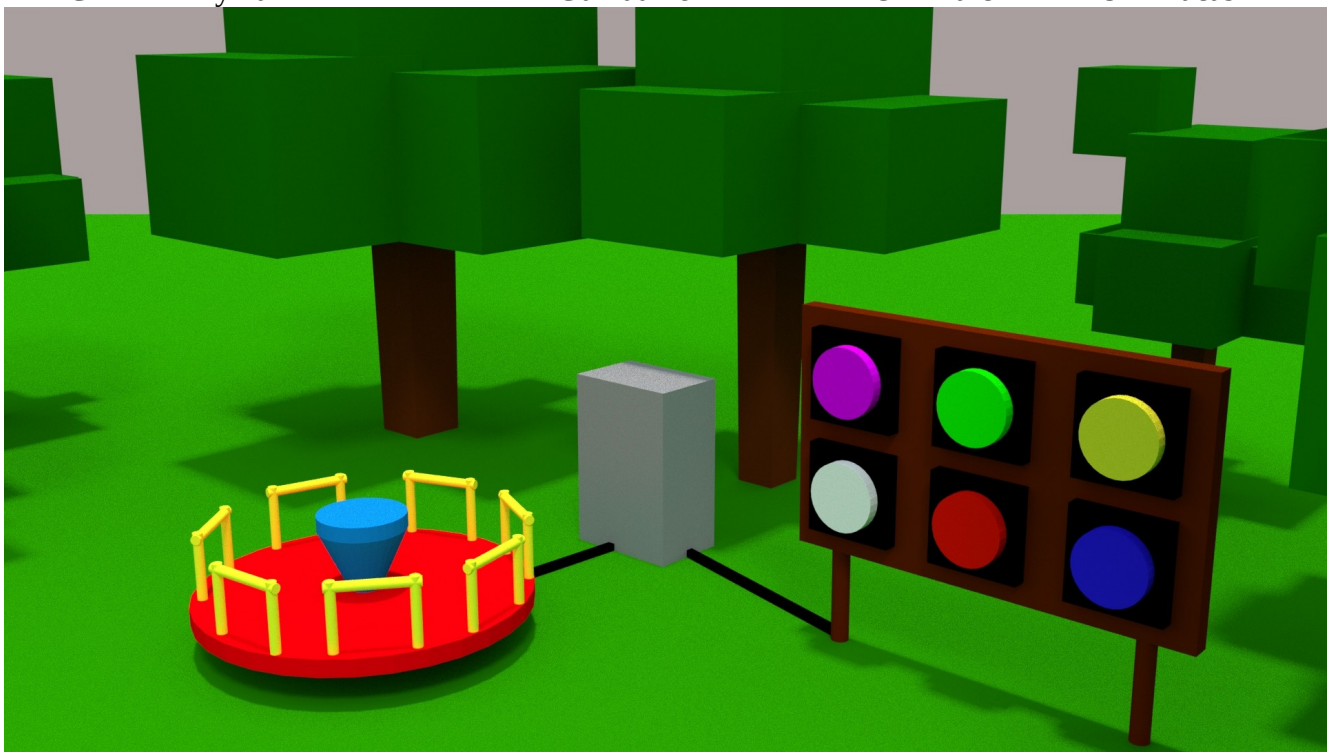
Compte rendu Travaux Pratique Encadré

Problématique: Comment rendre plus ludique un parc communal pour enfant ?

KACHMAR Ayman DINDART Guillaume RATTON Marc RUIZ Lucas



2018 – 2019

Table des matières

I – Introduction

Le divertissement et l'éducation sont des mots très liés de nos jours, c'est pour cela que nous avons réfléchis sur une problématique pour un TPE basé sur ces deux mots. Notre but est de promouvoir l'éducation des enfants tout en y combinant le loisir. Depuis les années 90 les parcs communaux se sont démocratisés. Ils n'étaient que des lieux de divertissement mais font aujourd'hui l'objet de lieu d'éducation. Notre projet consiste à rendre plus ludique un parc communal pour enfant pendant leur tournée de jeu tout en récupérant de l'énergie créée grâce aux différentes attractions afin de la retransmettre dans une toute nouvelle attraction ludique.

A l'aide de ce projet, nous serons sans doute capables de répondre à la problématique suivante :

Comment rendre plus ludique un parc communal pour enfant ?

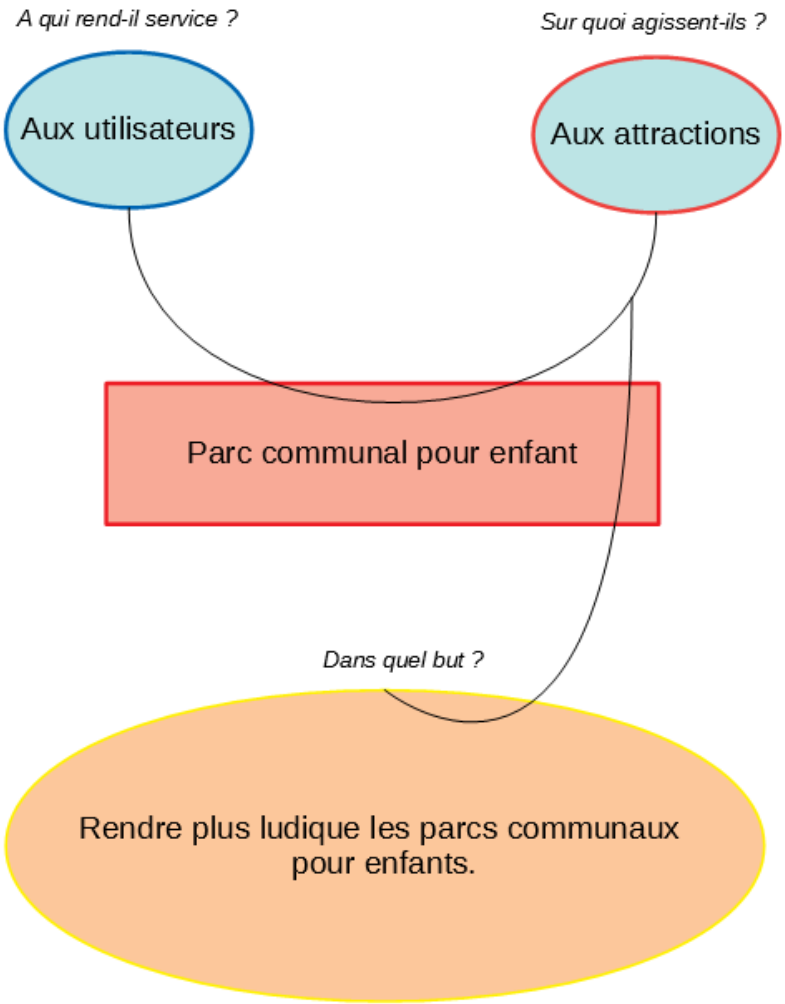
Nous avons choisi ce projet car il s'agit d'un projet éducatif et ludique qui permet aux enfants d'apprendre des notions de base tout en prenant plaisir à jouer. L'étude se base principalement sur deux attractions et est répartie en deux parties : l'une mécanique, l'autre électronique. Les attractions sont des objets capables de produire de l'énergie tel que la balançoire à double assise (ou simple), le tourniquet et bien d'autres. L'attraction choisie en premier est le tourniquet car il a des caractéristiques
spéciales que nous vous expliquerons tout au long du projet. Ensuite, la deuxième attraction choisie est le tableau d'apprentissage de couleurs qui requiert une partie électronique et programmation.

II – Cahier des charges

Avant de débuter le projet, un cahier des charges doit être élaboré afin de lister avec précision les attentes et les exigences liées au projet. Il permet aussi de délimiter les conditions de réalisations de ce projet. C'est donc un document essentiel à la réalisation et à l'élaboration de ce dernier. Dans notre cas il contient différents types de diagrammes : la bête à corne, la pieuvre, un tableau des fonctions, le FAST et la hiérarchisation des fonctions.

A) Diagramme bête à corne

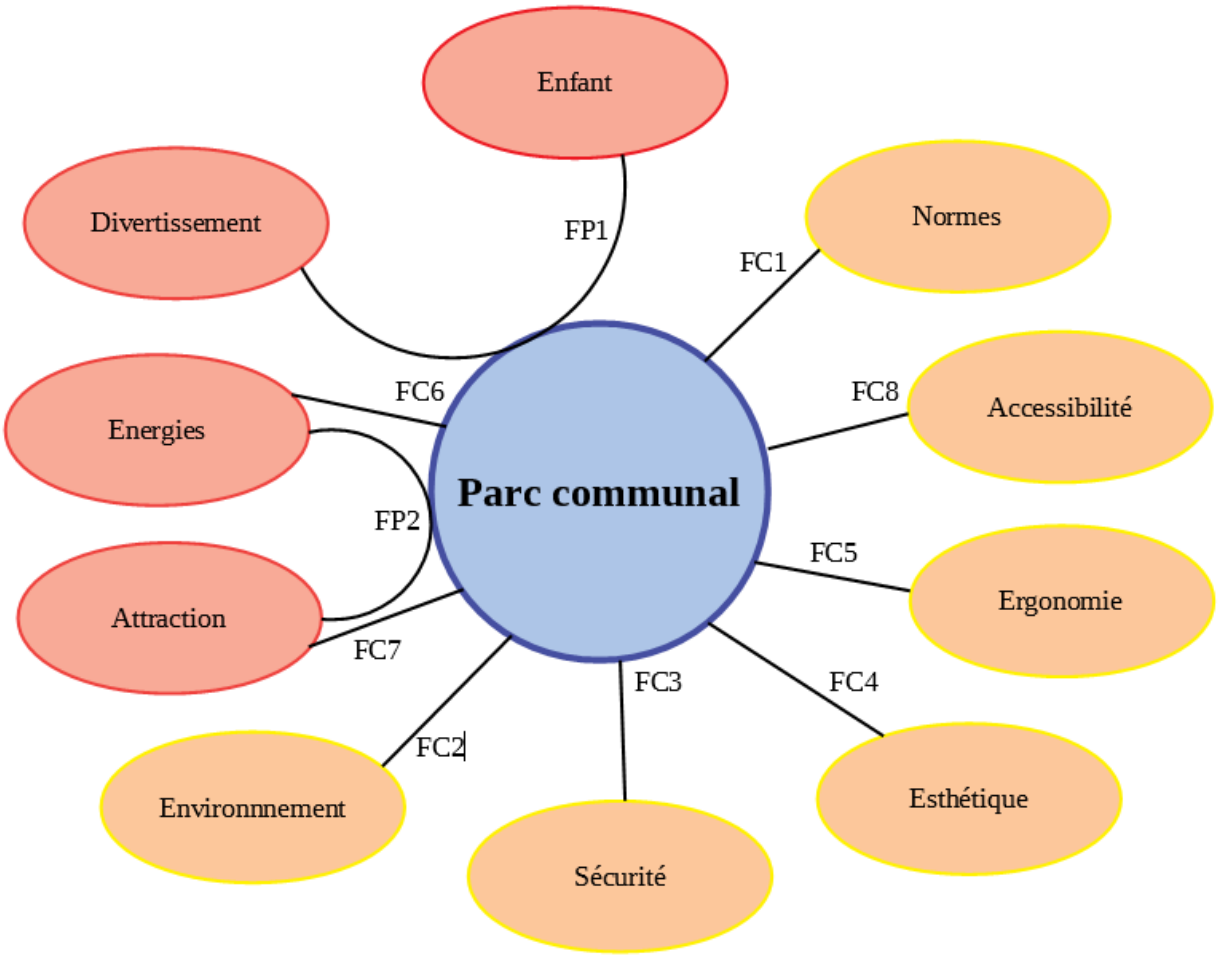
La bête à corne, de manière générale, est un outil d'analyse du besoin. Il permet de formuler le besoin sous forme de fonction principale (de base). Dans notre cas, on constate que les utilisateurs (principalement les enfants) agissent sur les attractions pour rendre plus ludique un parc communal pour enfant.



B) Diagramme Pieuvre

Le diagramme pieuvre est un outil d'analyse du besoin, il sert à identifier et à mettre en évidence les relations entre les différentes fonctions de service d'un produit. Ces fonctions dont les éléments obligatoire à prendre en compte pour la réussite du projet. Elles se divisent en deux parties : les fonctions principales (FP) et les fonctions contraintes (FC).

Voici le diagramme pieuvre de notre projet :



N°	Fonction	Mot dans la pieuvre
FP1	Divertir les enfant de manière plus efficace.	Divertissement et Enfant
FP2	Réutiliser l'énergie produite par les attractions.	Attraction et énergie
FC1	Respecter les normes en vigueurs	Normes
FC2	Résister aux condition climatique et extérieur	Environnement
FC3	Garantir et assurer la sécurité des enfants.	Sécurité
FC4	Attirer les enfant et rendre le parc plus agréable visuellement	Esthétique
FC5	Rendre facile l'utilisation du parc et des attractions	Ergonomie
FC6	Récupérer l'énergie produite par les enfants	Énergie
FC7	Mettre en place de différentes attractions	Attraction
FC8	Adapter le parc pour les personnes handicapées	Accessibilité

C) Diagramme FAST

je sais pas ou est le dossier ????????
JSP c’est toi qu l’avait

Notre projet est scindé en deux grandes parties comme expliqué précédemment, nous commencerons par la partie mécanique qui porte sur le tourniquet.

III – Mécanique

A) Choix de l'Attraction

Les parcs communaux sont constitués de plusieurs attractions toutes différentes. la plupart d'entre elles peuvent faire l'objet de notre projet cependant certaines sont plus aptes que d'autres à récupérer plus ou moins d'énergie. Pour cela nous nous sommes penchés sur la question : avec quelle attraction pouvons nous produit le plus d'énergie ?
Notre choix fut celui du tourniquet car le tourniquet est utilisé dans 95% des parcs communaux, de plus il est facile de récupérer de l'énergie avec son mouvement de rotation.

B) les Normes

En effet les parcs communaux sont soumis à des textes réglementaires, créés par le Comité Européen de Normalisation (CEN) et l'Afnor (association française de normalisation). Ces textes sont divisés en trois catégories : les décrets et arrêtés, les normes ainsi que les codes de la consommations.(voir annexe page /).

Cependant ce ne sont pas les seuls puisque les attractions sont elles même soumises à d'autres normes afin de garantir la sécurité et le bon fonctionnement de l'objet. Voici quelques unes de ces normes pour le tourniquet :

- La norme européenne EN-1176-5 indique que le plateau central du tourniquet doit être plein et circulaire. Son diamètre doit être compris entre 0,50 m et 3 m.
- La hauteur de chute libre ne peut être supérieure à 1m pour tout enfant placé sur le tourniquet.
- La présence d'une jupe est préconisée. Celle-ci doit être rigide et ne pas toucher le sol (à une distance comprise entre 8 cm et 11 cm du sol). En l'absence de jupe, le plateau doit être à une distance du sol comprise entre 8 cm et 11 cm, ou à une distance supérieure à 40 cm et ce, afin d'éviter les risques de coincement.
- La zone de sécurité doit être d'au moins 2 m à partir de n'importe quel point extérieur du tourniquet, y compris en hauteur.
- Un revêtement amortissant, non dégradable, doit être installé sur une largeur d'au moins 1,5 m au-delà de l'aplomb du plateau tournant sur tout le périmètre du tourniquet. Le niveau du sol sous le plateau doit être au même niveau que celui du revêtement de sécurité.

C) Les Composants

Le tourniquet est un ensemble d'objet liés entre eux. Ces objets doivent respecter les normes citées précédemment tout en étant le plus adapté à l'utilisation de celui-ci.
nous avons donc regroupé les composants pour notre tourniquet en trois catégories :

- Les composants extérieurs
  un plateau de 1,50m de diamètre en bois pour que l'attraction soit plus écologique
  des barres ou des bancs selon les critères de l'acheteur
  Un axe central ou les enfants pourront s'aider a tourner
  Un sol en tartan de 2m autour du tourniquet

- Les composants intérieurs
  deux roulements à billes qui permettent à faire tourner l'arbre par rapport à l'axe de rotation du tourniquet
  multiplicateur (suite d'engrenages) sert à augmenter la vitesse initial de notre tourniquet en sortie
  dynamo sert à transformer l'énergie mécanique en énergie électrique
  arbre sert à transmettre une puissance entre deux éléments
  entretoise est une pièce reliant deux autres pièces tout en gardant le même écart en elles

- les composants liés
  câbles
  batteries

mettre l'image du parc seulement avec les cables pour ne pas dévoiler la nouvelle attraction mais attendre que guillaume me l'envoie sur le drive

D) Le Mécanisme

Afin de récupérer l'énergie des attractions, il y a tout un cheminement à respecter en voici le mécanisme. tout d'abord nous allons définir les énergies mises en jeu.

Au départ nous avons une énergie cinétique de rotation créée par les enfants puis à la sortie du mécanisme nous aurons une énergie électrique créée par la dynamo.
Cette enchevêtrement se fait sous différentes étapes :

1ère étape : il y a une rotation du tourniquet ainsi que de l'axe central

2ème étape : l'axe central entraîne avec lui l'engrenage composé de plusieurs roues dentées qui augmentent la vitesse

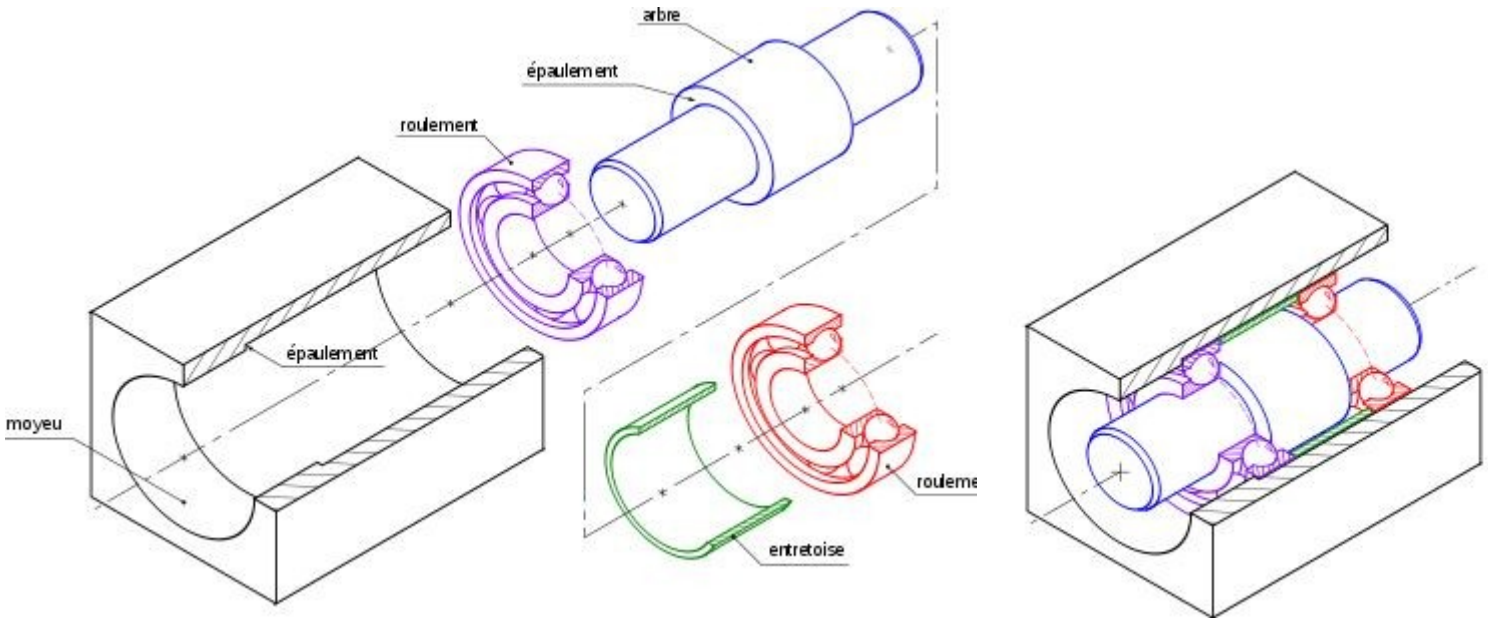
3ème étape : la dernière roue dentée de l'engrenage emporte dans son mouvement de rotation le galet de la dynamo à la vitesse de cette dernière

4ème étape : la dynamo transforme l'énergie de cette rotation en énergie électrique et cette énergie est transféré dans des batteries grâce aux câbles

Maintenant nous allons vous expliquer comment l'axe central fait-il pour rester droit et bien transmettre le mouvement de rotation ?

C'est très simple l'axe central fixe (aussi appelé moyeu) à un épaulement (petite marche sur la surface) soutenant un roulement à billes. Sur ce roulement vient s'ajouter un arbre composé d'un autre épaulement entouré d'une entretoise puis le deuxième roulement vient compléter le mécanisme.
Si ce mécanisme à roulement à billes n'existait pas alors l'axe central mobile du tourniquet pourrait se déplacer dans l'axe fixe.

Schémas représentants le fonctionnement d'un roulement à billes



Dans notre projet nous ne prendrons pas de roulements à billes à contact radial (roulement à billes les plus courants comme dans le schéma au dessus) mais nous choisirons les roulements à rouleaux coniques car ils supportent mieux les efforts axiaux et radiaux (dans notre cas ce sera le poids des enfants aux extrémités du tourniquet).

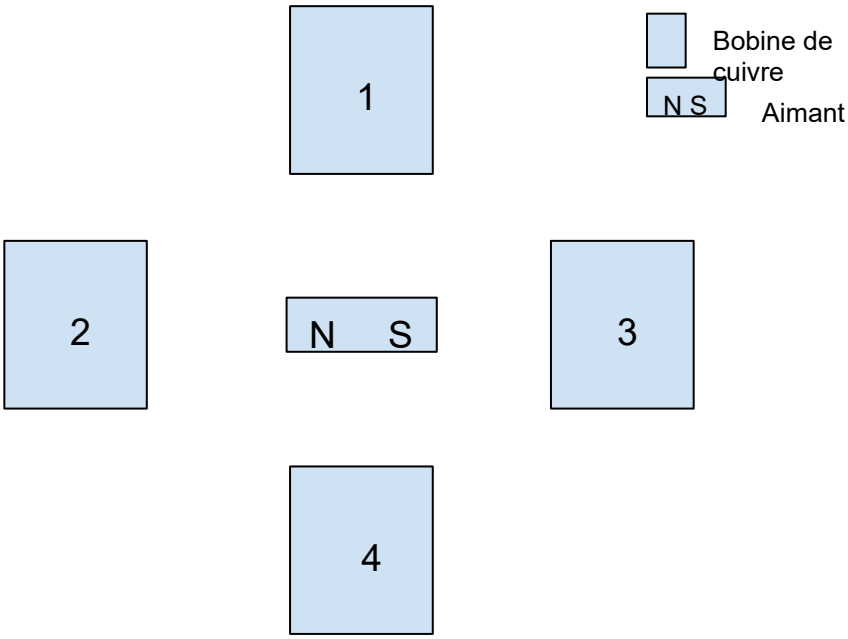
Roulement à billes

Roulements à rouleaux coniques

E) La Dynamo

La dynamo est l'élément le plus important puisqu'elle transforme l'énergie mécanique en énergie électrique. Afin de mieux comprendre comment la dynamo transforme cette énergie nous nous concentrerons sur son fonctionnement pas très compliqué.

Voici le schéma du fonctionnement d'une dynamo :



Les bobines 1 et 2 ont une face sud magnétique et les bobines 3 et 4 ont une face nord magnétique. Lorsque les bobines 2 et 3 vont s'allumer, l'aimant va se positionner de façon à ce que le côté nord de l'aimant se retrouve en face du côté sud de la bobine 2. Ensuite ont éteint les bobines 2 et 3 et on allume les 1 et 4. Ainsi l'aimant va tourner de façon à ce que le côté nord de l'aimant se retrouve en face du côté sud de la bobine 1. Il ne reste plus qu'à répéter les mêmes actions pour que l'aimant tourne sur lui même et ainsi créer un courant électrique qui sera récupéré par des câbles.

IV – Électronique et Programmation

A) Choix des Composants

Pour réaliser le tableau apprentissage des couleurs, nous avons commencé à nous poser la question de quoi on avait besoin pour réaliser le tableau. Après réflexion, nous avons réalisé une liste de matériel pour prototyper le tableau d'apprentissage des couleurs (par la suite, abrégé en tableau). Nous avons volontairement omis l'extérieur du boîtier pour nous concentrer uniquement sur la partie électronique. Nous avons alors obtenu la liste de matériels suivante :

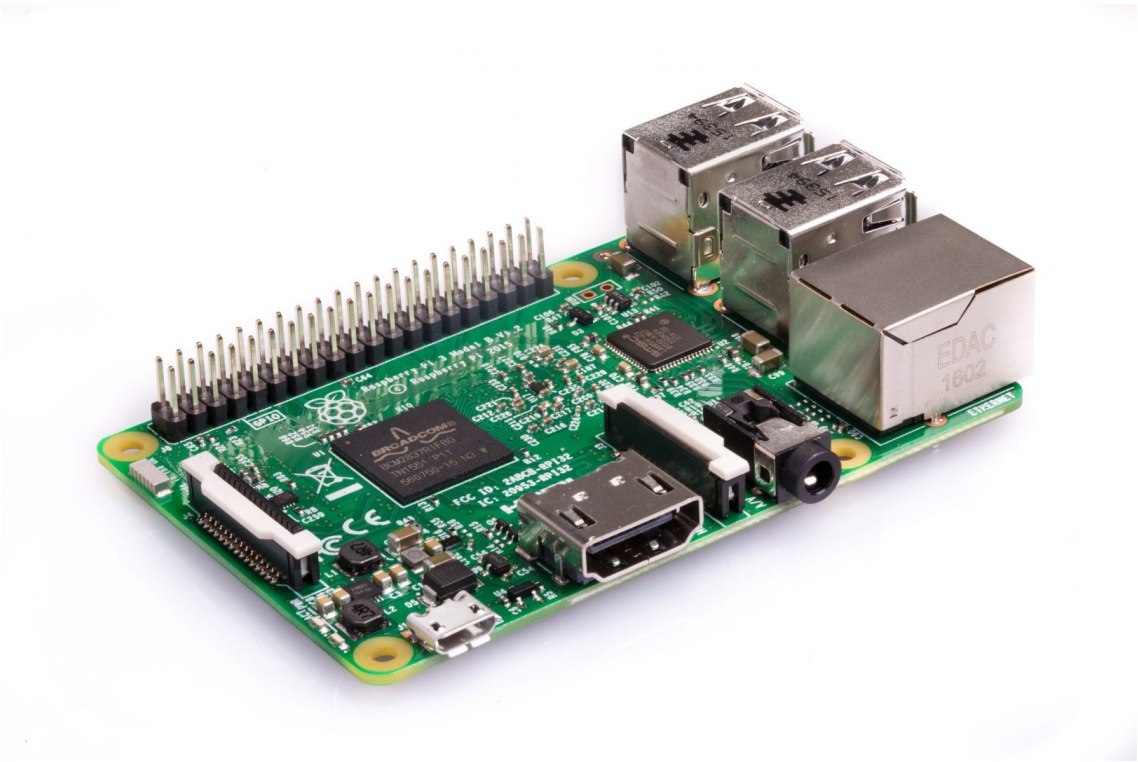
- un contrôleur
- 3 LEDs (les trois couleurs primaires : rouge, vert et bleu)
- 3 LEDs RGB (pour les trois couleurs secondaires : jaune, magenta et bleu cyan)
- un haut-parleur pour le son
- un amplificateur audio
- des boutons poussoir

Nous allons maintenant détailler quel composant on a pris et pourquoi.

1) Le Contrôleur

Pour réaliser le tableau, nous avions besoin d'un contrôleur pour coordonner le son, les lumières et l'appui sur les boutons. Nous avons donc recherché des contrôleurs, trois ont retenu notre attention.

Le premier est un Raspberry Pi, il a l'avantage d'avoir déjà un module sonore et de pouvoir traiter une grande quantité d'information et d'en stocker facilement.



Nous avions aussi comme autre hésitation deux cartes Arduino. Une carte de type Mega et une autre de type Uno. L'avantage de ces deux cartes c'est qu'elles sont pourvues d'un grand nombre d'entrées et de sorties et qu'elles consomment que très peu.

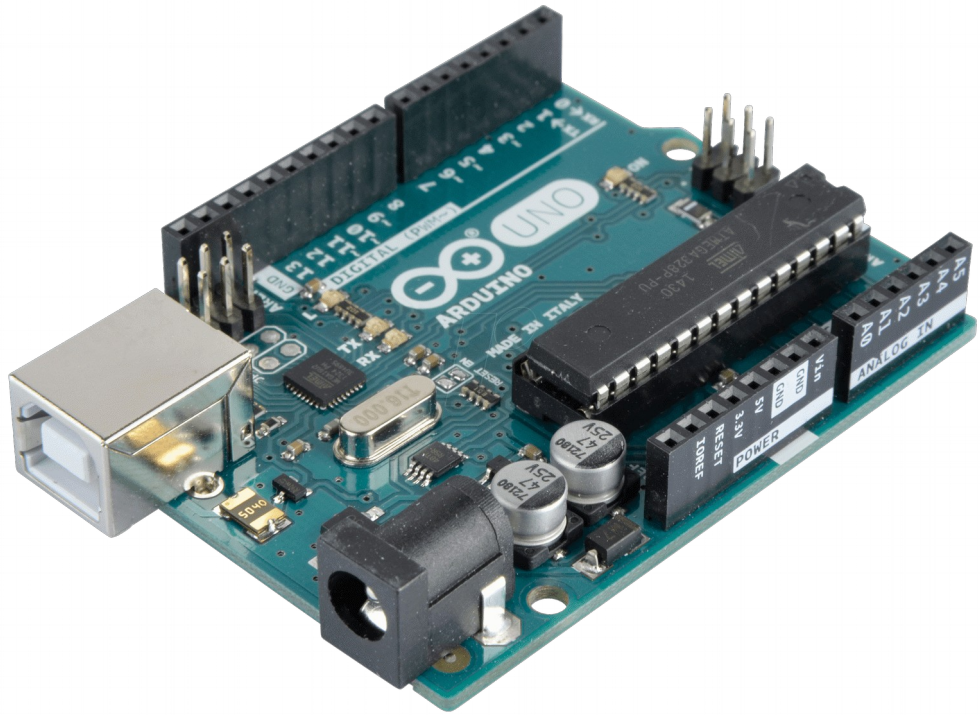


Illustration 1: Arduino UNO

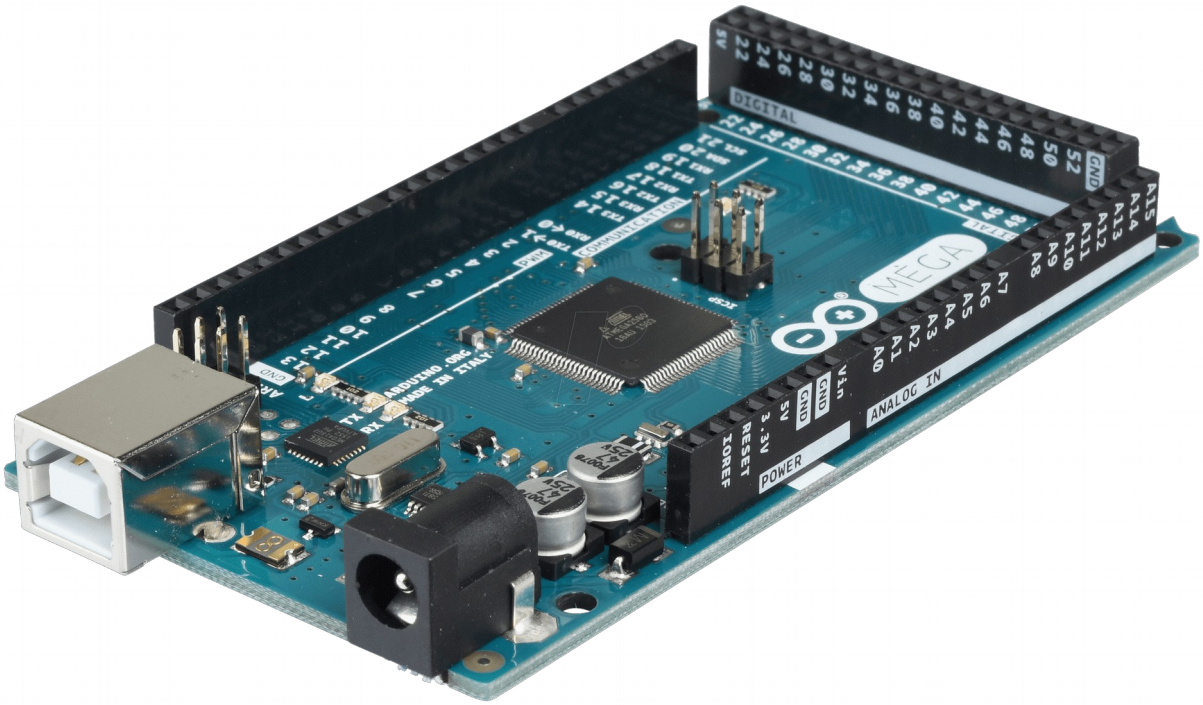


Illustration 2: Arduino Mega

Nom	Entrées numérique	Entrées analogique	Sortie numérique	Sortie analogique
Raspberry Pi	26 (à partager avec les sortie)	0 (sauf avec carte extension)	26 (à partager avec les entrée)	0 (sauf avec carte extension)
Arduino Uno	14	6	14	6
Arduino Mega	54	16	54	15

Nom	Module sonore	Stockage	Langage de programmation	Consommation électrique
Raspberry Pi	Intégrée et matériel (prise jack)	Intégrée (microSD)	Multiple (python, C, Java …)	2 W en fonctionnement
Arduino Uno	Par programme	Module SD à ajouter	C++	0,1W en fonctionnement
Arduino Mega	Par programme	Module SD à ajouter	C++	0,2W en fonctionnement

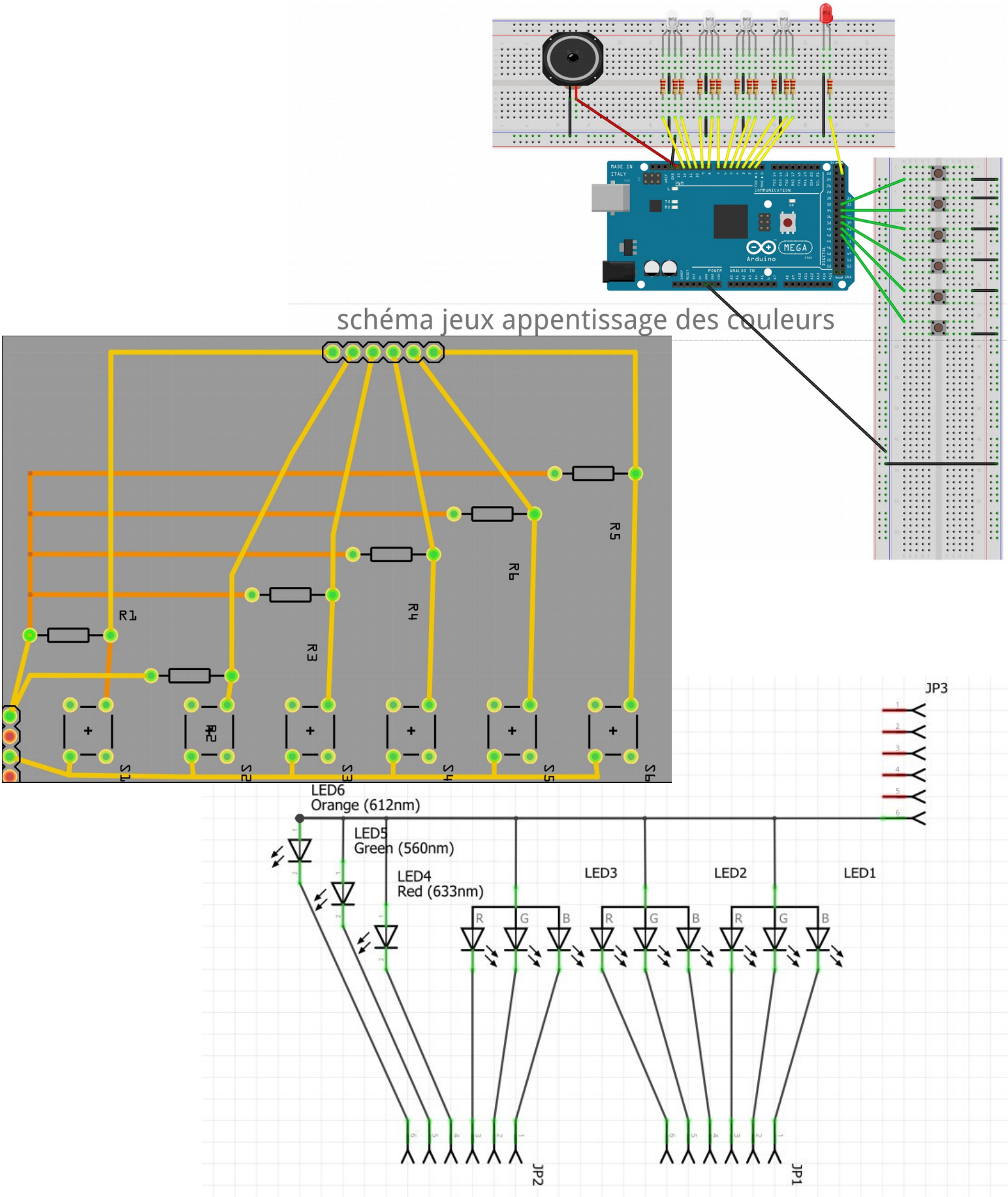
Lors de nos réflexion sur les contrôleurs, nous avons assez vite éliminé le Raspberry Pi en raison de son coût, de sa consommation électrique et de sa fragilité.
Il ne nous est resté que deux contrôleurs possibles, l'Arduino Uno et l'Arduino Méga. Notre choix s'est finalement porté sur l'Arduino Méga, ce qui poussait à faire ce choix est le fait que l'Arduino Méga est capable de gérer une plus grande quantité d'information, tout en ayant plus d'entrés et de sorties.

2) Partie Audio

L'Arduino ne disposant pas de module audio, la partit audio est géré par un librairie dans le code du programme (voir la suite). Pour nous permettre d'obtenir un son assez fort on est obligé de l'amplifié.
On a choisi d'utiliser des AOP (amplificateur opérationnel) pour amplifier le signal audio. Après des recherche notre choix c'est porté sur des amplificateurs d'une puissance de 2,5W avec une alimentation en 5V. Nous n'avons pas réussi à utiliser les amplificateurs. Nous n'avons pas réussi car l'Arduino ne produit pas une intensité suffisant pour alimenter l'amplificateur.

Image ampli

B) Schéma électronique des LED et des boutons.



A) Code Arduino

Le contrôleur ayant été choisi, il faut aussi le programmer. Pour programmer l'Arduino, nous avons dut passer par son IDE. Le code de l'Arduino est écrit en C++, puis compiler par l'IDE à l'aide duc compilateur avr-g++ (appartenant à la famille des GCC). Notre code se découpe en 3 grande partit. Dans un premiers temps, l'initialisation des variable, des entrée sortie et de certain module. Ensuite nous parleront du code exécuter en boucle, puis des fonction crée pour simplifier le programme.

1) Initialisation

```
#include <SPI.h>
#include <SD.h> // Inclure la librairie SD
#define SDPIN 30 // Chip Select du lecteur SD
#include <TMRpcm.h>

TMRpcm audio;

const int pinbuton1 = 33;
const int pinbuton2 = 12;
const int pinbuton3 = 37;
const int pinbuton4 = 39;
const int pinbuton5 = 41;
const int pinbuton6 = 35;

const int temps_delais = 5000;

boolean erreur = false;
```

```
const byte PIN_LED_ROUGE = 23;

const byte PIN_LED2_R = 4;
const byte PIN_LED2_G = 5;
const byte PIN_LED2_B = 6;

const byte PIN_LED3_R = 7;
const byte PIN_LED3_G = 7;
const byte PIN_LED3_B = 9;

const byte PIN_LED4_R = 10;
const byte PIN_LED4_G = 11;
const byte PIN_LED4_B = 12;

const byte PIN_LED5_R = 13;
const byte PIN_LED5_G = 14;
const byte PIN_LED5_B = 15;

const byte PIN_LED_BLANC = 22;
```

Dans cette parti du code nous avons définie tout les différente variable utiliser le long de notre code. Les 4 première ligne code corresponde à l'ajout des trois bibliothèque et à l'initialisation de la carte SD. À la ligne 6 on indique eu code que pour utiliser bibliothèque TMRpcm on utilisera le préfixe « audio ». De la ligne 8 à 13 on donne des valeur au pin des bouton utiliser dans le code. En suite nous définissant le temps utiliser pendant les pause du code. Par la suite nous mettons la variable erreur sur faux. Dans la seconde colonne nous indiquons au programme les pin des LEDs à utiliser.

```
void setup() {
  Serial.begin(9600);

  //initialisation boutons
  pinMode(pinbuton1, INPUT);
  pinMode(pinbuton2, INPUT);
  pinMode(pinbuton3, INPUT);
  pinMode(pinbuton4, INPUT);
  pinMode(pinbuton5, INPUT);
  pinMode(pinbuton6, INPUT);

  //initialisation leds
  pinMode(PIN_LED_ROUGE, OUTPUT);
  pinMode(PIN_LED_BLANC, OUTPUT);
  pinMode(PIN_LED2_R, OUTPUT);
  pinMode(PIN_LED2_G, OUTPUT);
  pinMode(PIN_LED2_B, OUTPUT);
  pinMode(PIN_LED3_R, OUTPUT);
  pinMode(PIN_LED3_G, OUTPUT);
  pinMode(PIN_LED3_B, OUTPUT);
```

```
  pinMode(PIN_LED4_R, OUTPUT);
  pinMode(PIN_LED4_G, OUTPUT);
  pinMode(PIN_LED4_B, OUTPUT);
  pinMode(PIN_LED5_R, OUTPUT);
  pinMode(PIN_LED5_G, OUTPUT);
  pinMode(PIN_LED5_B, OUTPUT);

  //initialisation de la partie audio
  audio.speakerPin = 13;
  audio.setVolume(4);
  audio.quality(1);

  // Initialisation de la carte SD
  if (!SD.begin(SDPIN)) {
    Serial.println("SD initialization failed!");
//  S'il y a un soucis "initialization failed!"
s'affichera au moniteur e  t le programme se
mettra en mode erreur
    erreur = true;
  }
}
```

Dans cette partie du code on indique à l'Arduino si elle doit mettre en entrée ou en sortie les pin indiquer. Pour ce faire on met le code dans une fonction appelée « setup » qui indique les entré et les sortie à l'Arduino. Dans la première partit, on indique que l'on met tout les pin des bouton en entré, dans la seconde on met tout les pin des LEDs en sortie. Puis nous indique on à la librairie audio que le haut-parleur se trouve sur le pin 13, que nous mettons le volume 4 (sur 4) et que nous coulons que qualité soit à meilleur possibles. On trouve dans la dernière partit l'initialisation de la carte SD. Dans cette partie il met en communication la SD avec l'Arduino. S'il y a un soucis lors de l'initialisation de la carte SD, le programme se met en erreur, en mettant la variable erreur sur vrai.

2) code exécuter en boucle

```
void loop() {

  if (!erreur){
    //allume les leds

      boolean buton1 = digitalRead(pinbuton1); //
rouge
      boolean buton2 = digitalRead(pinbuton2); //
bleu
      boolean buton3 = digitalRead(pinbuton3); //
vert
      boolean buton4 = digitalRead(pinbuton4); //
jaune
      boolean buton5 = digitalRead(pinbuton5); //
magenta
      boolean buton6 = digitalRead(pinbuton6); //
blanc

      Serial.print("bouton6: ");
      Serial.print(buton6);
      Serial.print("    bouton1: ");
      Serial.println(buton1);

      if (buton1 == 1){
        led_son(PIN_LED_ROUGE,
(char*)"sounds/rouge.wav");
      }

if (buton2 == 1){
       ledRGB_son(0, 0, 255, 2,
(char*)"sounds/bleu.wav");
      }

      if (buton3 == 1){
        ledRGB_son(0, 255, 0, 3,
(char*)"sounds/vert.wav");
      }

      if (buton4 == 1){
        ledRGB_son(255, 255, 0, 4,
(char*)"sounds/jaune.wav");
      }

      if (buton5 == 1){
        ledRGB_son(255, 0, 255, 5,
(char*)"sounds/mangenta.wav");
      }

      if (buton6 == 1){
        led_son(PIN_LED_BLANC,
(char*)"sounds/blanc.wav");
      }

    } else {
      //allume les leds
      digitalWrite(PIN_LED_BLANC, HIGH);
      digitalWrite(PIN_LED_ROUGE, HIGH);
      delay(300);
    }
  }
```

Dans cette partie du code, on traite le code qui sera exécuté en boucle par l'Arduino tant quelle est alimentée. C'est dans cette partie du code que l'on y effectue la gestion d'erreur. Cette gestion s'éfectue dès la 3e ligne de code. En effet cette ligne dit que si la variable erreur est vrai cela met l'Arduino en erreur comme dit à la fin du code. C'est-à-dire qu'elle va faire clignoter 2 LEDs rapidement pour indiquer l'erreur.
Si tout se passe normalement, l'Arduino récupère l'appuie des boutons. Ensuite elle regarde quel bouton a été appuyé puis lance la fonction correspondante au bouton appuyé. Une fois la fonction exécuter elle reviens au début de la boucle « loop ».

3) Les Fonctions

```
  void displayColor(byte r, byte g, byte b, byte
led) {
    // fonction pour l'allumage avec une couleur
donnée
    if (led == 2){
       analogWrite(PIN_LED2_R, ~r);
       analogWrite(PIN_LED2_G, ~g);
       analogWrite(PIN_LED2_B, ~b);
    }

    if (led == 3){
       analogWrite(PIN_LED3_R, ~r);
       analogWrite(PIN_LED3_G, ~g);
       analogWrite(PIN_LED3_B, ~b);
    }
    if (led == 4){
       analogWrite(PIN_LED4_R, ~r);
       analogWrite(PIN_LED4_G, ~g);
       analogWrite(PIN_LED4_B, ~b);
    }

    if (led == 5){
       analogWrite(PIN_LED5_R, ~r);
       analogWrite(PIN_LED5_G, ~g);
       analogWrite(PIN_LED5_B, ~b);
    }
}
```

Cette fonction sert à allumer les LEDs RGB se la couleur voulue. Pour cela, on lui donne une LED à allumer avec le numéro correspondant (de 2 à 5), puis la couleur sous forme RGB. Avec pour chacune des couleur primaires (rouge, vert et bleu) une valeur étant compris entre 0 et 255.

```
void ledRGB_son(byte r, byte g, byte b, byte led,
char* son) {
  displayColor(~r, ~g, ~b, ~led);
  audio.play(son);
  delay(temps_delais);
  displayColor(0, 0, 0, ~led);
}
```

```
void led_son(byte pinLed, char* son) {
  digitalWrite(pinLed, HIGH);
  audio.play(son);
  delay(temps_delais);
  digitalWrite(pinLed, LOW);
}
```

Nous avons au-dessus les deux fonction régissant le son et les LED. Une et destiner aux LEDs unicolore et l'autre aux LEDs RGB. Nous allons commencer par celle pour les LED RGB. Nous avons appelé cette fonction « ledRGB_son ». Cette fonction prend en entrée, le numéro de la LED, les valeur RGB pour la couleur et l'emplacement du fichier son dans la carte SD. À partir de c'est entrée la fonction fait appelle à la fonction vue précédemment et lance le son à l'aide d'une librairie. Ensuit l'Arduino fait une pause le temps que le fichier audio soit lue, puis elle éteint la LED.
La seconde fonction fait strictement la même chose que la première à l'extption qu'elle ne prend que le pin de la LED et pas de couleur RGB. Elle n'utilise pas non plus la fonction « displaycolor ».

4) Code Complet

Nous avons réalisé pendant ce TPE deux version du code, nous vous avons présentée la première. Dans cette partis on va inclure les deux version du code.

1ere version :

```
#include <SPI.h>
#include <SD.h> // Inclure la librairie SD
#define SDPIN 30 // Chip Select du lecteur SD
#include <TMRpcm.h>

TMRpcm audio;
File myFile;

const int pinbuton1 = 4; //33
const int pinbuton2 = 12;
const int pinbuton3 = 37;
const int pinbuton4 = 39;
const int pinbuton5 = 41;
const int pinbuton6 = 2;//35

const byte temps_delais = 5000;

const byte PIN_LED_ROUGE = 10; //23

const byte PIN_LED2_R = 33; //4
const byte PIN_LED2_G = 5;
const byte PIN_LED2_B = 6;

const byte PIN_LED3_R = 7;
const byte PIN_LED3_G = 10; //8
const byte PIN_LED3_B = 9;

const byte PIN_LED4_R = 23; //10
const byte PIN_LED4_G = 11;
const byte PIN_LED4_B = 35; //12

const byte PIN_LED5_R = 13;
const byte PIN_LED5_G = 14;
const byte PIN_LED5_B = 15;

const byte PIN_LED_BLANC = 8; //22

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  //initialisation boutons
  pinMode(pinbuton1, INPUT);
  pinMode(pinbuton2, INPUT);
  pinMode(pinbuton3, INPUT);
  pinMode(pinbuton4, INPUT);
  pinMode(pinbuton5, INPUT);
  pinMode(pinbuton6, INPUT);

  //initialisation leds
  pinMode(PIN_LED_ROUGE, OUTPUT);
  pinMode(PIN_LED_BLANC, OUTPUT);
  pinMode(PIN_LED2_R, OUTPUT);
  pinMode(PIN_LED2_G, OUTPUT);
  pinMode(PIN_LED2_B, OUTPUT);
  pinMode(PIN_LED3_R, OUTPUT);
  pinMode(PIN_LED3_G, OUTPUT);
  pinMode(PIN_LED3_B, OUTPUT);
  pinMode(PIN_LED4_R, OUTPUT);
  pinMode(PIN_LED4_G, OUTPUT);
  pinMode(PIN_LED4_B, OUTPUT);
  pinMode(PIN_LED5_R, OUTPUT);
  pinMode(PIN_LED5_G, OUTPUT);
  pinMode(PIN_LED5_B, OUTPUT);
  //initialisation de la partie audio
  audio.speakerPin = 13;
  audio.setVolume(4); // gestion du volume de 0 à
7
  audio.quality(1);

  // Initialisation de la carte SD
  if (!SD.begin(SDPIN)) {
    Serial.println("SD initialization failed!");
//  S'il y a un soucis "initialization failed!"
s'affichera au moniteur
    return;
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  boolean buton1 = digitalRead(pinbuton1);
//rouge
  boolean buton2 = digitalRead(pinbuton2); //bleu
  boolean buton3 = digitalRead(pinbuton3); //vert
  boolean buton4 = digitalRead(pinbuton4);
//jaune
  boolean buton5 = digitalRead(pinbuton5);
//magenta
  boolean buton6 = digitalRead(pinbuton6);
//blanc

  Serial.print("bouton6: ");
  Serial.print(buton6);
  Serial.print("    bouton1: ");
  Serial.println(buton1);

  if (buton1 == 1){
    digitalWrite(PIN_LED_ROUGE, HIGH);
    audio.play("sounds/rouge.wav");
    delay(temps_delais); //mettre une variable à
place de 5000
    digitalWrite(PIN_LED_ROUGE, LOW);
  }

  if (buton2 == 1){
    displayColor(0, 0, 255, 2);
    //led bleu
    audio.play("sounds/bleu.wav");
    delay(temps_delais);
    displayColor(0, 0, 0, 2);
  }

  if (buton3 == 1){
    displayColor(0, 255, 0, 3);
    //led vert
    audio.play("sounds/vert.wav");
    delay(temps_delais);
    displayColor(0, 0, 0, 3);
  }

  if (buton4 == 1){
    displayColor(255, 255, 0, 4);
    //led jaune
    audio.play("sounds/jaune.wav");
    delay(temps_delais);
    displayColor(0, 0, 0, 4);
  }
```

```
  if (buton5 == 1){
    displayColor(255, 0, 255, 5);
    //led magenta
    audio.play("sounds/magenta.wav");
    delay(temps_delais);
    displayColor(0, 0, 0, 5);
  }

  if (buton6 == 1){
    digitalWrite(PIN_LED_BLANC, HIGH);
    //led blanc
    audio.play("sounds/blanc.wav");
    delay(temps_delais);
    digitalWrite(PIN_LED_BLANC, LOW);
  }

  }

  void displayColor(byte r, byte g, byte b, byte
led) {
    // fonction pour l'allumage avec une couleur
donnée
    if (led == 2){
       analogWrite(PIN_LED2_R, ~r);
       analogWrite(PIN_LED2_G, ~g);
       analogWrite(PIN_LED2_B, ~b);
    }

    if (led == 3){
       analogWrite(PIN_LED3_R, ~r);
       analogWrite(PIN_LED3_G, ~g);
       analogWrite(PIN_LED3_B, ~b);
    }

    if (led == 4){
       analogWrite(PIN_LED4_R, ~r);
       analogWrite(PIN_LED4_G, ~g);
       analogWrite(PIN_LED4_B, ~b);
    }

    if (led == 5){
       analogWrite(PIN_LED5_R, ~r);
       analogWrite(PIN_LED5_G, ~g);
       analogWrite(PIN_LED5_B, ~b);
    }
}
```

2e version (celle présenté) :

```
#include <SPI.h>
#include <SD.h> // Inclure la librairie SD
#define SDPIN 30 // Chip Select du lecteur SD
#include <TMRpcm.h>

TMRpcm audio;

const int pinbuton1 = 33;
const int pinbuton2 = 12;
const int pinbuton3 = 37;
const int pinbuton4 = 39;
const int pinbuton5 = 41;
const int pinbuton6 = 35;

const int temps_delais = 5000;

boolean erreur = false;

const byte PIN_LED_ROUGE = 23;

const byte PIN_LED2_R = 4;
const byte PIN_LED2_G = 5;
const byte PIN_LED2_B = 6;

const byte PIN_LED3_R = 7;
const byte PIN_LED3_G = 7;
const byte PIN_LED3_B = 9;

const byte PIN_LED4_R = 10;
const byte PIN_LED4_G = 11;
const byte PIN_LED4_B = 12;

const byte PIN_LED5_R = 13;
const byte PIN_LED5_G = 14;
const byte PIN_LED5_B = 15;

const byte PIN_LED_BLANC = 22;
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  //initialisation boutons
  pinMode(pinbuton1, INPUT);
  pinMode(pinbuton2, INPUT);
  pinMode(pinbuton3, INPUT);
  pinMode(pinbuton4, INPUT);
  pinMode(pinbuton5, INPUT);
  pinMode(pinbuton6, INPUT);

  //initialisation leds
  pinMode(PIN_LED_ROUGE, OUTPUT);
  pinMode(PIN_LED_BLANC, OUTPUT);
  pinMode(PIN_LED2_R, OUTPUT);
  pinMode(PIN_LED2_G, OUTPUT);
  pinMode(PIN_LED2_B, OUTPUT);
  pinMode(PIN_LED3_R, OUTPUT);
  pinMode(PIN_LED3_G, OUTPUT);
  pinMode(PIN_LED3_B, OUTPUT);
  pinMode(PIN_LED4_R, OUTPUT);
  pinMode(PIN_LED4_G, OUTPUT);
  pinMode(PIN_LED4_B, OUTPUT);
  pinMode(PIN_LED5_R, OUTPUT);
  pinMode(PIN_LED5_G, OUTPUT);
  pinMode(PIN_LED5_B, OUTPUT);

  //initialisation de la partie audio
  audio.speakerPin = 13;
  audio.setVolume(4); // gestion du volume
  audio.quality(1);

  // Initialisation de la carte SD
  if (!SD.begin(SDPIN)) {
    Serial.println("SD initialization failed!");
//  S'il y a un soucis "initialization failed!"
s'affichera au moniteur e  t le programme se
mettra en mode erreur
    erreur = true;
  }
}
```

```
void loop() {

  if (!erreur){
    //allume les leds

    // put your main code here, to run
repeatedly:
      boolean buton1 = digitalRead(pinbuton1); //
rouge
      boolean buton2 = digitalRead(pinbuton2); //
bleu
      boolean buton3 = digitalRead(pinbuton3); //
vert
      boolean buton4 = digitalRead(pinbuton4); //
jaune
      boolean buton5 = digitalRead(pinbuton5); //
magenta
      boolean buton6 = digitalRead(pinbuton6); //
blanc

      Serial.print("bouton6: ");
      Serial.print(buton6);
      Serial.print("    bouton1: ");
      Serial.println(buton1);

      if (buton1 == 1){
        led_son(PIN_LED_ROUGE,
(char*)"sounds/rouge.wav");
      }

      if (buton2 == 1){
       ledRGB_son(0, 0, 255, 2,
(char*)"sounds/bleu.wav");
      }

      if (buton3 == 1){
        ledRGB_son(0, 255, 0, 3,
(char*)"sounds/vert.wav");
      }

      if (buton4 == 1){
        ledRGB_son(255, 255, 0, 4,
(char*)"sounds/jaune.wav");
      }

      if (buton5 == 1){
        ledRGB_son(255, 0, 255, 5,
(char*)"sounds/mangenta.wav");
      }

      if (buton6 == 1){
        led_son(PIN_LED_BLANC,
(char*)"sounds/blanc.wav");
      }

    } else {
      //allume les leds
      digitalWrite(PIN_LED_BLANC, HIGH);
      digitalWrite(PIN_LED_ROUGE, HIGH);
      delay(300);
    }
  }
```

```
  void displayColor(byte r, byte g, byte b, byte
led) {
    // fonction pour l'allumage avec un couleur
donné
    if (led == 2){
       analogWrite(PIN_LED2_R, ~r);
       analogWrite(PIN_LED2_G, ~g);
       analogWrite(PIN_LED2_B, ~b);
    }

    if (led == 3){
       analogWrite(PIN_LED3_R, ~r);
       analogWrite(PIN_LED3_G, ~g);
       analogWrite(PIN_LED3_B, ~b);
    }

    if (led == 4){
       analogWrite(PIN_LED4_R, ~r);
       analogWrite(PIN_LED4_G, ~g);
       analogWrite(PIN_LED4_B, ~b);
    }

    if (led == 5){
       analogWrite(PIN_LED5_R, ~r);
       analogWrite(PIN_LED5_G, ~g);
       analogWrite(PIN_LED5_B, ~b);
    }
}

void ledRGB_son(byte r, byte g, byte b, byte led,
char* son) {
  displayColor(~r, ~g, ~b, ~led);
  audio.play(son);
  delay(temps_delais);
  displayColor(0, 0, 0, ~led);
}

void led_son(byte pinLed, char* son) {
  digitalWrite(pinLed, HIGH);
  audio.play(son);
  delay(temps_delais);
  digitalWrite(pinLed, LOW);
}
```

V – Bibliographie

Pour réaliser ce TP nous nous somme aider des plusieurs source comme de plusieurs logiciel et service. Nous nous sommes aidés des site Web suivant :

- framapad.org (Traitement de texte en ligne collaboratif)
- arduino.cc (Site officiel et documentions du code et des carte Arduino)
- idehack.com (Blog de création de projet à partir d'Arduino)
- bases-brevets.inpi.fr (Site Web de l'INPI [gestion des brevet français])

Nous nous sommes aussi aider de différent logiciel tel que :

- Mind View (logiciel de carte mental)
- Fritzing (Logiciel de modélisation de schéma électronique)
- LibreOffice (Suite de logiciel Bureautique)
- Arduino IDE (IDE de développement pour carte Arduino)
- Blender 3D (Logiciel de modélisation 3D)