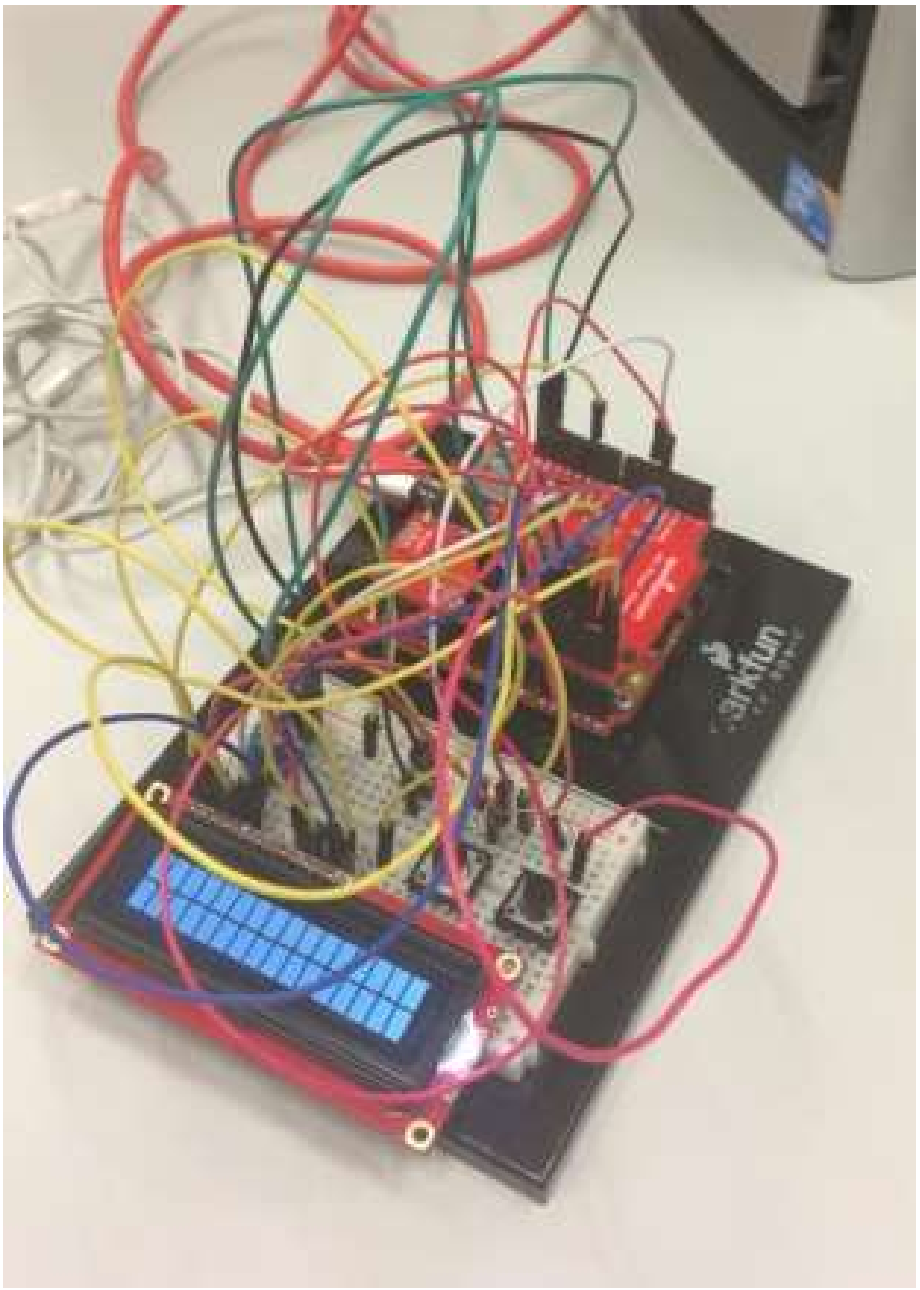
Réalisation d'un lecteur MP3



I. Introduction

Le but de ce TP est de réaliser un lecteur MP3 à l'aide d'une RedBoard Arduino (composée entre autres d'un microprocesseur ATmega328) et d'un Shield MP3 Player.

Ce lecteur MP3 possède une interface homme-machine improvisée qui permet :

- Le choix de la chanson à écouter
- Le réglage du volume
- L'utilisation d'un bouton play/pause
- L'utilisation d'un bouton pour passer à la piste suivante
- L'affichage sur un écran LCD du titre de la chanson et du temps écoulé depuis le début de la lecture de cette dernière

II. Architecture du programme

Le programme est composé :

- Des instructions du préprocesseur contenant les inclusions des bibliothèques, des déclarations des ports entrées/sorties ainsi que des variables et des configurations matérielles.
- De la fonction ***setup()*** qui comme son nom l'indique initialise les bus d'entrées sortie du périphérique.
- Du code des différents fonctions qui servent à réaliser les actions citées ci-dessus.
- De la fonction ***loop()*** exécutée en boucle lors du fonctionnement du programme, elle appelle les fonctions.

III. Choisir la chanson à écouter

Pour choisir une piste, nous nous sommes servis de l'USART, dans la fonction ***choisirpiste()***. Pour ce faire nous testons d'abord si la liaison série est disponible. Ensuite nous lisons le caractère lu par le port série et nous retranchons 48 qui correspond à 0 en ASCII. Le choix de l'utilisateur est entré dans le terminal et testé s'il est compris entre 1 et le nombre de musique sur la carte SD. La musique qui est écoutée est ensuite interrompue et remplacée par la musique sélectionnée à l'aide de la commande ***playTrack()***.

IV. Réglage du volume via un potentiomètre

Le réglage du volume est fait à l'aide d'un potentiomètre et de la fonction ***reglerVolume()***. La valeur lue sur le potentiomètre branchée sur l'entrée A0 est alors redimensionnée à l'aide la fonction ***map()*** pour faire passer une valeur allant de 0 à 1023 à une valeur allant de 0 à 100. Cette valeur de 100 est arbitraire et peut être mise à 255 puisque la fonction ***setVolume()*** prend en entrée un entier non signé sur 8 bits. Le choix de la valeur 100 est juste dans le but de limiter le volume maximal afin de ne pas brider la qualité sonore en saturant le son et de préserver l'ouïe de l'utilisateur.

V. Boutons poussoirs

1. Variables

Les variables récurrentes sont :

- bout1/bout2 : Noms données respectivement au bouton 1 et 2
- etat1/etat2 : Représentent respectivement l'état des boutons 1 et 2 (LOW ou HIGH)
- time1/time2 : Temps servant au système anti-rebond pour les boutons 1 et 2

2. Utilisations

Dans ce code, nous avons besoin de deux boutons, un pour gérer la lecture de la piste actuelle en effectuant la mise en pause et sa lecture (appelé bout1) et un autre pour permettre le passage à la piste suivante (appelé bout2).

Nous avons créé la fonction ***Appuyer()*** qui permet de changer l'état associé à chaque bouton en évitant les rebonds. Elle prend en entrée une variable représentant un bouton, l'état actuel du bouton sélectionné et le temps associé à ce bouton.

Cette fonction change l'état du bouton concerné s'il a été activé ou relâché en testant sa valeur toutes les 50 millisecondes.

3. Bouton play/pause

La fonction ***PlayPause()*** permet de mettre sur pause la musique actuelle et de continuer sa lecture si elle est sur pause. Pour ce faire la valeur du bouton numéro 1 (noté bout1) est changée à l'aide de la fonction ***Appuyer()***. Si le bouton 1 est actionné, alors la musique est arrêtée si elle est en lecture et inversement.

4. Bouton piste suivante

La fonction ***pisteSuivante()*** permet de passer à la chanson suivante. Pour ce faire la valeur du bouton numéro 2 (noté bout2). Pour ce faire la valeur du bouton numéro 1 (noté bout1) est changée à l'aide de la fonction ***Appuyer()***. Ensuite la valeur de indiceMusique est incrémentée jusqu'à 4 (le nombre de musique sur la carte sd) et à 5 la valeur de indiceMusique repasse à 1. La musique correspondant à indiceMusique est ensuite jouée.

5. Affichage sur l'écran LCD

L'affichage sur l'écran est rafraichi toutes les secondes (pour mettre à jour le temps écoulé) à l'aide des fonctions ***clear()*** et ***print()***. Les minutes et les secondes sont obtenues en divisant par 60 le temps écoulé depuis le début de la lecture et en prenant respectivement le quotient et le reste.

A . Annexe

```
#include <SPI.h>
#include <SdFat.h>
#include <SdFatUtil.h>
#include <SFEMP3Shield.h>

#define pot A0
#define bout1 4
#define bout2 5

SdFat sd;
SFEMP3Shield MP3player;

unsigned char resultat;
int indiceMusique;

int etat1 = LOW, etat2 = LOW;
unsigned int time1, time2;

#define rs 10
#define enable A1
#define d4 A5
#define d5 A4
#define d6 A3
#define d7 A2

LiquidCrystal lcd(rs, enable, d4, d5, d6, d7);
unsigned int time_lcd;

void setup() {
  pinMode(pot,INPUT);
  pinMode(bout1, INPUT);
  pinMode(bout2, INPUT);

  //Initialisation de la liaison série.
  Serial.begin(115200);

  //Initialisation de la carte SD.
  if(!sd.begin(SD_SEL, SPI_FULL_SPEED)) sd.initErrorHalt();
  if(!sd.chdir("/")) sd.errorHalt("sd.chdir");

  //Initialize the MP3 Player Shield
  resultat = MP3player.begin();

  time1 = millis();
  time2 = millis();

  lcd.begin(16,2); //ecran : 16 colonnes, 2 lignes
```

```
  time_lcd = millis();

}


void loop() {
 choixPiste();
 reglerVolume();
 PlayPause();
 pisteSuivante();

}

// Choix de la chanson à écouter
void choixPiste(){

  if (Serial.available() > 0) {
    indiceMusique = Serial.read()-48;
    if (indiceMusique>4){
      Serial.print ( "choisir un  numero entre 1 et 4");}

    if(MP3player.isPlaying()==1) {
      MP3player.stopTrack();}
    resultat = MP3player.playTrack(indiceMusique);
  }
}

void reglerVolume(){
  unsigned int volume;
  volume = analogRead(pot);
  uint8_t vol = map (volume, 0, 1023, 0, 100);
  MP3player.setVolume(vol);
}

int Appuyer(int button, int& state, unsigned int& time){
  int flag = 0;
  int newstate = digitalRead(button);

  if((newstate!=state)&&(millis()-time>=50)){
    if(state == LOW && newstate == HIGH ) {
      flag=1;
      state = HIGH;
    }
    else if (newstate == LOW) {state = LOW;}
    time = millis();
  }
  return flag;
}

void PlayPause(){
  int val = Appuyer(bout1,etat1, time1);
  if(val){
    if(MP3player.getState()==playback) {
      MP3player.pauseMusic();
    }
```

```
    else if(MP3player.getState()==paused_playback){
      MP3player.resumeMusic();
    }
 }
}

void pisteSuivante(){
  int val = Appuyer(bout2,etat2, time2);
  if(val){
    indiceMusique++;
    if (indiceMusique==5) indiceMusique=1;
    if(MP3player.isPlaying()) MP3player.stopTrack();
    MP3player.playTrack(indiceMusique);
  }
}


void affiche(){
  if((MP3player.getState()==playback)&&(millis()-time_lcd>=1000)){
    lcd.clear();
    lcd.print("Piste ");
    lcd.print(indTrack%nbTrack+1);
    lcd.setCursor(0, 1);
    unsigned int posit = MP3player.currentPosition();
    lcd.print(posit/60); lcd.print("' "); lcd.print(posit%60);
lcd.print("'' ");
  }
```