[p][0]);
        delay (gamme[p][3]);
    }
}
dmd.clearScreen( true );
printTime(); //print time très rapide pour affichage 00:00 premiere lecture
delay (100);
}

```

Nous voici arrivés dans la partie LOOP

A chaque boucle, nous vérifions si l'heure correspond à une alarme.

La comparaison ne s'effectue que si aucune alarme n'est en route.

```

/*-----
loop
Arduino architecture main loop
-----*/

void loop()
{
    byte b;
    //affichage de l'heure
    printTime();
    //comparaison heure en cours avec tableau des alarmes
    if (alarmeFerme == 0){
        for (int p = 0; p < 20; p++){
            if (jhFermeture[p][0] != -1){
                if (jhFermeture[p][0] == atoi(bufj)){
                    if (jhFermeture[p][1] == atoi(bufh)){
                        if (jhFermeture[p][2] == atoi(bufm)){
                            alarmeFerme = 1;
                            minFerme = refFerme;
                        }
                    }
                }
            }
        }
    }
}

```

```

        minFermeture = refFermeture;
        joueM1 = 1;
        joueM2 = 1;
        minFermeture2 = atoi(bufm);
        fermeture();
    }
}
}
}
}
delay (2000);
}
else {
    fermeture();
}
}

```

Nous avons créer une boucle de début d'alarme avant fermeture.

Cela nous permet d'éviter des boucles ou conditions inutiles au traitement.

Nous commençons par vérifier si c'est le premier déclenchement d'alarme. Si c'est le cas, une musique retenti, sinon, 5 bips sont joués à chaque nouvelle minute pour attirer l'attention.

Enfin, nous affichons le message d'information de fermeture.

```

/*-----
Fonction affichage avant fermeture
-----*/

void fermeture(){
    if (minFermeture != 0){
        dmd.clearScreen( true );
        dmd.selectFont(SystemFont5x7);
        dmd.drawString(1, 0, « FERME » , 5, GRAPHICS_NORMAL );
        dmd.drawString(4, 8, « DANS » , 4, GRAPHICS_NORMAL );
        if ( minFermeture2 != atoi(bufm)){
            minFermeture = minFermeture - 1;
            minFermeture2 = atoi(bufm);
            if (minFermeture == 0){
                goto bailout;
            }
        }
        //tonalités de changement de minutes
        for (int nbr=0; chgtMinute[nbr][0] != -1; nbr++) {

```

```

    if (chgtMinute[nbr][0] != -1){
        tone (chgtMinute[nbr][0], chgtMinute[nbr][1]);
        delay (chgtMinute[nbr][2]);
    }
    if (chgtMinute[nbr][3] != -1){
        noTone(chgtMinute[nbr][0]);
        delay (chgtMinute[nbr][3]);
    }
}
}
}
if (joueM1 == 1){
    musique();
    joueM1 = 0;
}
delay (3000);
dmd.clearScreen( true );
int p = minFermeture;
char cstr[16];
int lg;
int lg2;
itoa(p, cstr, 10);
if (p > 9){
    lg = 1;
    lg2 = 14;
}
else {
    lg = 4;
    lg2 = 11;
}
dmd.drawString(lg, 4, cstr , 2, GRAPHICS_NORMAL );
dmd.drawString(lg2, 4, « Min » , 3, GRAPHICS_NORMAL );
delay (3000);
dmd.clearScreen( true );
dmd.selectFont(Arial_Black_16);
dmd.drawMarquee(« Veuillez ranger les locaux »,27,(32*DISPLAYS_ACROSS)-1,0);
long start=millis();
long timer=start;

```

```

boolean ret=false;
while(!ret){
    if ((timer+30) < millis()) {
        ret=dmd.stepMarquee(-1,0);
        timer=millis();
    }
}
else {
    bailout:
    minFermeture2 = atoi(bufm);
    ferme();
}
}

```

Lors de la fermeture, nous procédons comme au début d'alarme mais avec de nouveaux textes.

```

/*-----
Fonction affichage fermé
-----*/

```

```

void ferme(){
    if (minFerme != 0){
        dmd.clearScreen( true );
        dmd.selectFont(SystemFont5x7);
        dmd.drawString(1, 3, « FERME », 5, GRAPHICS_NORMAL );
        dmd.drawLine(1,12,29,12,GRAPHICS_NORMAL);
        if ( minFermeture2 != atoi(bufm)){
            minFerme = minFerme - 1;
            minFermeture2 = atoi(bufm);
            if (minFerme == 0){
                goto bailout;
            }
        }
        //tonalités de changement de minutes
        for (int nbr=0; chgtMinute[nbr][0] != -1; nbr++) {
            if (chgtMinute[nbr][0] != -1){
                tone (chgtMinute[nbr][0], chgtMinute[nbr][1]);
                delay (chgtMinute[nbr][2]);
            }
            if (chgtMinute[nbr][3] != -1){

```

```

        noTone(chgtMinute[nbr][0]);
        delay (chgtMinute[nbr][3]);
    }
}
}
if (joueM2 == 1){
    musique();
    joueM2 = 0;
}
delay (5000);
dmd.clearScreen( true );
dmd.selectFont(Arial_Black_16);
dmd.drawMarquee(« Veuillez liberer les locaux »,27,(32*DISPLAYS_ACROSS)-1,0);
long start=millis();
long timer=start;
boolean ret=false;
while(!ret){
    if ((timer+30) < millis()) {
        ret=dmd.stepMarquee(-1,0);
        timer=millis();
    }
}
}
else {
    bailout:
    alarmeFerme = 0;
    printTime();
}
}

```

Une condition a été mise en place dans la boucle musique pour permettre de jouer trois airs différents suivants la date du jour. Ainsi vous pouvez intégrer ici une mélodie de Noël....

```
/*-----
```

Fonction musique alarme

il y a une musique pour chaque jour

```
-----*/
```

```

void musique(){
    if (atoi(bufday) <= 10){
        for (int p=0; StarWars[p][0] != -1; p++) {

```

```

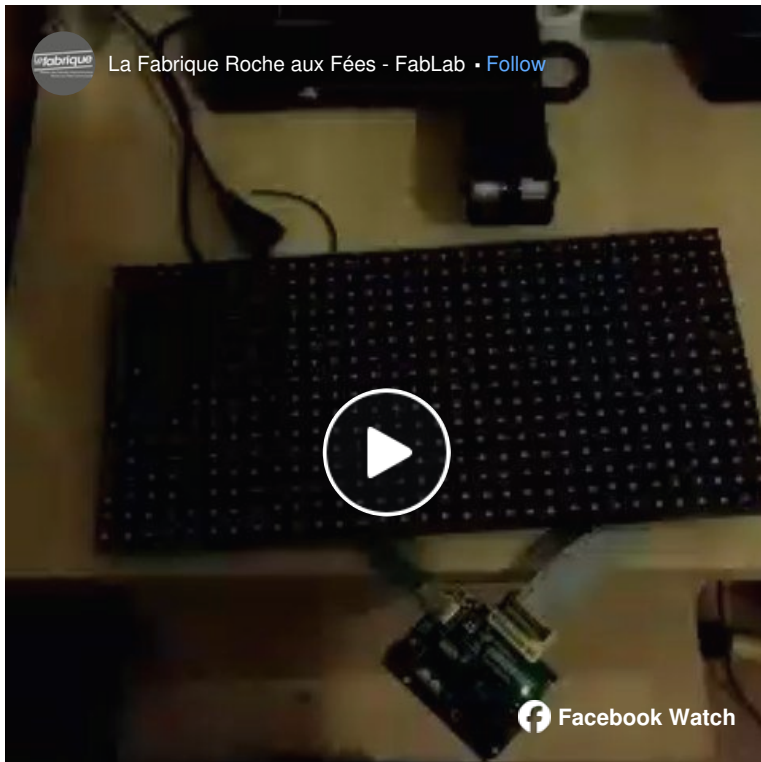
if (StarWars[p][1] != -1){
    tone (StarWars[p][0], StarWars[p][1]);
    delay (StarWars[p][2]);
}
if (StarWars[p][3] != -1){
    noTone(StarWars[p][0]);
    delay (StarWars[p][3]);
}
}
}
else {
    if (atoi(bufday) <= 20){
        for (int p=0; IleEnfant[p][0] != -1; p++) {
            if (IleEnfant[p][1] != -1){
                tone (IleEnfant[p][0], IleEnfant[p][1]);
                delay (IleEnfant[p][2]);
            }
            if (IleEnfant[p][3] != -1){
                noTone(IleEnfant[p][0]);
                delay (IleEnfant[p][3]);
            }
        }
    }
    else{
        for (int p=0; PtBonhomme[p][0] != -1; p++) {
            if (PtBonhomme[p][1] != -1){
                tone (PtBonhomme[p][0], PtBonhomme[p][1]);
                delay (PtBonhomme[p][2]);
            }
            if (PtBonhomme[p][3] != -1){
                noTone(PtBonhomme[p][0]);
                delay (PtBonhomme[p][3]);
            }
        }
    }
}
}

```

rgba(255,255,255,1)

Résultats

Premier test de l'affichage du texte :



Résultat final avec ajout de la musique et mise à jour automatique de l'heure

Publié le 04 janvier 2020